CAPÍTULO 2:

HOLA MUNDO

Hola mundo

¿Qué es lo primero que hacemos al entrar a un lugar? Saludamos... en la jerga de la programación, ese primer contacto, se le llama programa "Hola mundo", o "Hello world", es el primer ejercicio de ingreso que nos asegura que todo está funcionando correctamente. En programación suele ser la visualización en pantalla del saludo "Hola mundo".

En este capítulo armaremos una estructura inicial, descargaremos el IDE Arduino, conoceremos sus partes y veremos como es el procedimiento para cargar programaciones a nuestra placa.

Consejo:

A la hora de descargar el IDE Arduino, tené en cuenta el sistema operativo que tenés. Podés descargarlo para Linux, Windows 7 en adelante, y Mac OS X.

En Windows 8 y 10, podés descargar la app en el Store.

¿Qué es Arduino?

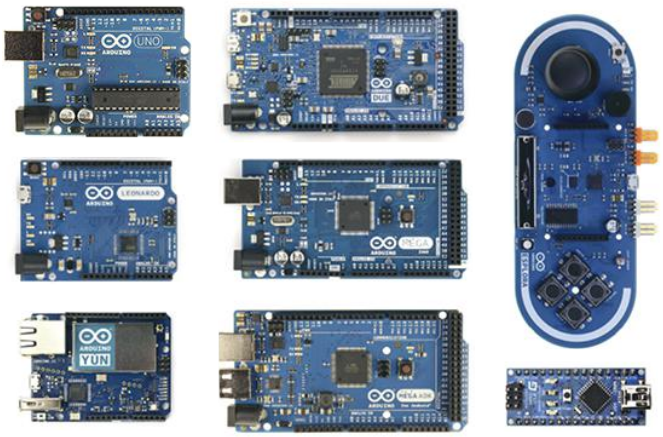
Se le llama Arduino a cuatro cosas que están relacionadas entre si. En primer lugar, se le llama Arduino a la empresa de hardware libre que produce las placas que integran circuitos impresos y microcontroladores.

Arduino es además el entorno de programación (IDE) en el cual programaremos nuestras creaciones.

También se llama Arduino la comunidad de usuarios que sostiene el proyecto.

Y finalmente Arduino también se le dice a esas placas que mencionamos dentro de nuestra caja.

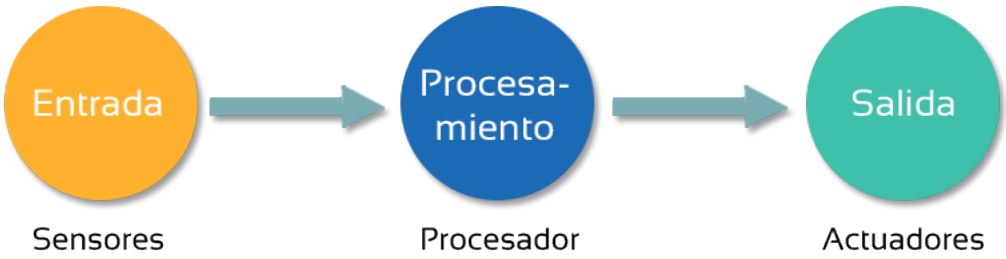
La placa de Arduino



La empresa produce diferentes placas Arduino para diversas necesidades según el tamaño de los proyectos. Cada placa Arduino tiene distintas posibilidades de conectarse con otros componentes y diferentes capacidades de procesamiento.

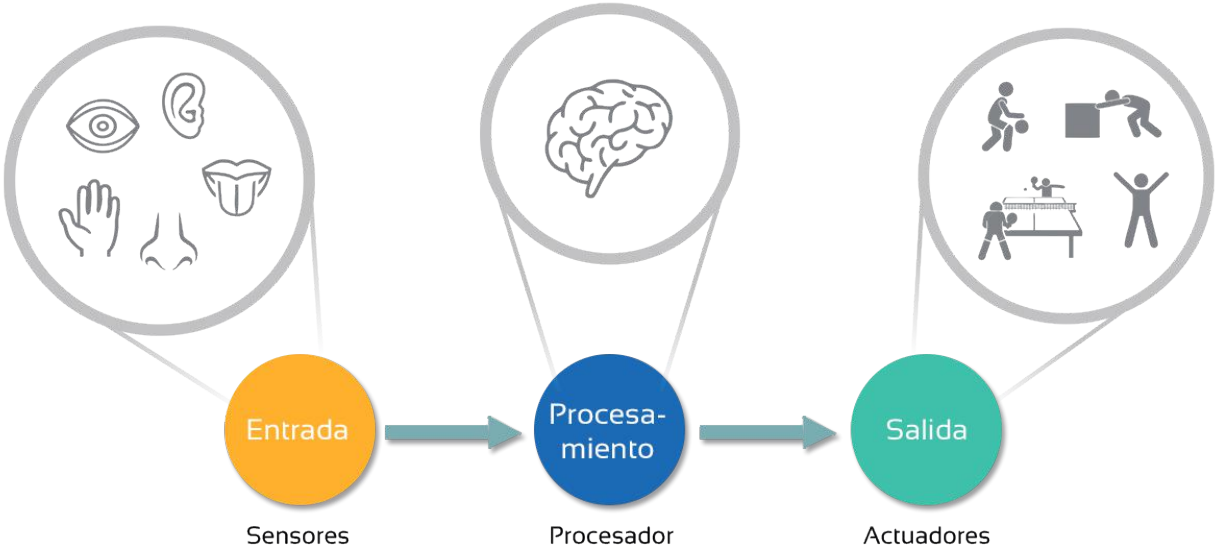
Nosotros, durante este curso, vamos a trabajar con la placa Arduino Uno, que es la más conocida, y muy adecuada para empezar a realizar proyectos de complejidad media.

Cuando hablamos de una placa Arduino estamos frente a un sistema de entrada - procesamiento - salida.

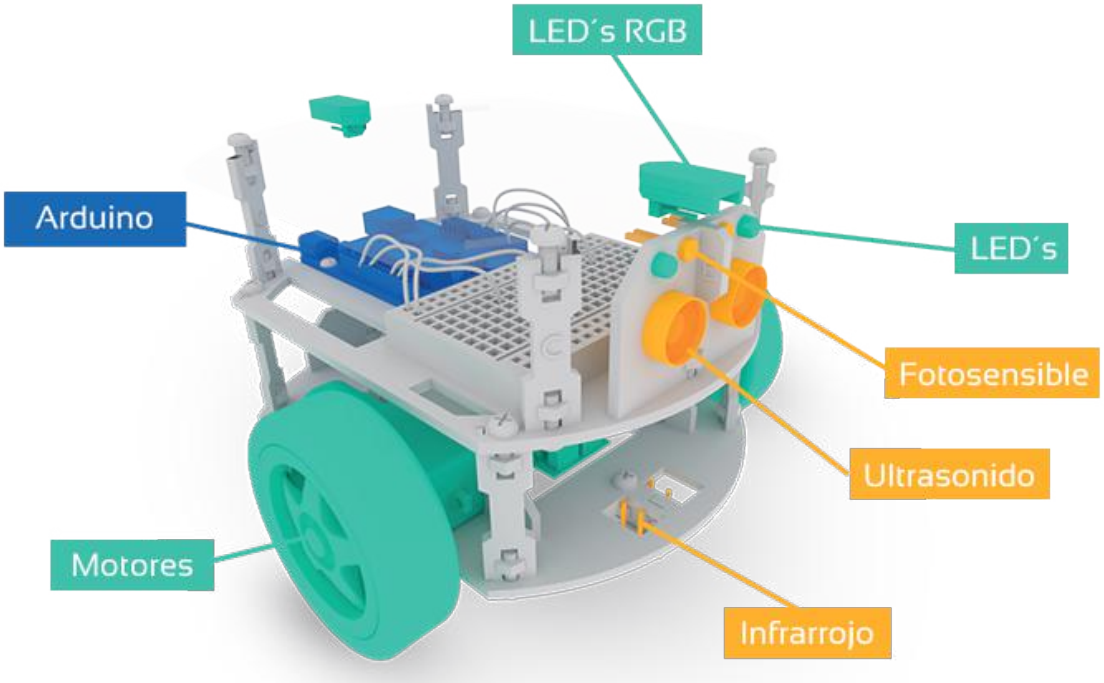


Tiene la capacidad de ingresar información del entorno a través de sensores, comprender y procesar esa información (esa es la parte que debemos programar nosotros) y luego realizar un trabajo sobre ese mismo entorno por medio de actuadores. Es decir que se dan las tres etapas de entrada, procesamiento y salida.

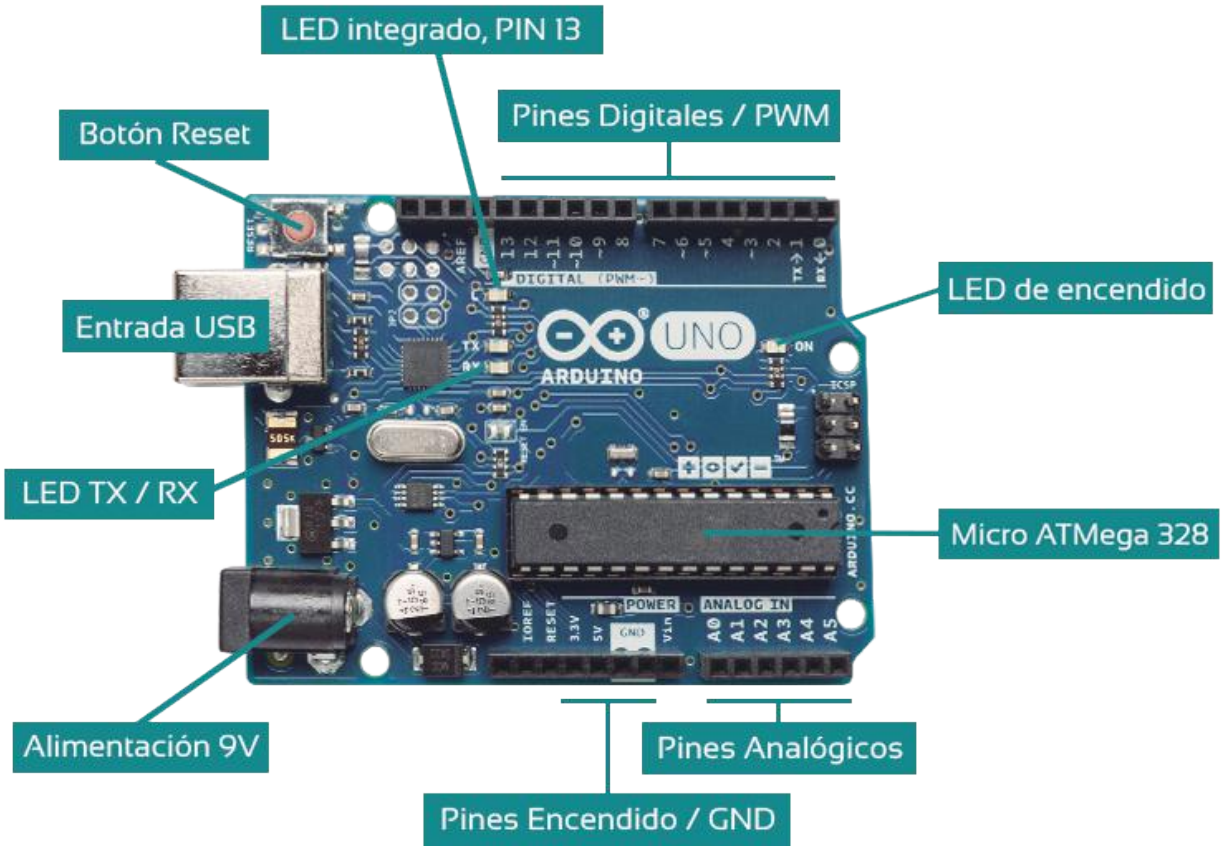
Hagamos una comparación de estos tres conceptos con el cuerpo humano: los sensores son como nuestros sentidos... se encargan de recibir la información del entorno (luces, sonidos, distancias...) El Arduino sería nuestro cerebro, el cual procesa la información que recibe de los sentidos. Y los actuadores vienen a ser las respuestas que da el cuerpo como respuesta a la información recibida.



Viendo nuestro robot en su totalidad vamos encontrar distintos sensores: el sensor ultrasónico, que nos permite medir distancias o movimientos... fotorresistencias, que son sensibles a la luz... y sensores infrarrojos, que nos permiten codificar por ejemplo diferentes colores... para el procesamiento tenemos nuestra placa arduino, la cual va a ir recibiendo la informaciones de nuestros sensores y procesandola para que nuestros actuadores trabajen... y los actuadores de nuestro robot son: los motores, con sus ruedas, las luces Led y los leds RGB.



A continuación veremos cada una de las partes que componen a una placa de Arduino para que puedas familiarizarte antes que nada con su ubicación. Luego, en la página siguiente encontrarás un listado detallado de las mismas junto con una breve explicación de su utilidad.



ENTRADA USB

Se usa para alimentar y cargar programas a nuestro Arduino. También es esencial para la comunicación con el IDE Arduino, mediante las instrucciones Serial.begin(), Serial.println()...

ALIMENTACIÓN 9V

Por medio de un cable de corriente eléctrica, y su respectivo adaptador, podemos alimentar a nuestro Arduino con una bateria, en este caso, de 9v.

Nota: No uses fuentes de alimentación superiores a 20v ya que pueden dañar la placa. La tensión recomendada es entre 7v y 12v.

PINES DIGITALES / PWM

Estos pines van del 0 al 13 y se pueden utilizar tanto para **entradas digitales** (por ejemplo, si se oprime un botón) como para **salidas digitales** (como encender un LED). Se usan a través de las funciones digitalWrite() y digitalRead().

Notarás que cinco de estos pines digitales (3, 6, 9, 10 y 11 respectivamente) están señalados con una pequeña onda (~). Esto significa que son PWM (permiten una salida semi analógica), por lo que pueden usarse con la función analogWrite().

MICROCONTROLADOR ATMEGA 328

Sobre él vamos a programar. Este microncontrolador es el cerebro de nuestro Arduino.

PINES ANALÓGICOS

Estos pines van de A0 a A5 y pueden leer la señal de un sensor analógico (como ser un sensor infrarrojo) y convertirlo en un valor digital que podemos leer. Se usan a través de analogRead().

PIN ENCENDIDO

Son los suministros de cinco voltios (5V) y tres punto tres voltios (3.3V) de potencia.

PIN GND

GND es la abreviatura de "tierra", GROUND en Inglés. Hay varios pines GND en el Arduino, cualquiera de ellos puede ser utilizado para conectar a tierra el circuito.

PIN 13 - LED INTEGRADO

Es el único componente que actúa como dispositivo de salida incorporado a la placa Arduino. Este led es muy útil para la depuración.

BOTÓN RESET

Al presionar este botón se conectará temporalmente el pin de reset a tierra, logrando que se reinicie cualquier código cargado en el Arduino. Esto puede ser muy útil si el código no se repite, pero quieres que se ejecute varias veces para realizar pruebas.

LEDS ENCENDIDA

Este LED debe encenderse cada vez que se conecte la placa Arduino a una toma eléctrica. Si esta luz no se enciende, hay una alta probabilidad de que algo anda mal.

LEDS RX/TX

Estos LEDs nos darán algunas indicaciones visuales siempre que nuestro Arduino esté recibiendo o transmitiendo datos (por ejemplo, la carga de un nuevo programa).

MICROCONTROLADOR ATMEGA 328

Sobre él vamos a programar. Este microncontrolador es el cerebro de nuestro Arduino.

El IDE de Arduino

El entorno de programación o IDE (Entorno Integral de Desarrollo), es un software pensado para programar, el cual viene preparado para asistir al programador en la creación de nuevos códigos. Es decir que el entorno nos permite escribir nuestras programaciones, compilarlas y ejecutarlas de una forma cómoda y simple.

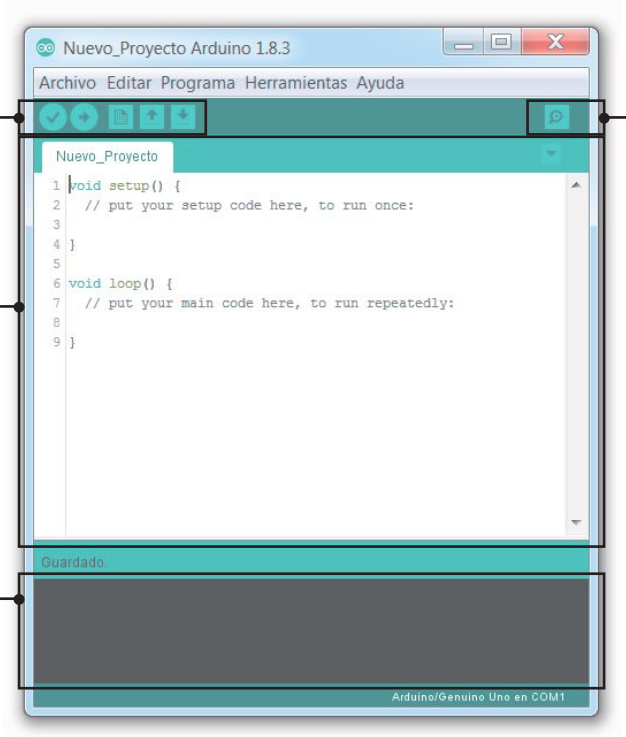
Arduino nos ofrece su propio IDE de forma gratuita. Para acceder a él simplemente debemos ingresar a su página web, descargar el archivo e instalarlo en nuestra computadora (http://arduino.cc/En/Main/Software) Sigue este link para descargarlo.

Partes del IDE Arduino:

Barra de acciones
Aquí tenemos las opciones de verificar nuestro código y para subirlo cuando queramos probarlo.

Programación
En esta parte vemos el nombre de nuestro proyecto y podemos escribir el código de la programación según el problema que queremos resolver.

Pantalla de mensajes
En esta espacio el IDE nos informa los errores, o bien la correcta compilación y carga de los programas.



Monitor serie
Este botón nos abre una ventana desde la que podemos comunicarnos con la pc por medio del puerto serie USB.

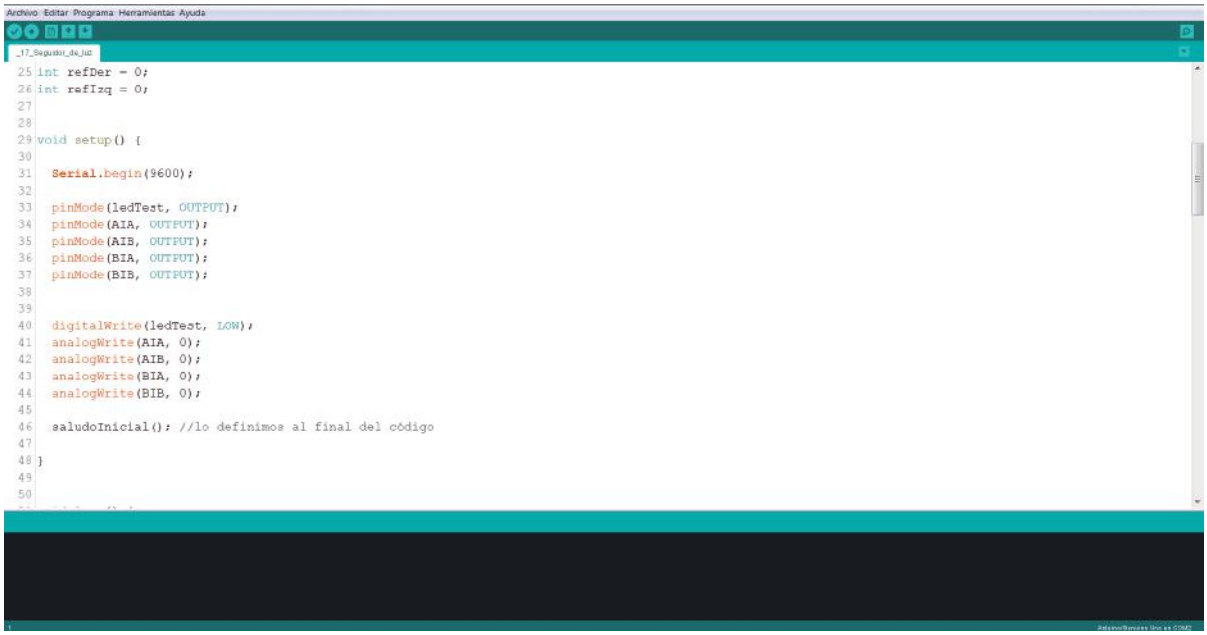
Carga de programación

A continuación veremos la forma adecuada de realizar la carga de una programación en el entorno IDE Arduino.

Escritura

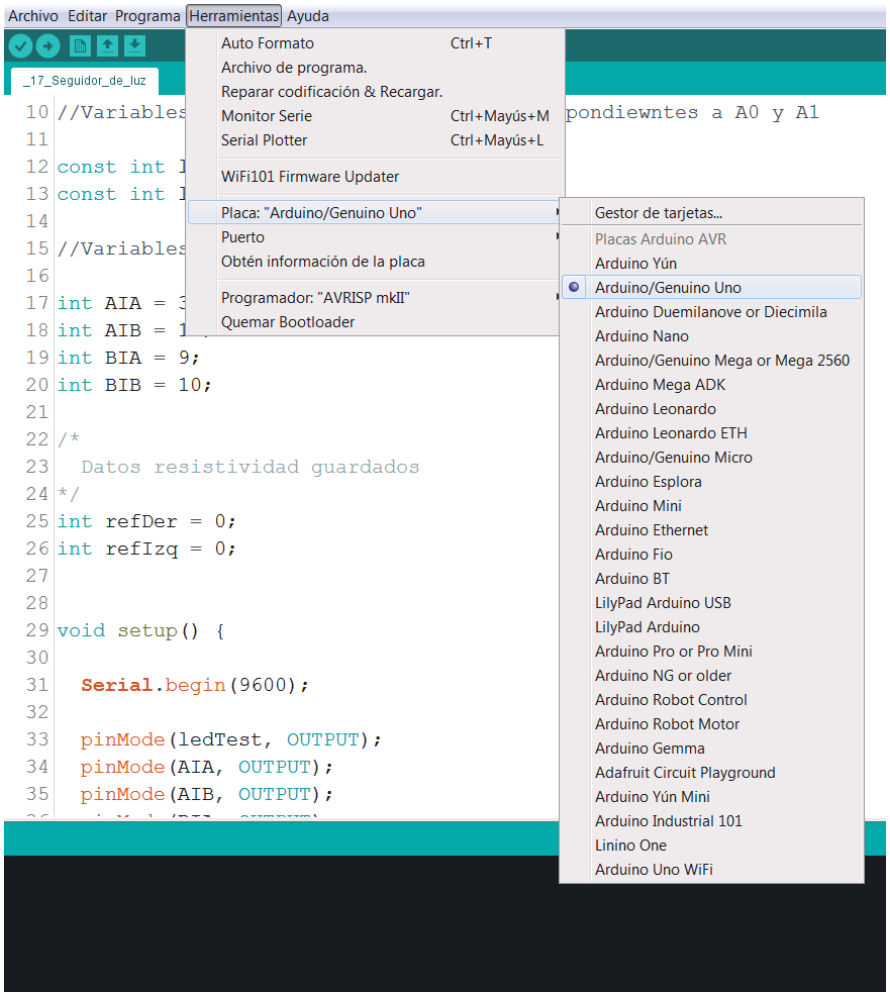
Lo primero que debemos hacer es desarrollar el código en su totalidad. Es recomendable verificar el código antes de subirlo a la placa.

Debemos estar atentos a la declaración de variables, apertura y cierre de llaves y paréntesis, el uso de puntos y comas, etc.



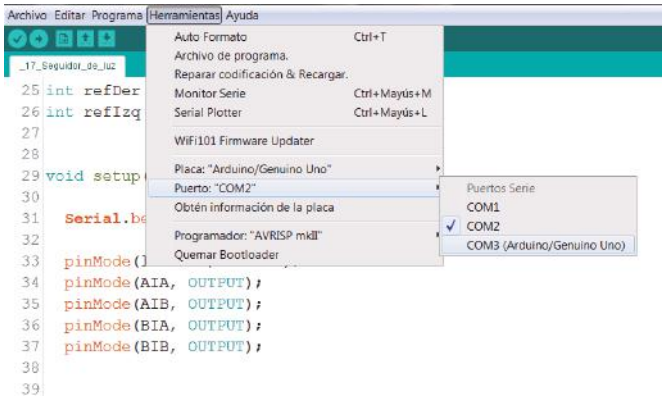
Verificación de la placa

Ingresando a Herramientas/Placa elegimos "Arduino/Genuino Uno".



Verificación de puerto

Es importante que seleccionemos el puerto USB que vamos a usar para la carga de datos. Ingresando a Herramientas/Puerto, elegimos el que diga "(Arduino/Genuino Uno)"

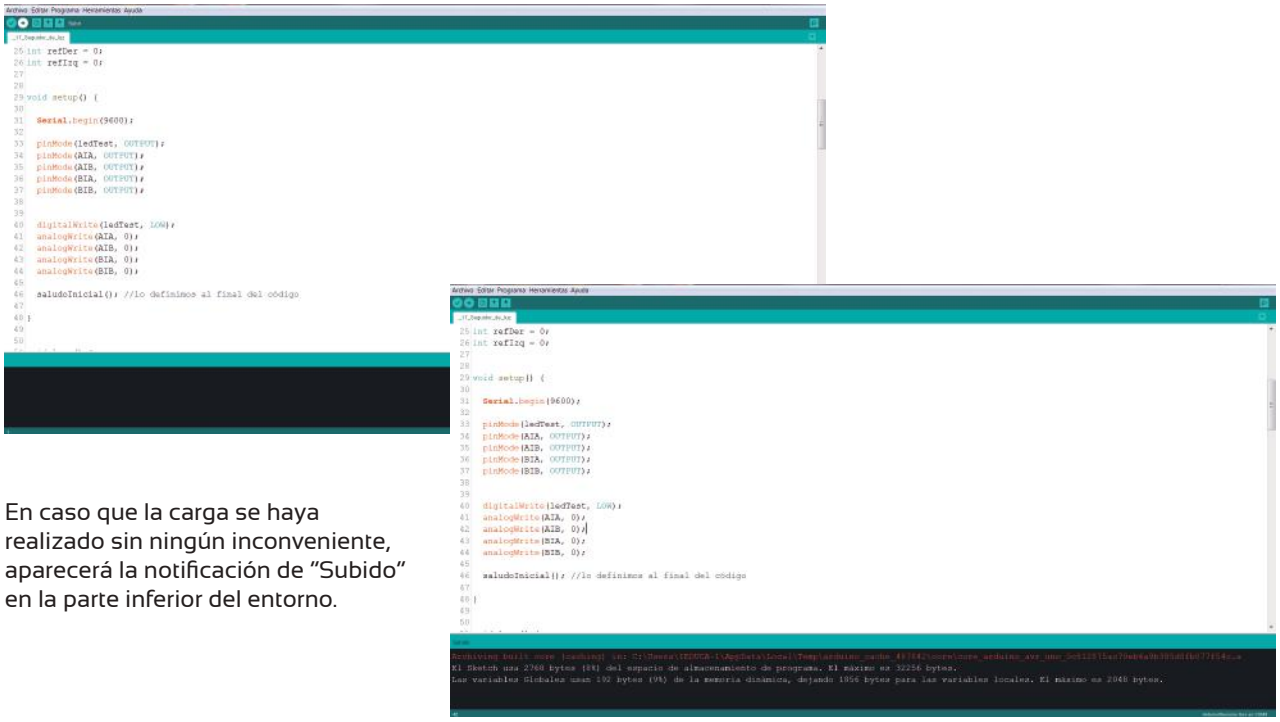


En caso que no reconozca el puerto, deberemos verificar que se haya realizado la correcta instalación de los controladores necesarios para la placa Arquino UNO.

Podemos probar conectando en otros puerto UBS o ingresar a los controladores de software USB de nuestra computadora.

Subir la programación

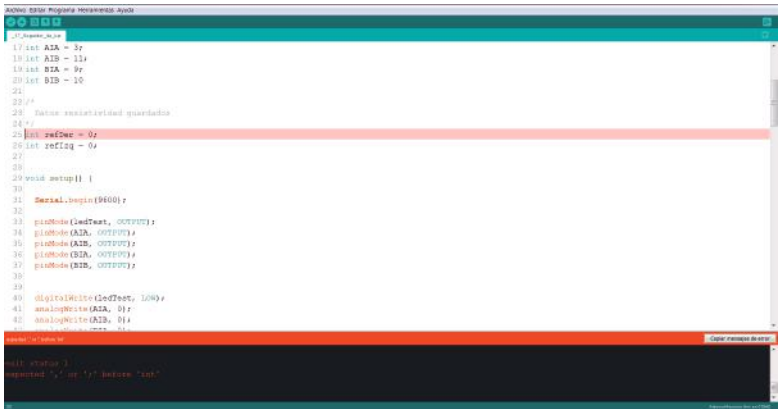
Con el botón "Subir", indicado con una fecha hacia la derecha, cargaremos la programación a nuestra placa.



En caso que la carga se haya realizado sin ningún inconveniente, aparecerá la notificación de "Subido" en la parte inferior del entorno.

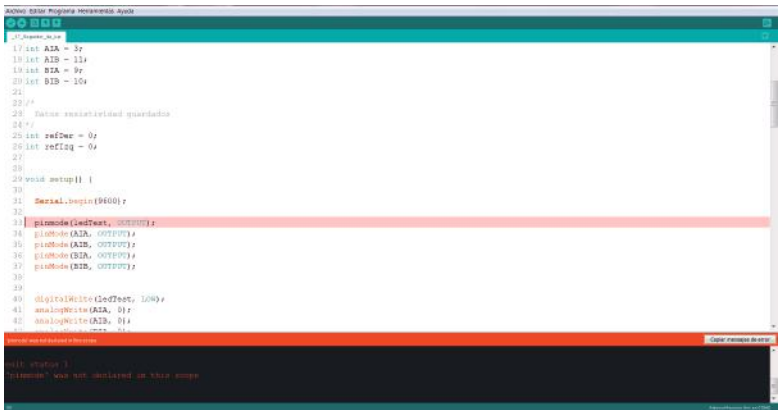
Error en la carga

A continuación presentaremos algunos errores que pueden surgirnos a la hora de cargar una programación:



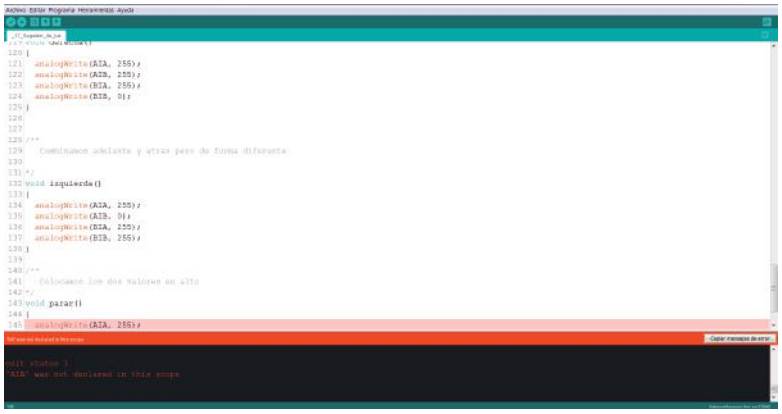
Error de escritura 1:

La ausencia de puntos y comas " ; " el finalizar una línea de programación.



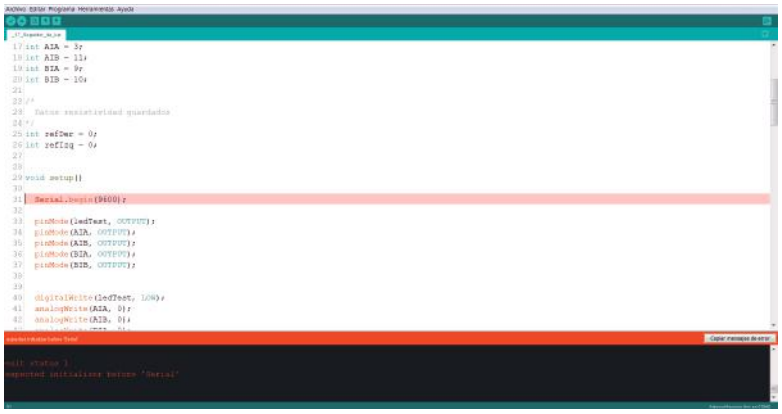
Error de escritura 2:

Si escribimos mal una función o instrucción va a pedirnos que la "declaremos" como una variable. Debemos prestar atención a la escritura de las palabras reservadas.



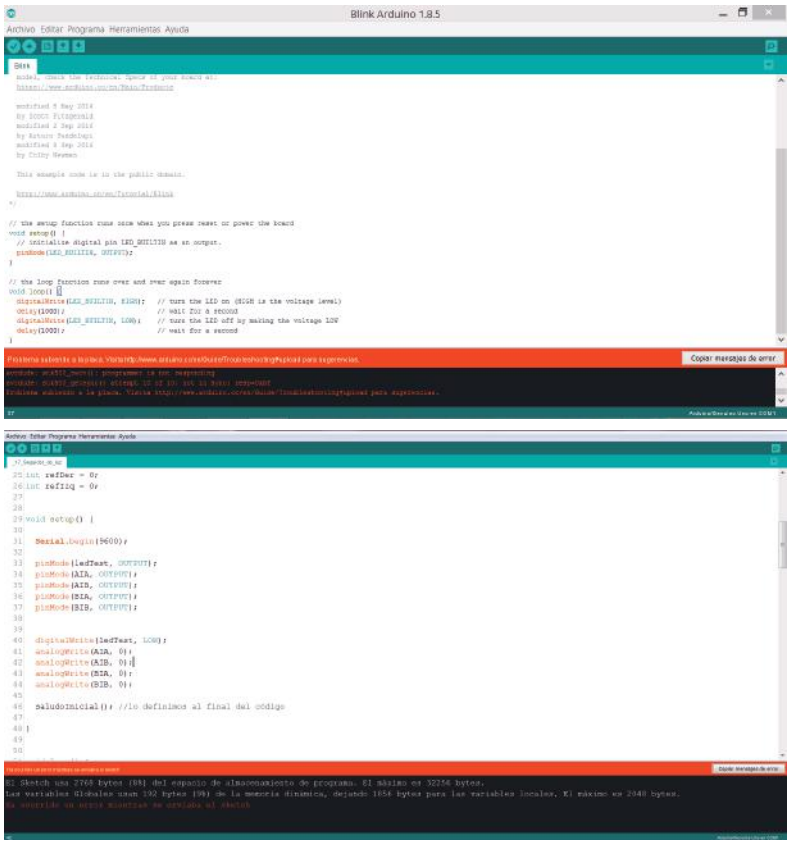
Error de escritura 3:

Debemos declarar bien las variables y prestar atención a la hora de usarlas durante la escritura del código.



Error de escritura 4:

La apertura y cierre incorrectas de las llaves " { } " y paréntesis " ( ) " puede causar problemas en la carga de nuestra programación.



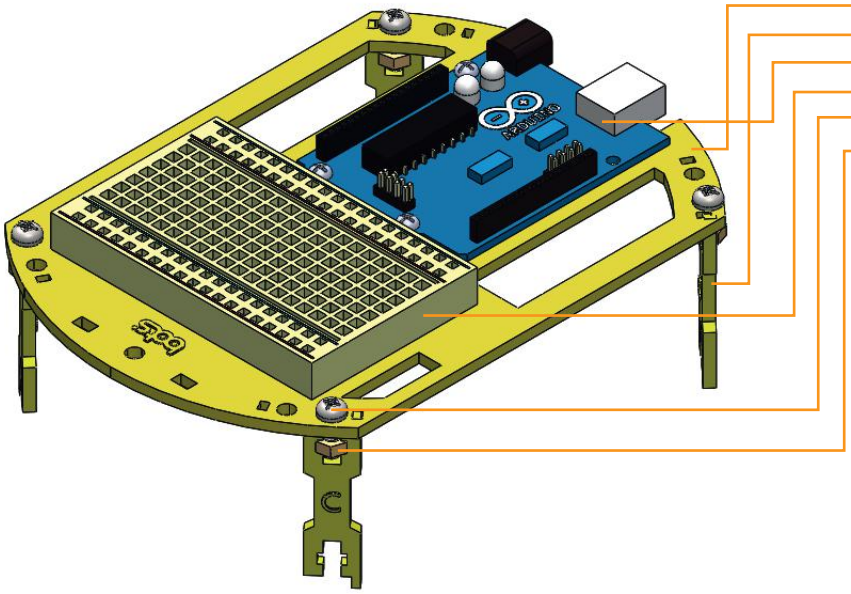
Problema subiendo a la placa:

Debemos verificar que la placa y el puerto estén bien seleccionados. Verifiquemos haciendo click en:

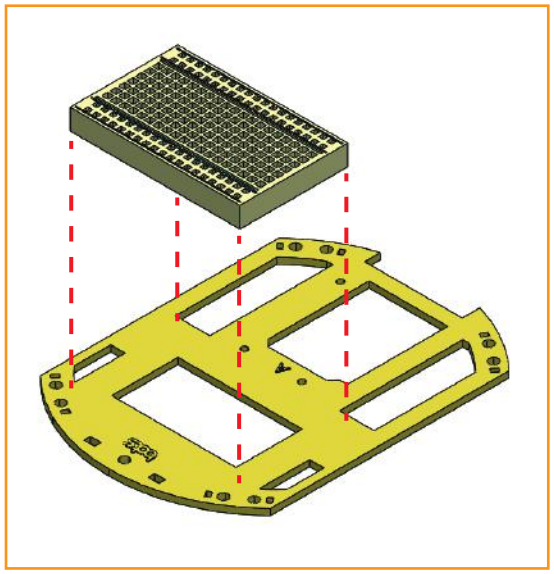
Herramientas/Placa ó
Herramientas/Puerto

Ensamble

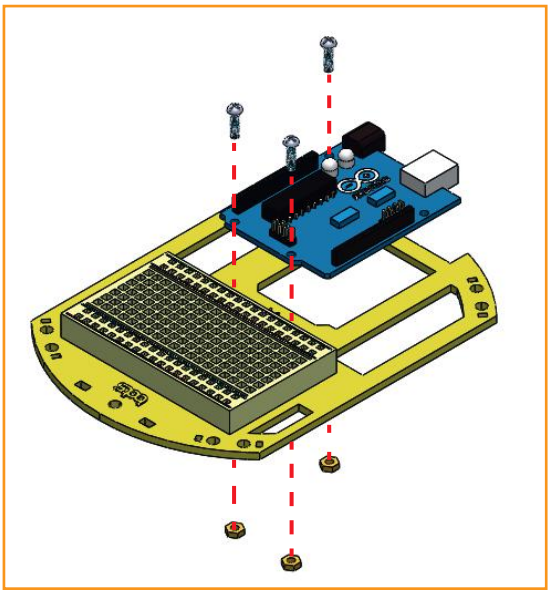
A continuaciòn veremos la estructura y componentes necesarios para realizar el saludo inicial de nuestro Arduino, y una guía de pasos.



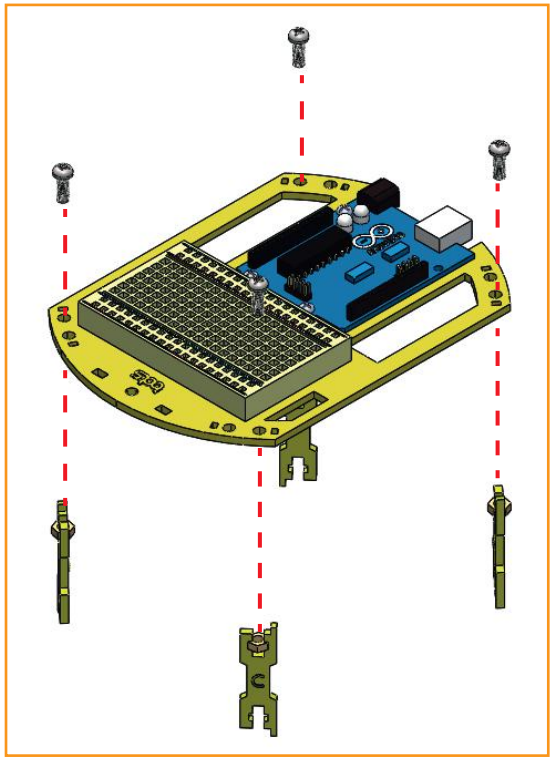
Materiales:
-Placa A
-Columnas C x4 unidades
-Arduino UNO
-Protoboard
-Bulón 4x12mm x 4 unidades
-Tuerca 4mm x 4 unidades



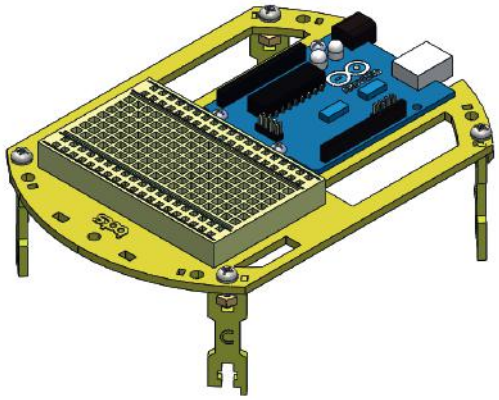
Paso 1:
Retirar la pegatina de la parte inferior de la protoboard y pegarla sobre el chasis A, entre el logo de Bots y la letra "A grabadoa en la placa.



Paso 2:
Colocar la placa Arduino UNO ajustando 3 bulones 3x12mm y sus respectivas tuercas.
ADVERTENCIA: No hacer mucha fuerza al ajustarlos para no dañar la placa.



Paso 3:
Ajustar 4 columnas C a la placa A, por medio de 4 bulones 4x12mm y sus respectivas tuercas.
RECOMENDACIÓN:



Hola mundo

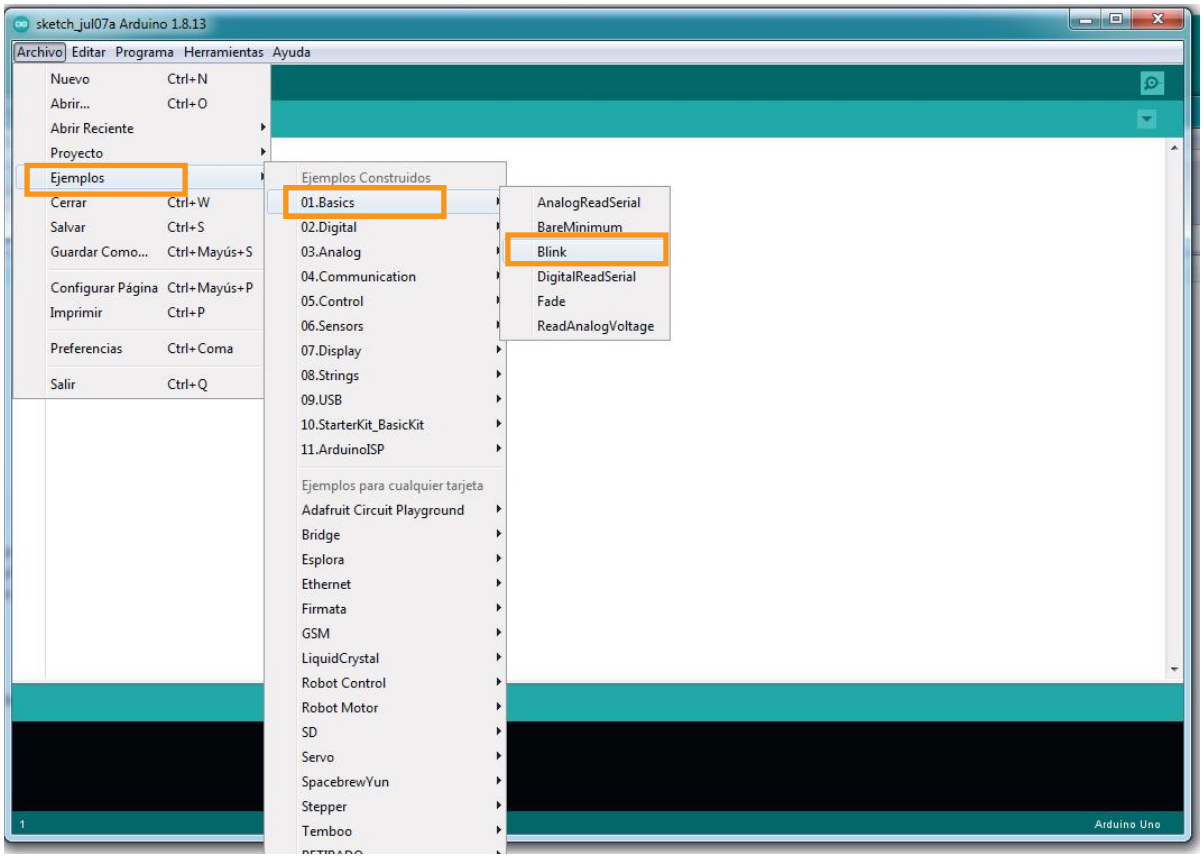
¿Qué es lo primero que hacemos al entrar a un lugar? Saludamos... en la jerga de la programación, ese primer contacto, se le llama programa "Hola mundo", o "Hello world", es el primer ejercicio de ingreso que nos asegura que todo está funcionando correctamente. En programación suele ser la visualización en pantalla el saludo "Hola mundo". Ahora bien, nuestra placa Arduino no está conectada todavía a ningún periférico o actuador, de modo que haremos parpadear el pequeño led integrado de la placa para simbolizar nuestro saludo.

El LED que usaremos se le llama Pin 13, que como dijimos antes, está integrado a la placa. Es el único componente onboard (en la placa) que tiene una función de salida es decir, comunicarse con el usuario por medio de la luz. Muchas veces utilizaremos el Pin 13 para detectar errores o comprobar el correcto funcionamiento de nuestro programa.

Es importante hacer esta verificación ya que podremos ver el correcto funcionamiento de la placa, el cable USB y el IDE Arduino para continuar sin problemas con el curso.

Para recibir nuestro saludo inicial vamos a usar el ensamble que hemos realizado anteriormente y el IDE de Arduino.

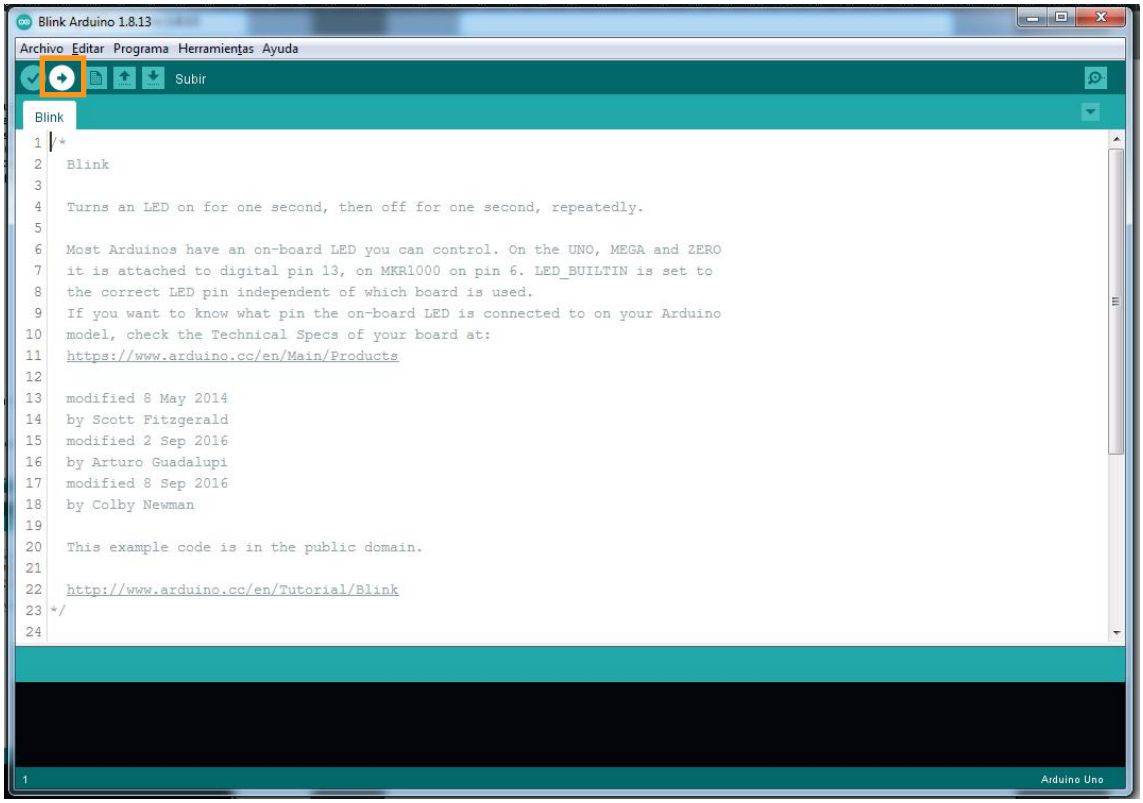
Esta programación es muy simple, ya que se encuentra dentro de nuestro IDE.



1. Abrimos el IDE Arduino para generar nuestro primer programa.

2. Seleccionamos el menú "Archivo" escogemos "Ejemplos" a continuación "01Basics" y por último "Blink". Se abrirá un archivo.

3. Finalmente, debemos "compilar" el código. No es más que cargar la programación al arduino. Para compilar y subir nuestro programa seleccionamos en el menú "Programa" y luego "Subir", o en botón con una flecha indicando hacia la derecha, de la pantalla principal.



Y ahora sí, podemos ver nuestro saludo inicial, el "hola mundo", con el parpadeo del LED del pin 13.

Glosario de programación Arduino

En este apartado encontrarás listadas en orden alfabético las instrucciones de programación utilizadas a lo largo del curso junto con una breve reseña de su utilidad.

analogRead (pin); // lee el voltaje del 'pin' de tipo analógico

analogWrite (pin, VALOR); // escribe un pseudo valor analógico de 0 a 255 en pines PWM

break; // fuerza el quiebre o salida de una función; normalmente se utiliza dentro de 'switch'

Byte // introduce variables de tipo 'byte', es decir, que trabajan con números enteros cortos de 8 bits. Su rango es de 0 a 255.

case // sirve para enumerar los casos posibles dentro de 'switch'

constrain (x, a, b); // fuerza a 'x' a permanecer entre los límites de 'a' y 'b'

default: // establece un caso por defecto dentro de 'switch' para que la programación siga este camino en caso de no coincidir con las condiciones de ninguno de los casos detallados.

delay (ms); // define un tiempo de demora o retardo. 1000 ms = 1 seg

digitalRead (pin); // lee el estado del 'pin'

digitalWrite (pin, LOW ó HIGH); // escribe un estado 'LOW' o 'HIGH' en 'pin'

FALSE // valor constante, significa 'falso'

Float // introduce variables de tipo 'flotante', es decir, que trabajan con números decimales.

for (inicialización; condición; expresión){...} // bucle que se repite determinada cantidad de veces

HIGH // valor constante, significa 'alto', equivalente a '255'

if (CONDICIÓN){...} // ejecuta las siguientes instrucciones si la condición es 'TRUE'

if (CONDICIÓN){...} else {...} // si la condición es 'TRUE' ejecuta sus instrucciones. Sino ('else') le indica que haga otra cosa

#include <...> // sirve para incluir librerías dentro de nuestros programas

INPUT // valor constante, significa 'entrada'

Int // introduce variables de tipo 'entero', es decir, que trabajan con números enteros 16 bits. Su rango es de 32767 a -32768

Long // introduce variables de tipo 'largo', es decir, que trabajan con números enteros de 32 bits. Su rango es de -2147483648 hasta 2147483647

```
LOW                                  // valor constante, significa 'bajo', equivalente a '0'

map (valor, a1, a2, b1, b2);         // a un 'valor' dentro de los rangos 'a1' y 'a2' le asigna su
                                     equivalente dentro del rango comprendido entre 'b1' y 'b2'

max (x,y);                           // al comparar entre dos valores, el programa se queda con el de
                                     mayor valor

min (x,y);                           // al comparar entre dos valores, el programa se queda con el de
                                     menor valor

OUTPUT                               // valor constante, significa 'salida'

pinMode (pin, INPUT);                // configura el 'pin' como entrada

pinMode (pin, OUTPUT);               // configura el 'pin' como salida

pulseIn (pin, VALOR);                //  lee un pulso ('HIGH o 'LOW') que ingresa por un determinado
                                     pin y entrega como resultado el tiempo en microsegundos que
                                     duró dicho pulso

Serial.begin (9600);                 // abre el puerto serie y fija la velocidad de transmisión de datos

Serial.print (dato, tipo de dato);           // imprime un valor en el puerto serie

Serial.println (dato, tipo de dato);         // imprime un valor en el puerto serie e incluye un salto
                                             de línea para facilitar la lectura.

Serial.Read (dato, tipo de dato);            // lee la información que se encuentra en el puerto serie

switch (variable) { case...          // controla el flujo del programa permitiéndonos especificar
                                     diferentes instrucciones ('case') que serán ejecutadas en base a
                                     una condición

TRUE                                 // valor constante, significa 'verdadero'

void                                 // sirve para ejecutar una rutina, no devuelve valores

void loop                            // sirve para contener el programa de nuestro robot, que se
                                     reproducirá cíclicamente

void setup                           // sirve para recoger la configuración de nuestro programa

while (CONDICIÓN){...}               // bucle que se repite mientras dure la 'condición'
```

Palabras reservadas del IDE de Arduino

Estas palabras claves son constantes, variables y funciones que se definen en el lenguaje de programación de Arduino. Las mismas están reservadas por el programa y no se deben usar en nombres de variables para evitar un mal funcionamiento.

Constantes

HIGH	SERIAL	TWO_PI	FALLING	null
LOW	DISPLAY	LSBFIRST	RISING	
INPUT	PI	MSBFIRST	false	
OUTPUT	HALF_PI	CHANGE	true	

Variables de designación de puertos y constantes

DDRB	PB3	PINC	protected	switch
PINB	PB4	PORTC	public	throw
PORTB	PB5	PC0	return	try
PB0	PB6	PC1	short	unsigned
PB1	PB7	PC2	signed	void
PB2	DDRC	private	static	

Tipos de datos

boolean	long	exp	HALF_PI	log
byte	delayMicroseconds	true	if	&&
char	micros	false	++	!
class	/	float	!=	||
default	/**	floor	int	
do	.	for	<<	
double	else	<	>	
int	==	<=	>=	

Otras

abs	cos	sin	pinMode	random
acos	{}	sq	analogRead	PC3
+=	--	sqrt	analogWrite	PC4
+	default	-=	attachInterrupts	PC5
[]	delay	switch	detachInterrupts	PC6
asin	loop	tan	tone	PC7
=	max	this	noTone	DDRD
atan	millis	true	pulseIn	PIND
atan2	min	TWO_PI	shiftOut	PORTD
&	-	void	map	PD0
|	%	while	pow	PD1
boolean	/*	Serial	max	PD2
byte	*	begin	min	PD3
case	new	read	sqrt	PD4
ceil	null	print	lowByte	PD5
char	()	write	highByte	PD6
char	PI	peek	bitRead	PD7
class	return	flush	bitWrite	
,	>>	println	bitSet	
//	;	available	bitClear	
?:	Serial	digitalWrite	bit	
constrain	Setup	digitalRead	randomSeed	

MATERIAL COMPLEMENTARIO

¿Problemas con el IDE de Arduino?

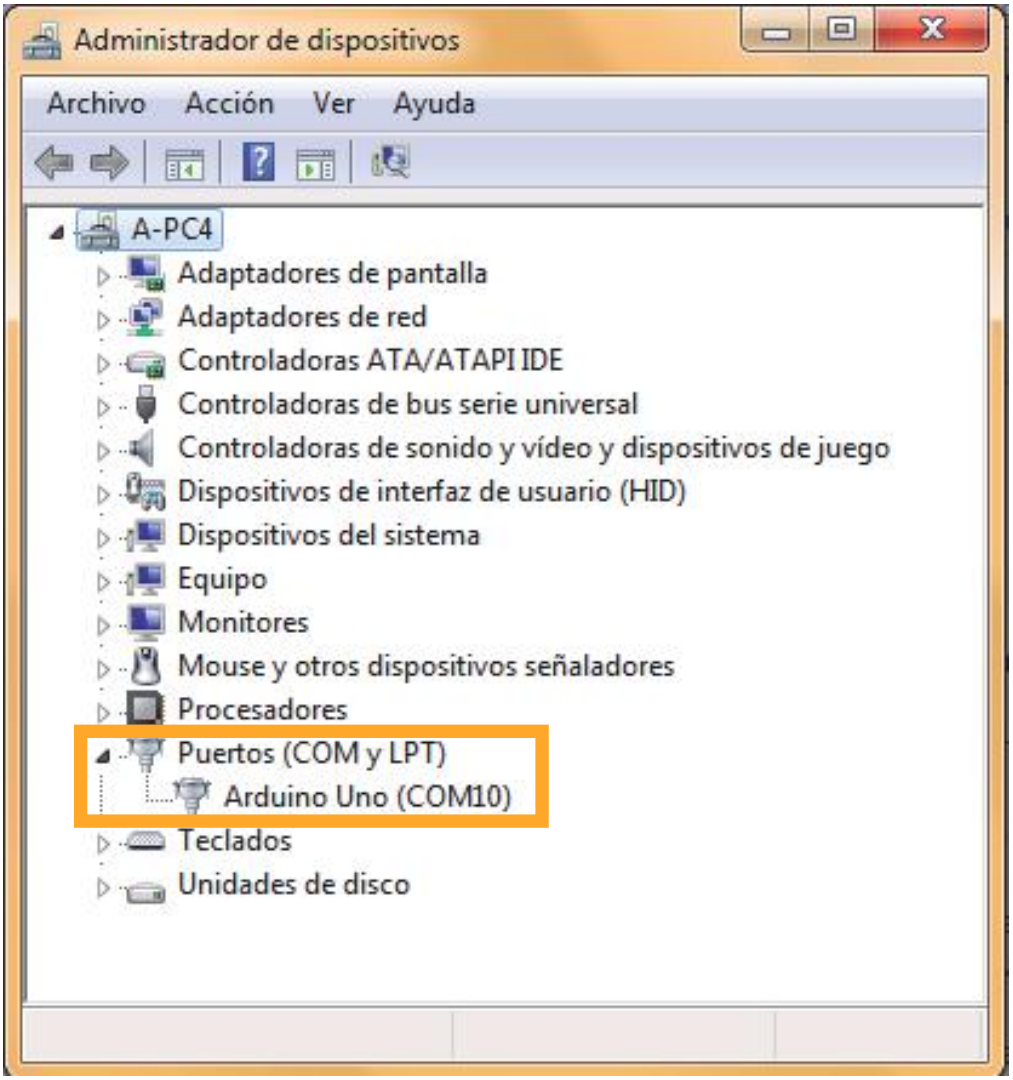
Si no podés bajar el sketch de prueba a la placa, debés seguir el siguiente procedimiento para ir descartando posibles problemas que puedas tener.

1. Verificar al conectar la placa por medio del USB que encienda el led "ON" de color verde, y luego parpadee el led 13, de color naranja. Si esto ocurre, nuestra placa está funcionando correctamente, si no, tenemos algún problema con la placa.

2. Controlar que esté seleccionada la placa correcta en el menú **Herramientas> Placa**. Debe decir **"Arduino/Genuino Uno"**.

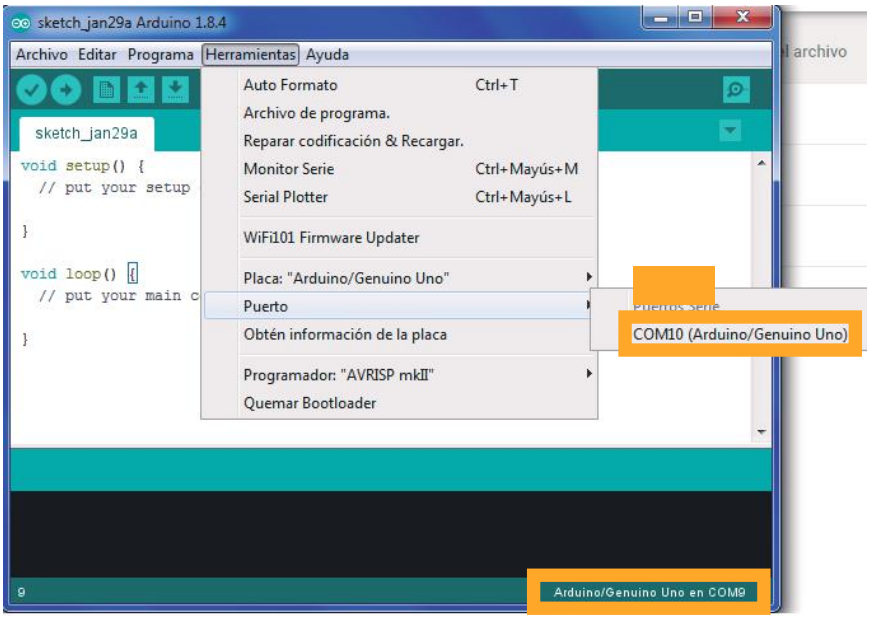
3. Luego, verificar que esté seleccionado el puerto correcto en el menú **Herramientas> Puerto** (si el puerto no aparece, o las opción está en gris, intentá reiniciar el IDE con la placa conectada a la computadora). Hay que verificar en el **Administrador de dispositivos** cuál es el correcto.

   Para abrir el administrador podemos ir al panel de control, en **"Hardware y sonido"**, o bien buscarlo haciendo clic en el botón de inicio y usando **"Buscar programas y archivos"**. Si el driver se instaló correctamente veremos en la sección **"Puertos (COM y LPT)"** el puerto que tenemos que seleccionar, en este caso, **COM10**.

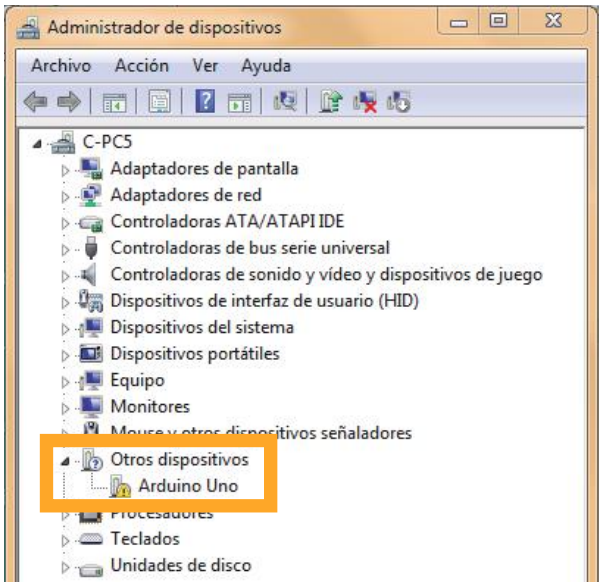


Mala comunicación entre puertos

En la siguiente imagen vemos cómo el IDE está configurado para enviar la información al **COM9** (abajo a la derecha) cuando Windows le está asignando el puerto **COM10**, por lo que tendremos un error al intentar bajar el sketch a la placa. Solucionaremos este problema de comunicación seleccionando el **COM10** desde **Herramientas > Puerto > COM10.**



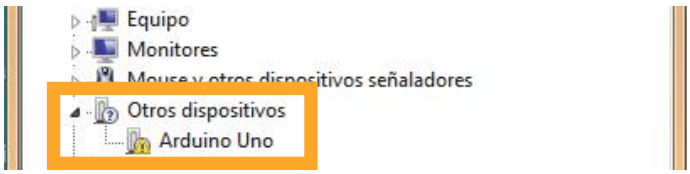
Si en el administrador de dispositivos no encontrás a tu arduino dentro de los puertos **COM**, buscalo en **"Otros dispositivos"** como **"Dispositivo desconocido"**. Si tenés más de un dispositivo desconocido en tu pc, desconecta la placa y vuelve a conectarla. El que desaparezca y vuelva a aparecer será tu arduino. Si todavía no encontrás a tu arduino dentro de **"Otros dispositivos"** puede ser que haya un problema con el cable USB. Buscá uno de impresora (que tienen los mismos conectores) y probá conectar nuevamente la placa.



Una vez identificado el dispositivo, tenemos que actualizar el *driver.* Cómo hacerlo depende de la versión de Windows que tengas instalada en tu computadora.

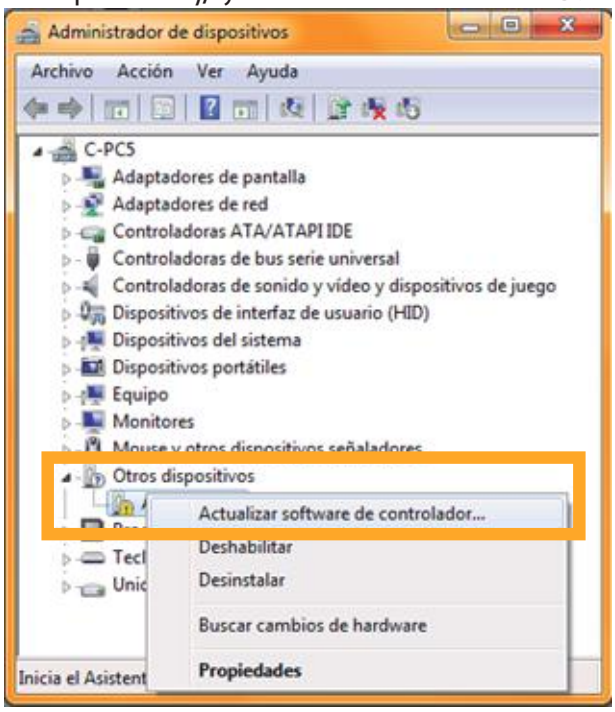
Problemas con el driver

Si tu PC no reconoce el driver o precisa actualizarlo, cuando vayas al **Administrador de dispositivos** (recuerda que puedes buscarlo a través del panel de control, en ***"Hardware y sonido"***, o a través de ***"Buscar programas y archivos"***) te aparecerá algo similar a esto:

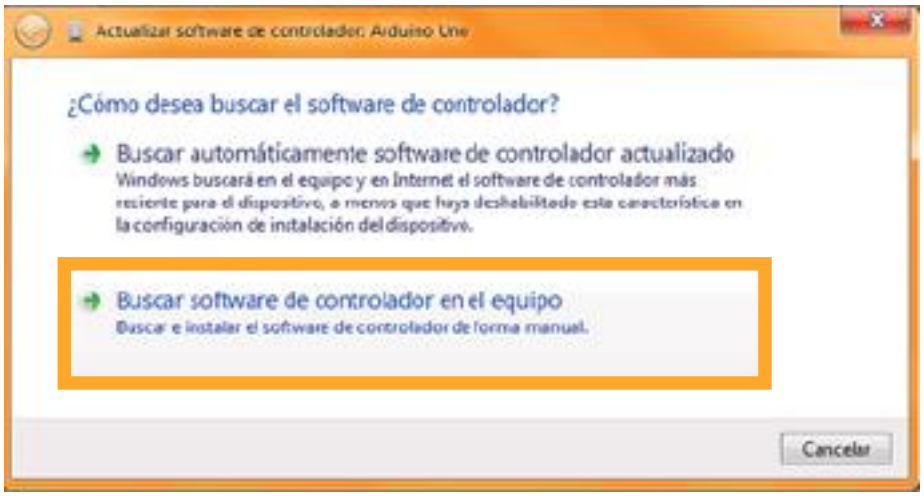


WINDOWS 7

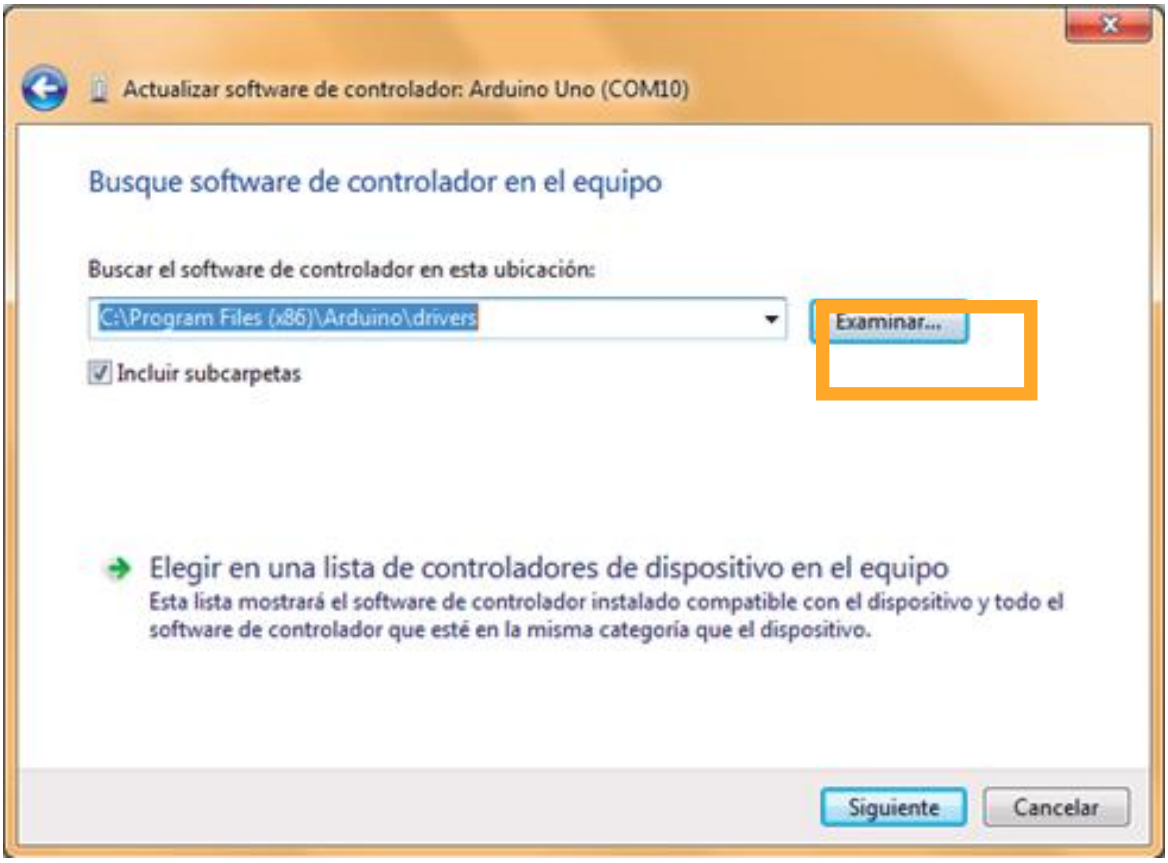
En Windows 7 tenés que hacer clic con el botón derecho en el dispositivo (la placa debe estar conectada a la computadora), y seleccionar **"Actualizar software del controlador"**.



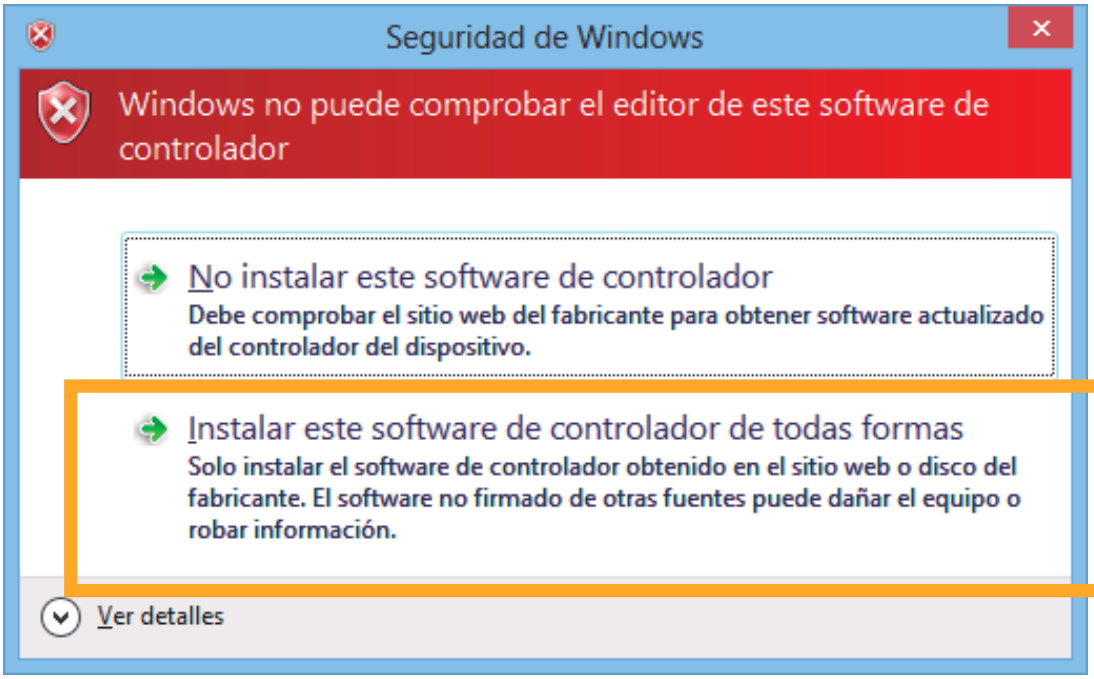
Después selecciona: **"Buscar software del controlador en el equipo"**.



Luego presiona **"Examinar"** y buscar la carpeta donde instalamos el IDE, en nuestro caso sería la siguiente:

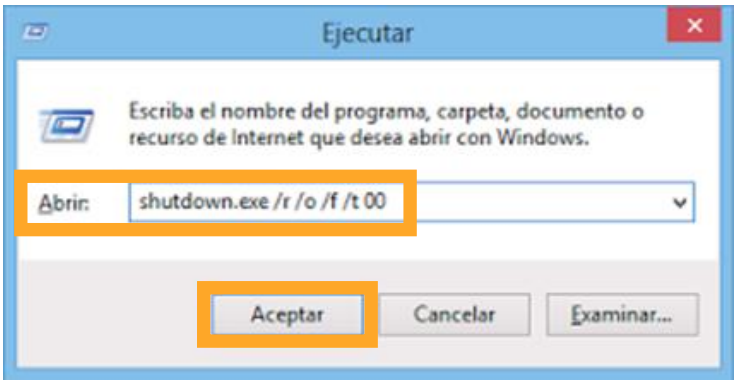


Para finalizar, hacemos clic en **"aceptar"** y después en **"siguiente"**. Si aparece algún mensaje advirtiendo sobre un posible problema de seguridad o compatibilidad, seleccionar **"Instalar de todas formas"**.



WINDOWS 8

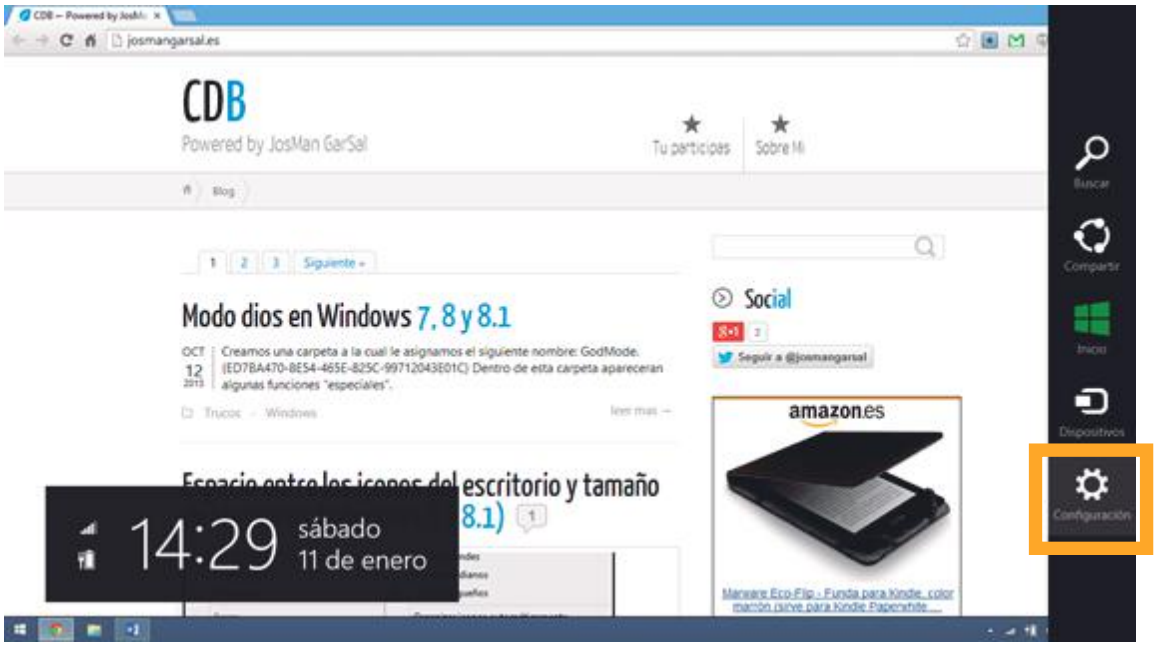
En Windows 8, antes de poder seguir los pasos especificados para Windows 7, debemos desactivar el uso obligatorio de drivers con firma. Esto se logra modificando una opción de Windows y reiniciando la PC siguiendo un procedimiento especial. Para esto debes abrir el cuadro de diálogo **"Ejecutar"** (puedes encontrarlo desde el escritorio, presionando la tecla de Windows + R) y escribir: **"shutdown.exe /r /o /f /t 00"**. Luego haz clic en **"Aceptar"**.



A continuación se reiniciará tu computadora y verás el menú **"Solucionar problemas"**. Si llegaste hasta ese punto, continúa leyendo a partir del **paso 5** del siguiente instructivo. Si no, intentá los pasos que siguen.

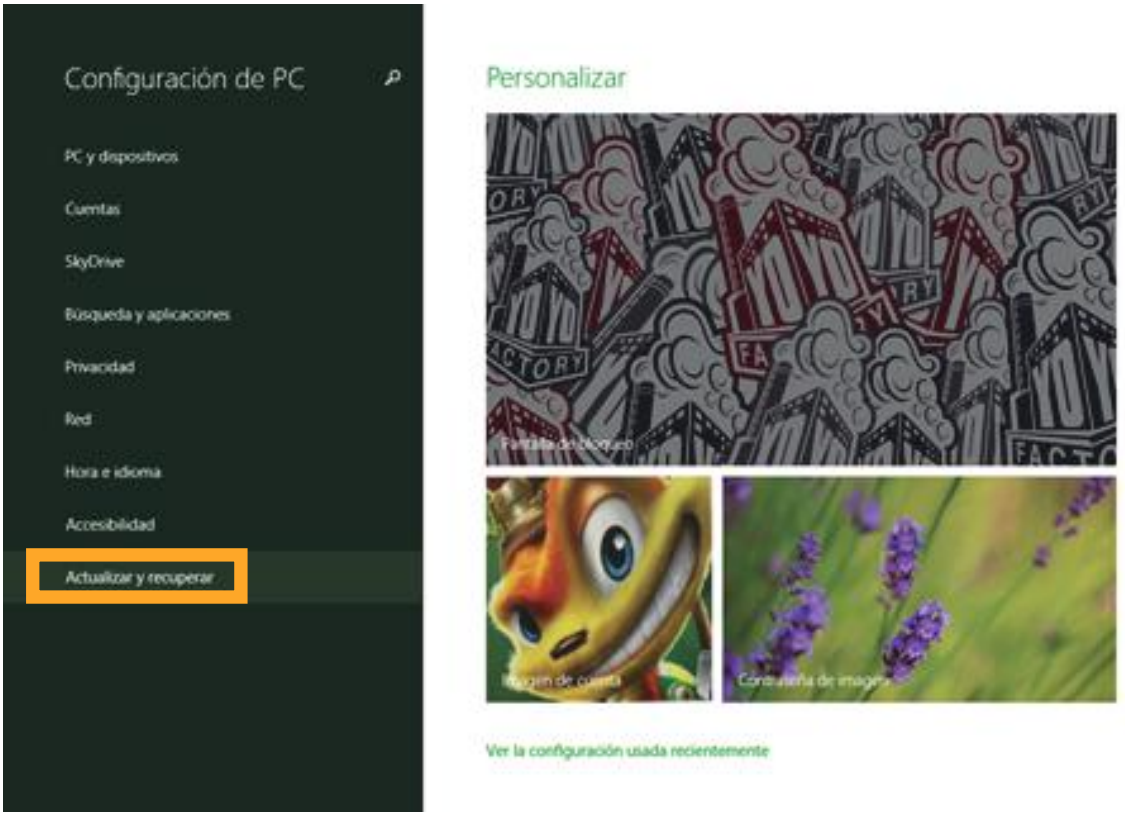
NOTA: Si tenés Windows 8 o 10, probá descargar la aplicación de arduino desde el STORE, en caso contrario, seguí con esta guía.

1. Desplegamos el menú que aparece al acercar el cursor a una de las esquinas derechas de la pantalla y pulsamos en **"Configuración"**.

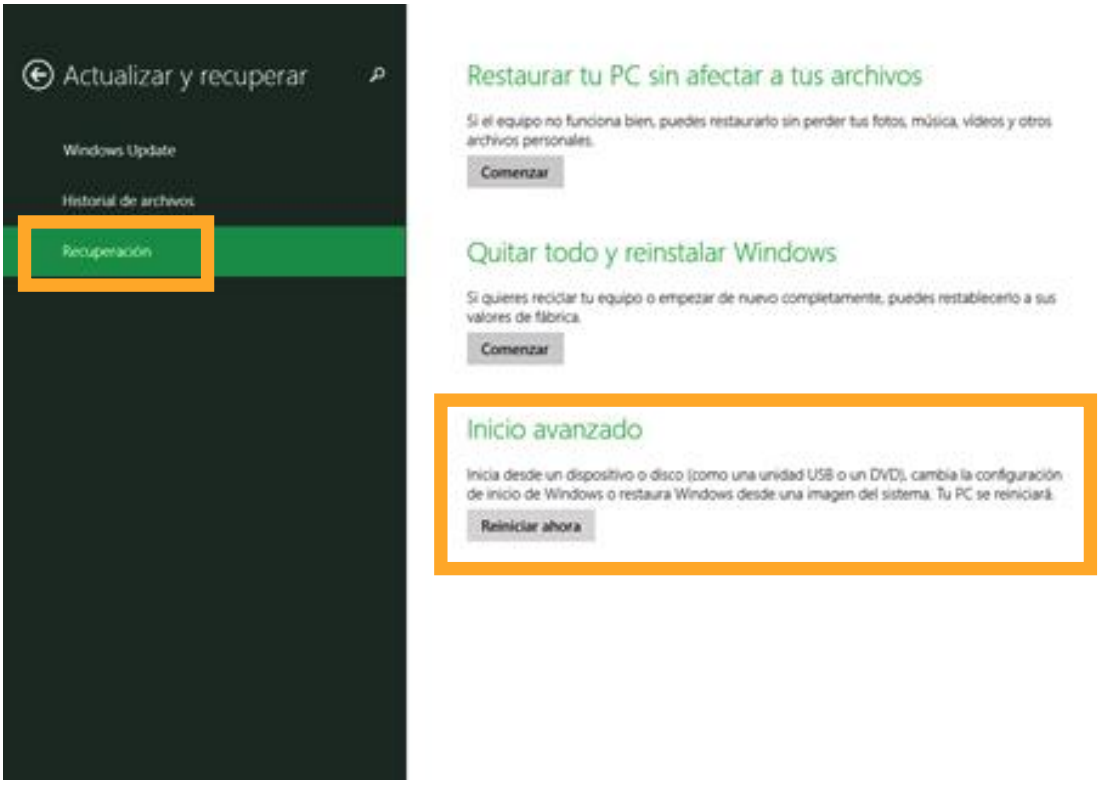


2. Aparecerá una barra con varias opciones, pulsamos sobre "**Cambiar configuración de PC**" en la parte inferior.

3. Se abrirá una aplicación para configurar la PC. En el menú de la izquierda busca la opción **"Actualizar y recuperar"**.



4. Ahora vamos a **"Recuperación"** en el menú de la izquierda y en el apartado **"Inicio avanzado"** seleccionamos **"Reiniciar ahora"**. El sistema se reiniciará y entrará en el menú de inicio avanzado.



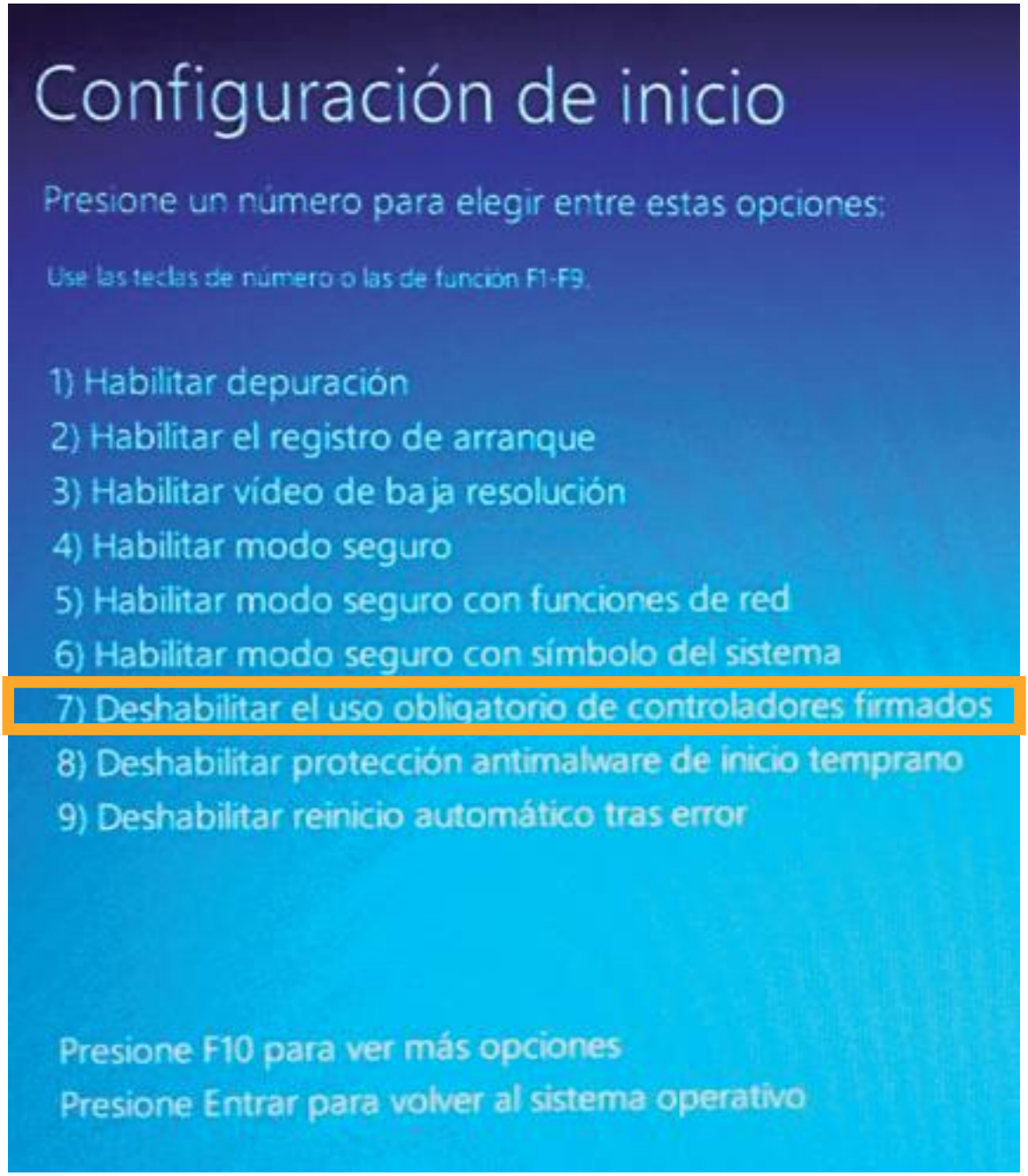
5. Una vez que la Windows se reinició, aparecerá este menú. Seleccionamos **"Solucionar problemas"**.

6. En la siguiente pantalla elegimos **"Opciones avanzadas"**.

7. Finalmente, en la pantalla de Opciones avanzadas, elegimos **"Configuración de inicio"**.

8. Nos aparecerá una nueva pantalla indicando las características que podremos modificar. Haz clic en **"Reiniciar"** para acceder al menú de configuración.

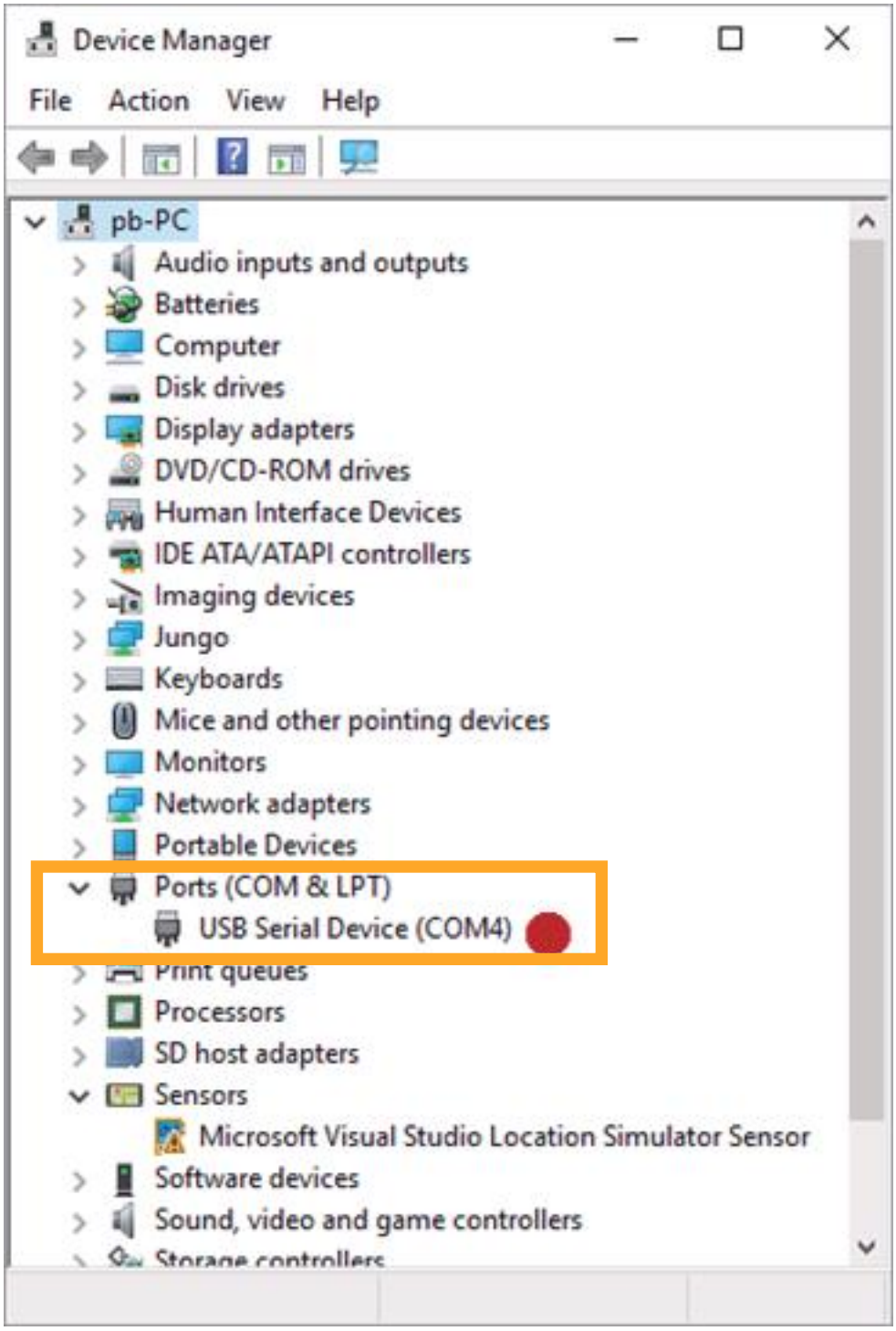
9. Para finalizar, pulsamos la tecla **7** que es la opción que nos interesa. El sistema se reiniciará y ya podremos instalar drivers no firmados.



Durante la instalación de los drivers el sistema nos pedirá confirmación para instalar drivers no firmados, simplemente se lo permitimos y listo. A partir de acá, seguí los pasos indicados previamente para Windows 7 para poder actualizar el controlador de la placa arduino.

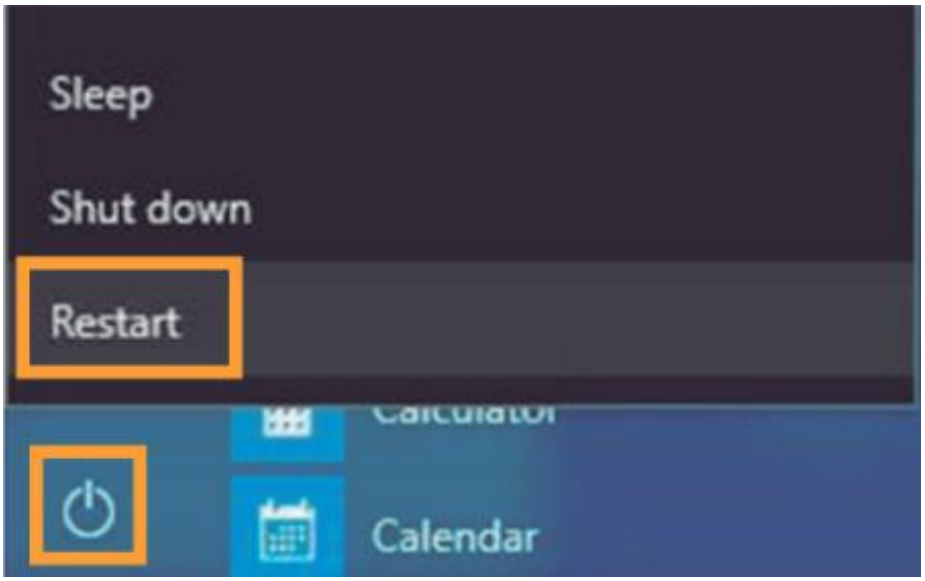
WINDOWS 10

En el caso de Windows 10 debes hacer clic derecho en el botón **"Inicio"** y seleccionar el **"Administrador de dispositivos"** en el menú. Una vez en el administrador, expandir **"Puertos (COM & LPT)"**. Vas a ver un puerto **COM** que será el configurado para tu arduino, como se muestra en la siguiente imagen.



Si tenés varios puertos COM, podés desconectar el arduino y conectarlo de nuevo para ver cuál es el puerto que aparece y de esa forma saber cuál es el asignado a tu arduino (COM4 en el ejemplo de abajo). El driver instalado es uno que instala windows por defecto, pero podemos instalar el que trae Arduino, haciendo clic en el puerto que identificamos y siguiendo detallados en este instructivo en la sección de Windows 7.

Si no podés instalarlo porque tu pc no permite instalar drivers no firmados hay que modificar la configuración de Windows para que nos permita hacerlo. Para esto, debemos acceder al **menú de reinicio**: debes hacer clic en el menú inicio, y en el botón **"Power"** y luego, manteniendo apretada la tecla **"Shift"**, apretar **"Reiniciar"**.



Una vez que la PC se reinicia, presioná 7 o F7, que corresponde a la opción **"Deshabilitar el uso obligatorio de drivers firmados"**. Hecho esto, tu computadora ya te debería permitir actualizar el controlador de la placa arduino sin inconvenientes.

ACTIVIDADES Y EJERCITACIÓN

Cap. 2: Actividades complementarias

En base a los contenidos vistos en el **"Capítulo 2: ¡Hola Mundo!"** realiza las siguientes actividades complementarias.

Actividad 1

Responde las siguientes preguntas.

a. ¿A qué llamamos Arduino?

b. ¿Cuáles son los componentes del sistema Arduino?

c. ¿Cuáles son los componentes de la placa Arduino (microcontrolador, pines, alimentación, leds,etc.)?

Actividad 2

Trata de interpretar el código de **"Blink"** leyendo lo que dice. Si es necesario puedes repasar el **"Glosario de programación Arduino"** para ayudarte. Una vez que lo hayas comprendido, ponte a prueba e intenta modificar la programación para que el parpadeo sea más rápido.

```
void setup() {  // La función setup corre sólo una vez cuando presionamos el reset o cuando alimentamos la placa.
    pinMode(LED_BUILTIN, OUTPUT);  // Se inicializa el led incorporado en la placa (led 13) como led de salida.
}

void loop() {    // La función loop se ejecuta después del setup y, como su nombre lo indica, se repetirá una y otra vez
                    mientras esté alimentada la placa.
    digitalWrite(LED_BUILTIN, HIGH);  // Enciende el led levantando el voltaje.
    delay(1000);        // Espera un segundo.
    digitalWrite(LED_BUILTIN, LOW);   // Apaga el led bajando el voltaje.
    delay(1000);        // Espera un segundo.
}
```

Cap. 2: Respuestas

A continuación encontrarás las respuestas a las actividades para que puedas evaluar tu propio desempeño.

Actividad 1

Responde las siguientes preguntas.

a. ¿A qué llamamos Arduino?
Arduino es una plataforma de electrónica. También se llama así al microcontrolador que nos permite trabajar con actuadores y sensores por medio de una programación.

b. ¿Cuáles son los componentes del sistema Arduino?
En nuestro sistema Arduino los sensores nos permiten ingresar información del entorno (**componentes de entrada**), la placa procesa esta información de acuerdo al programa que hayamos escrito, y envía órdenes o instrucciones a los actuadores para que estos den respuestas al medio (**componentes de salida**).

c. ¿Cuáles son los componentes de la placa Arduino (microcontrolador, pines, alimentación, leds,etc.)?
Los componentes de la placa Arduino UNO son:

- Entrada USB
- Alimentación 9V
- Pines digitales
- Pines PWM (~).
- Pines analógicos
- Pines de encendido
- Pines GND o "tierra"
- PIN 13 - LED integrado
- Botón reset
- LEDs RX / TX
- Microcontrolador Atmega 328

Actividad 2

Para que el parpadeo de **"Blink"** sea más rápido lo que se debe cambiar es la cantidad de delay especificada dentro del loop.

Por ejemplo:

```
delay(500);    // Esperará medio segundo.
```