TP: Introducción a Arduino

En este trabajo practico se describirá el entorno Arduino, se dará una breve introducción al uso de entradas salidas digitales y se explicará paso a paso como cargar el código generado en una placa real, así como simular la misma.

Que es Arduino?

Arduino es una plataforma para prototipado rápido de electrónica, se caracteriza por llevar al hardware la idea de código abierto (open-source). Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Las placas Arduino se basan en la utilización de un microcontrolador el cual puede ser programado mediante lenguaje Arduino (un lenguaje muy similar al Lenguaje C). Un microcontrolador, salvando las distancias, es una computadora de recursos limitados este contara con un microprocesador, memoria y periféricos.

Al igual que en la programación tradicional nos interesara realizar acciones que resulten en condiciones de las entradas a los periféricos provenientes del mudo exterior. En el caso de los microcontroladores estos periféricos serán sus pines, pudiendo detectar señales digitales o analógicas, las cuales pueden provenir de pulsadores, sensores de temperatura, tensión, distancia etc. De la misma manera nos interesara ejercer alguna acción en el

exterior que en este caso la haremos también a través de los pines del microcontrolador, activando o desactivando motores, leds, etc.

Como dijimos las placas Arduino llevan el concepto de código abierto al hardware, esto significa que tanto el software así como los diseños del hardware (archivos CAD) están disponibles bajo licencia open-source, por lo que es posible descargarlos gratuitamente y adaptarlas a nuestras necesidades.

Por qué Arduino?

Aunque hay muchos otros microcontroladores y plataformas microcontroladoras disponibles para computación física, Arduino en particular toma los desordenados detalles de la programación de un microcontrolador y los encierran en un paquete fácil de usar. Además de simplificar el proceso de trabajo con microcontroladores, le podemos adjuntar las siguientes características:

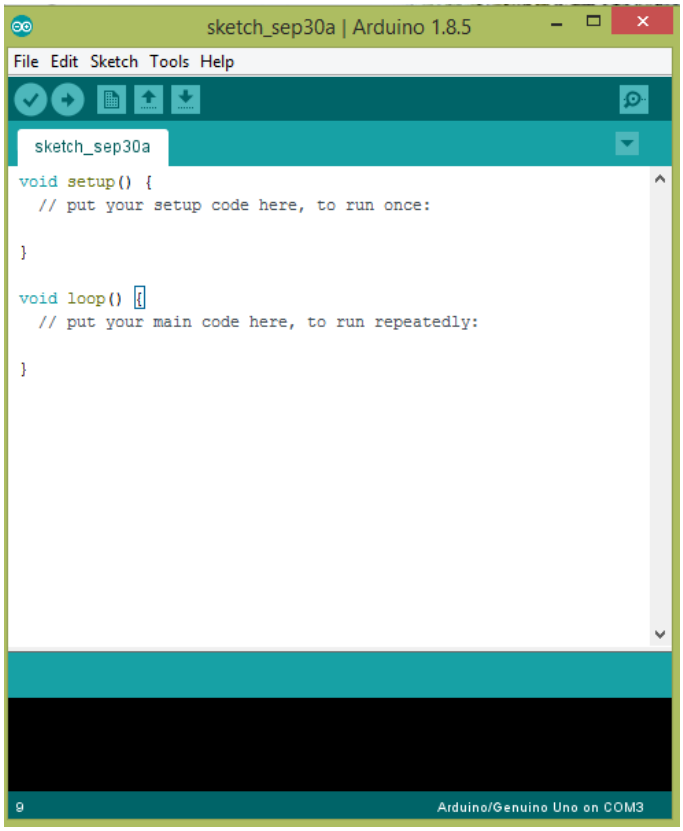
- **Barato:** Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras.
- **Multiplataforma:** El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux.
- **Entorno de programación simple y claro:** El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también.
- **Código abierto y software extensible:** El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerias C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado.
- **Código abierto y hardware extensible:** El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo.

Como comenzar?

Lo primero que debemos hacer es descargar el entorno de desarrollo de Arduino (IDE), en el cual realizaremos nuestros programas, este los podemos descargar desde:

https://www.arduino.cc/

Allí, yendo a la sección de ***programas -> descargas*** podremos obtener la versión del IDE correspondiente a nuestro sistema operativo. Una vez descargada e instalada al ejecutarlo tendremos lo siguiente en pantalla:



Aquí podemos ver la pantalla principal del IDE, donde desarrollaremos nuestros programas. Por defecto vemos la estructura básica del lenguaje de programación de Arduino que se compone de al menos dos partes. Estas dos partes necesarias (funciones), encierran bloques que contienen declaraciones, estamentos o instrucciones.


```
void setup()
{
//estamentos;
}
void loop()
{
//estamentos;
}
```

La función **setup()**, es la parte encargada de realizar la configuración. Es la primera función a ejecutarse en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar pinMode (modo de trabajo de las E/S), configuración de la comunicación en serie, declaración de variables y otras.
La función **loop()** contiene el código que se ejecutara continuamente (lectura de entradas, activación de salidas, etc). Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.
Luego de estas funciones estándar, podremos agregar las funciones que nosotros deseemos crear y o utilizar, a diferencia de C no necesitamos escribir un prototipo, sino que una vez desarrollado el código de la función podemos llamarle desde la función loop().

Primer programa:

El primer programa que realizaremos consistirá en encender y apagar un led. A continuación, se encuentra su código:

```
// Ejemplo: led que parpadea
void setup()
{
 pinMode(13, OUTPUT);
}
void loop()
{
 digitalWrite(13, HIGH);   // enciende el LED
 delay(1000);              // espera por un segundo
 digitalWrite(13, LOW);    // apaga el LED
 delay(1000);              // espera por un segundo
}
```

En el código vemos algunas funciones especifica del lenguaje Arduino:

pinMode(pin, OUTPUT);
Esta instrucción es utilizada en la parte de configuración setup() y sirve para configurar el modo de trabajo de un PIN pudiendo ser INPUT (entrada) u OUTPUT (salida). El pin se puede especificar ya sea como una variable o como una constante.

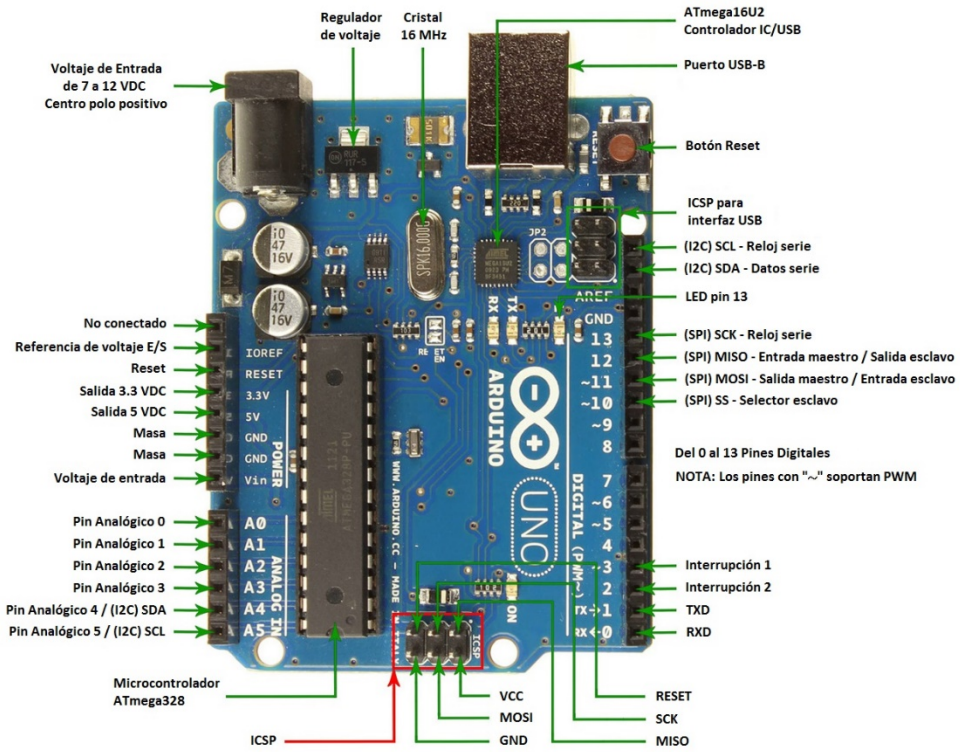
digitalWrite(pin, HIGH); o digitalWrite(pin, LOW);
Envía al "pin" definido previamente como OUTPUT el valor HIGH o LOW (poniendo en 1 (5V) o 0 (0V) la salida).

delay(x);
Esta instrucción frena la ejecución del programa durante "x" milisegundos.

Conociendo las funciones antes detalladas, podemos ver que el código del ejemplo, primero configura el pin 13 de la placa como salida dentro de la función setup(). Luego en la función loop() se pone en "1" el pin 13 mediante digitalwrite, esta situación se mantiene durante un segundo usando la instrucción delay, para luego poner en estado bajo el mismo pin y volver a esperar un segundo.

A diferencia de la programación tradicional, la programación de microcontroladores se piensa como un programa que se repetirá infinitamente, o a lo sumo hasta que se lleve al programa a una condición de alto.

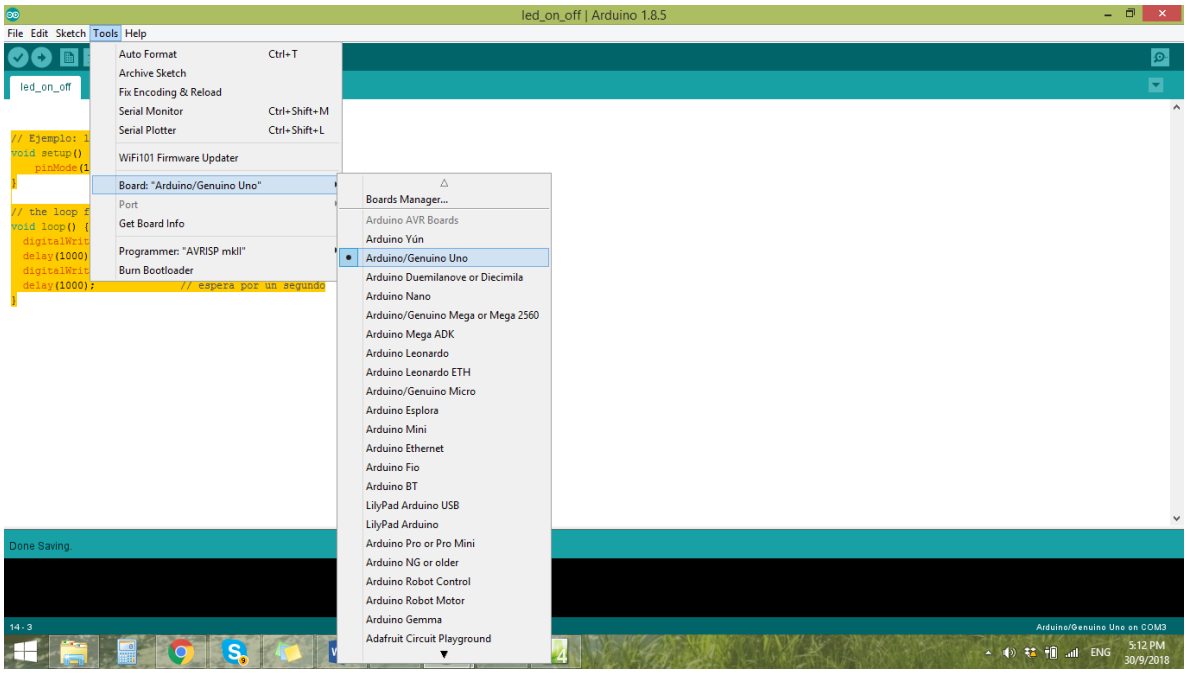
- **Los pines de las placas Arduino se encuentran numerados. Al usar pines digitales simplemente remplazamos por el numero de pin que deseamos usar (ver esquema de placa)**
- **Los pines de las placas Arduino en general se encuentran agrupados por su funcion entrada-salida digital, entradas analogicas, de potencia, etc. Es comun que un pin tenga mas de una funcion para esto es interesante ver el datasheet de la placa con la que estemos trabajando.**



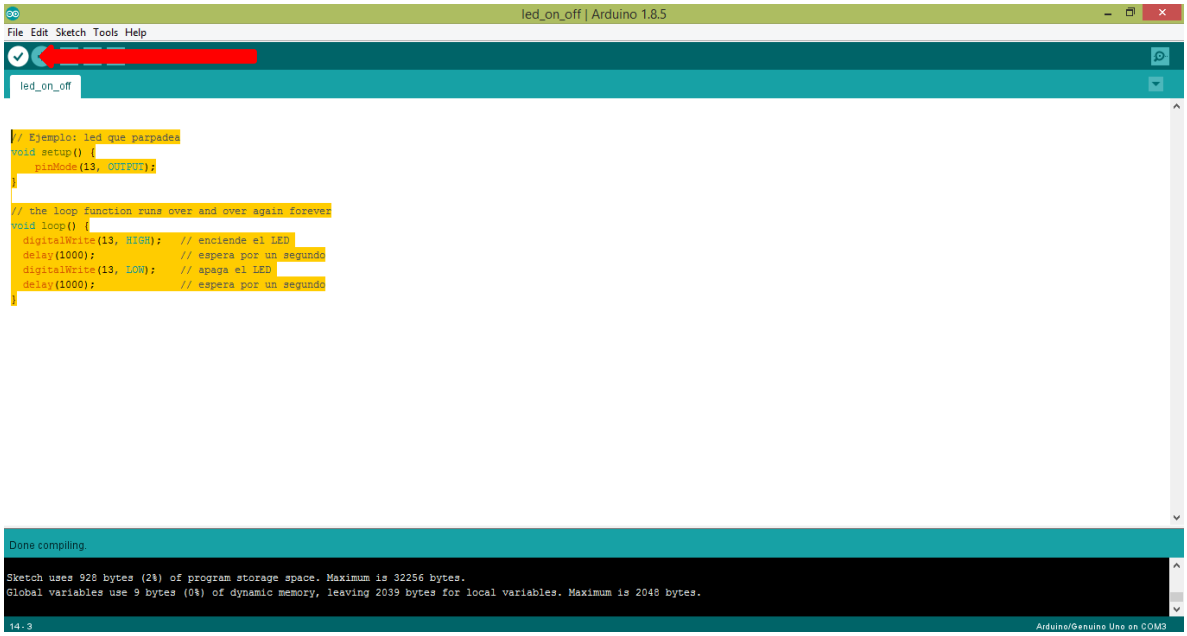
Una vez escrito nuestro programa debemos compilarlo para asegurarnos que no contenga errores y para generar archivos binarios necesarios para la carga del programa en el microcontrolador. Para esto debemos realizar los siguientes pasos:

1. Si no lo hicimos antes, guardamos nuestro programa en una carpeta de trabajo a elección. Es recomendable guardar en una carpeta conocida, y que no contenga otros programas a fin de evitar sobre escribir archivos. Para esto vamos al menú **File->Save as** y seleccionamos donde guardar nuestro programa.

2. Debemos seleccionar la placa Arduino con la que trabajaremos, para esto vamos al menú **Tools -> Boards ->** seleccionamos placa deseada. (En nuestro primer ejemplo utilizaremos la Arduino Uno).



3. Realizamos la compilación, cliqueando sobre el botón con un tilde en la parte superior izquierda, si todo sale bien nos deberían indicar un mensaje de que esta ha sido realizada correctamente.



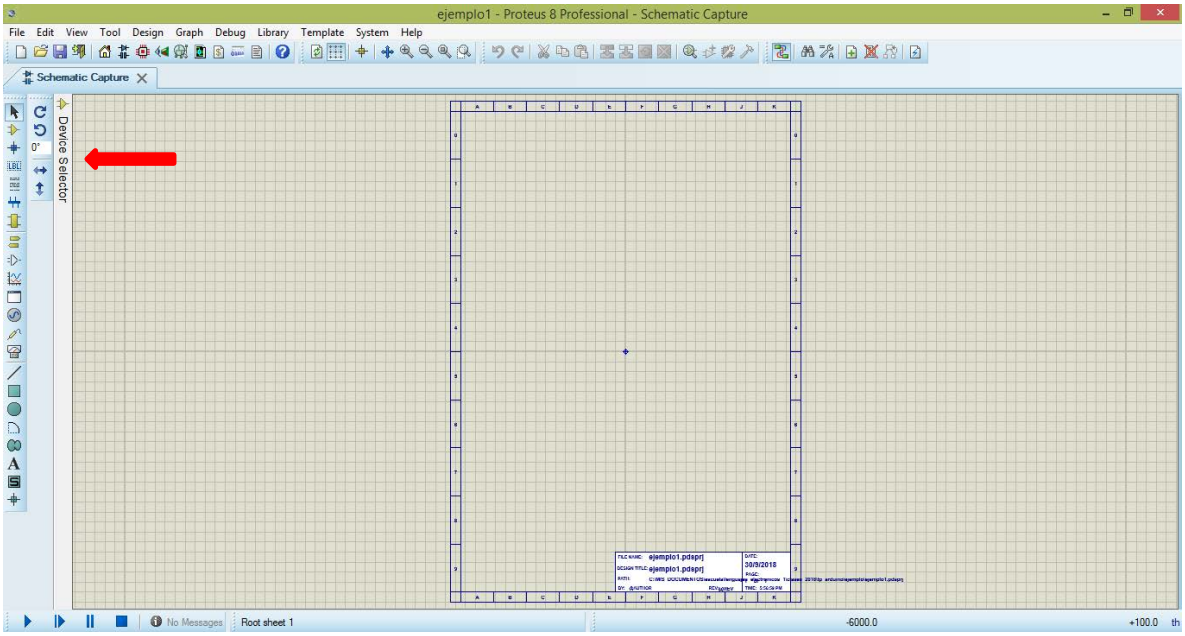
A continuación, siguen dos opciones, o bien simular nuestro programa o probarlos sobre el circuito real:

Simulación:

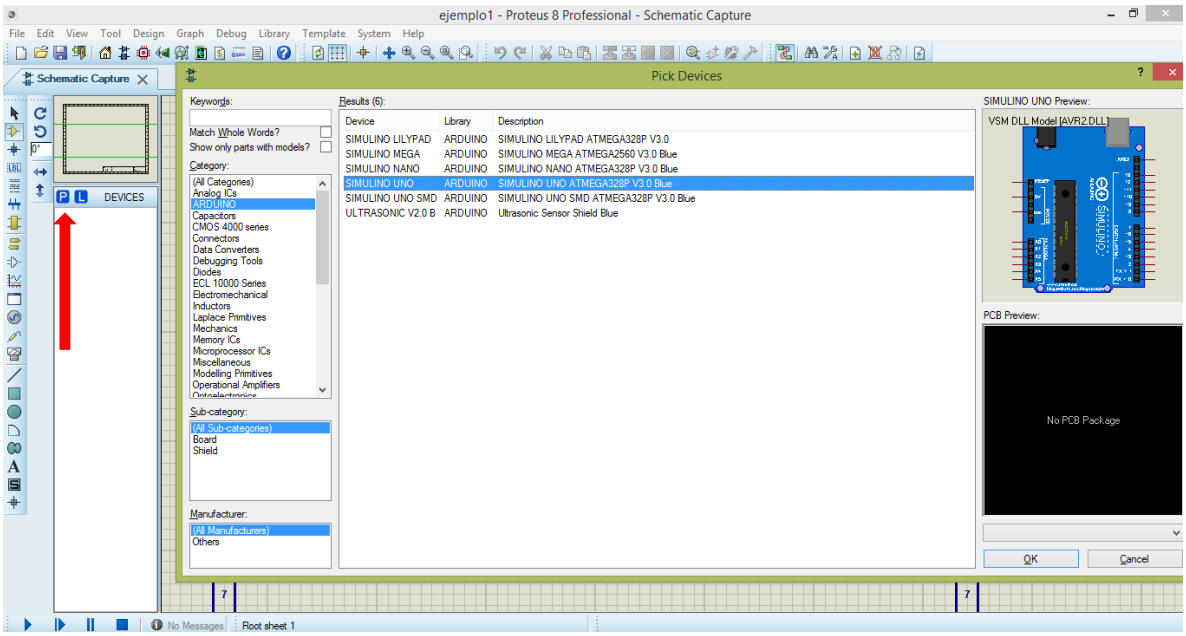
Para simular nuestro proyecto utilizaremos el programa Proteus, este es un clásico programa de simulación de circuitos que además cuenta con la opción de desarrollo de circuitos impresos.

Al abrir el programa seleccionamos **File -> New Project** allí se abrirá una ventana donde nos da la opción de nombrar nuestro proyecto, y seleccionar donde guardarlo, es recomendable guardarlo en una subcarpeta de la carpeta que ya creamos para nuestro programa Arduino.

Al realizar lo anterior y darle siguiente, nos pide seleccionar un "template" esto es el formato de la hoja de dibujo donde montaremos nuestro circuito, seleccionamos la más adecuada, por ejemplo, Portrait A4. Dándole siguiente a todas las ventanas que van apareciendo y finalizar en la última veremos en pantalla la hoja de trabajo:



Ahora deberemos montar nuestro circuito sobre esta hoja de trabajo, para esto seleccionando en la solapa **Device selector** (ver imagen superior), se abre un menú lateral donde al seleccionar el botón **P** (de place) abrirá una ventana con todas las librerías del programa:



Navegando por las librerías podremos seleccionar los componentes deseados, para este caso:

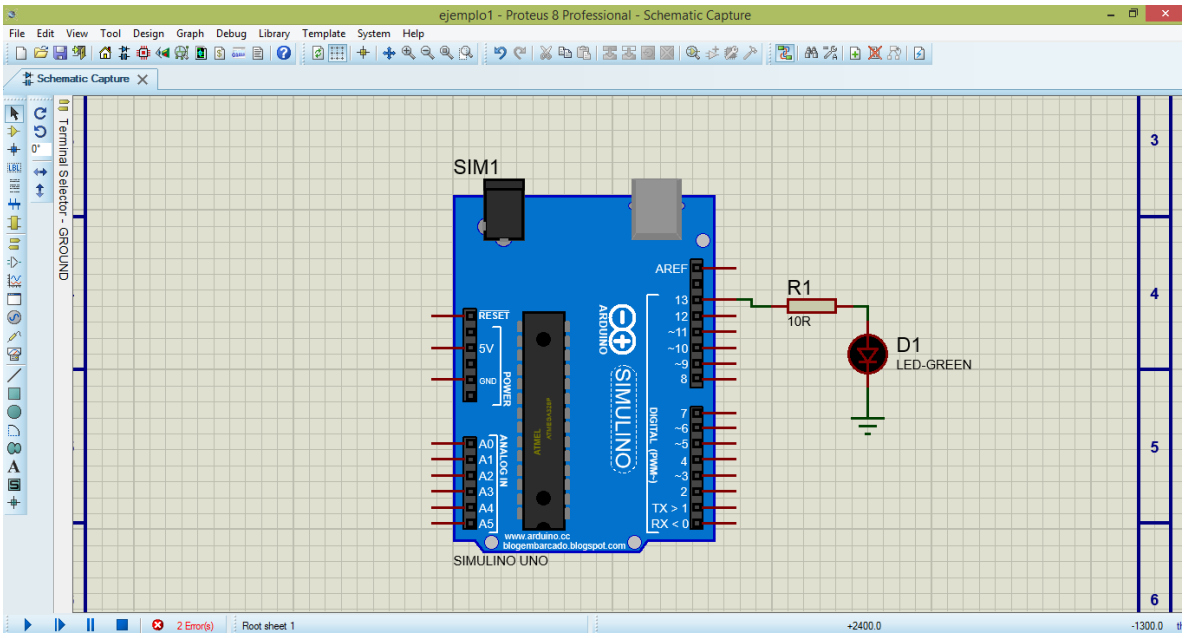
De la librería Arduino -> SIMULINO UNO

De la librería ACTIVE -> LED-GREEN

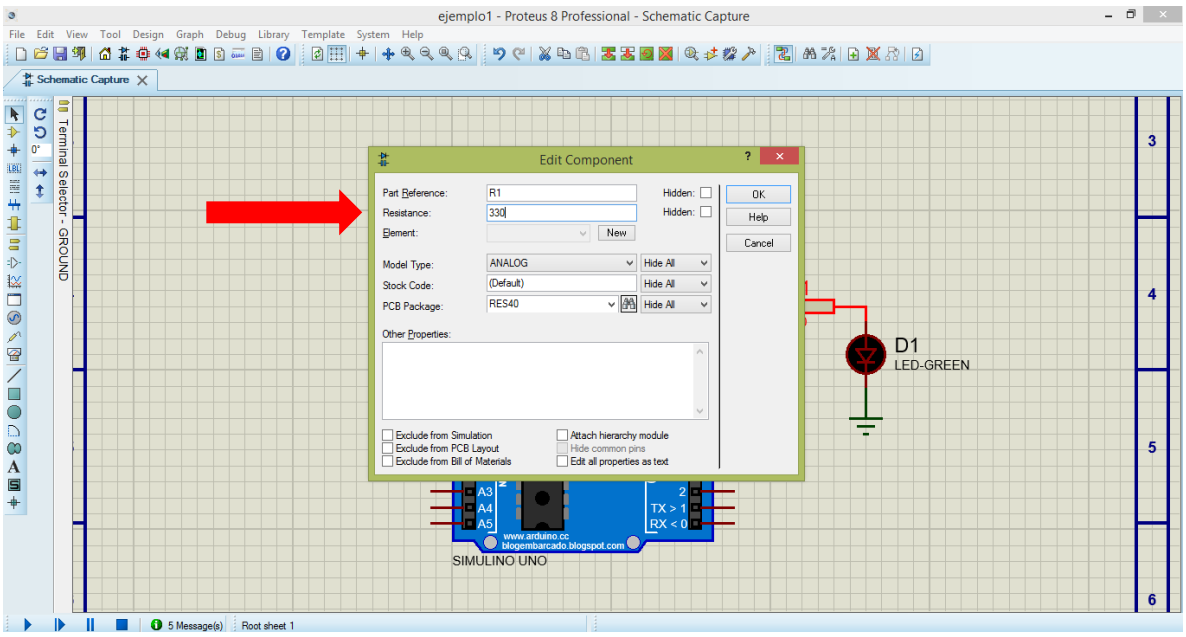
De la librería Resistors -> una resistencia cualquiera

Una vez seleccionados todos los componentes y colocados en la hoja de trabajo, con el botón derecho del mouse sobre sus terminales podemos conectarlos dibujando las líneas de conexión.

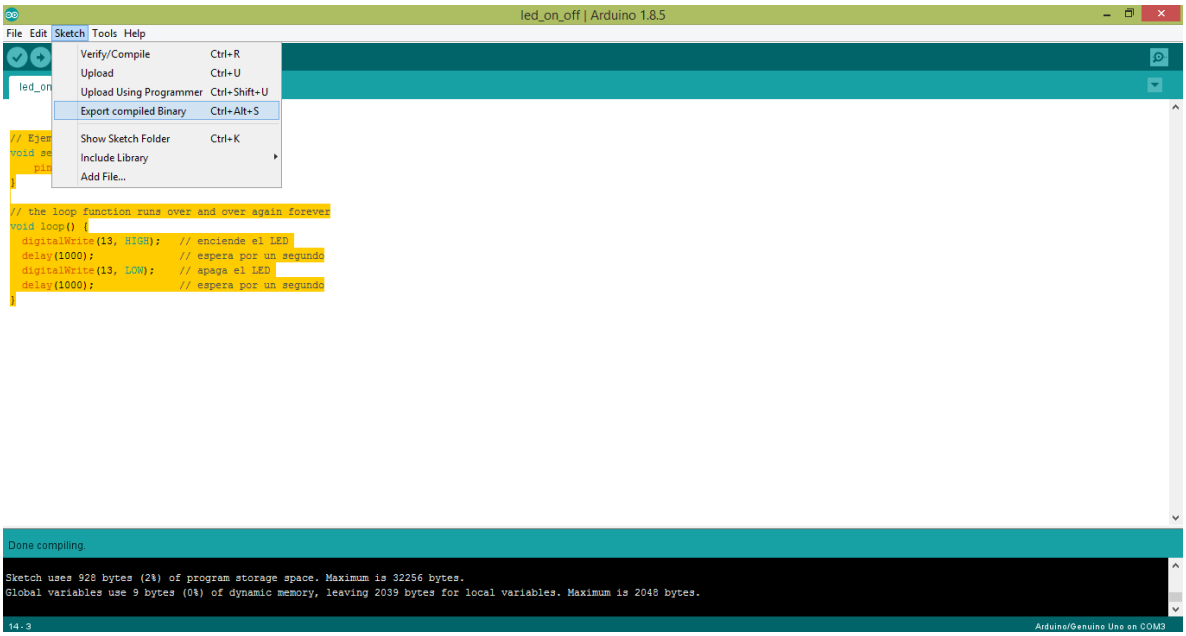
Luego deberemos colocar la masa del **circuito (en este simulador siempre deberemos colocar una masa para que funcione)**. Para esto buscamos el botón “terminal mode” en la barra de la derecha y seleccionamos GROUND, luego colocamos el componente en la hoja de trabajo. Deberíamos tener algo similar a:



Debemos setear el valor de la resistencia para esto haciendo doble clic sobre el componente, se abre una ventana de propiedades en la cual podemos modificar este valor, para este caso a 330Ω

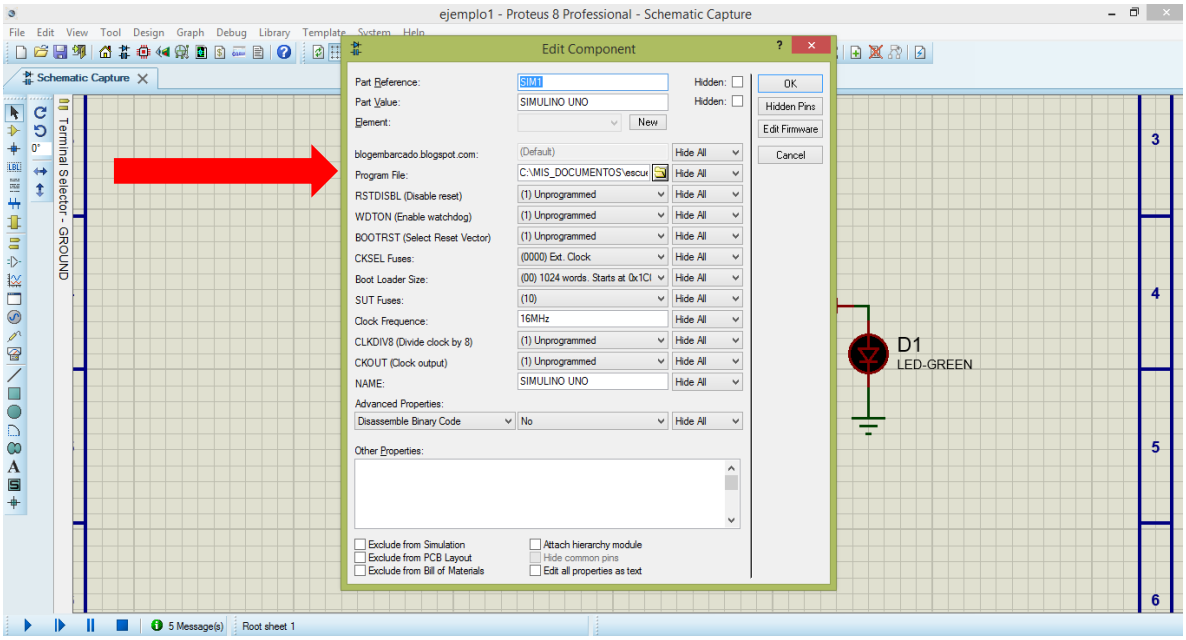


Finalizado el montaje de nuestro circuito debemos cargar el programa que realizamos en el entorno Arduino, para esto debemos volver al entorno Arduino y generar los archivos binarios necesarios para el simulador. Para hacer esto vamos al menú **Sketch -> Export compiled Binary**

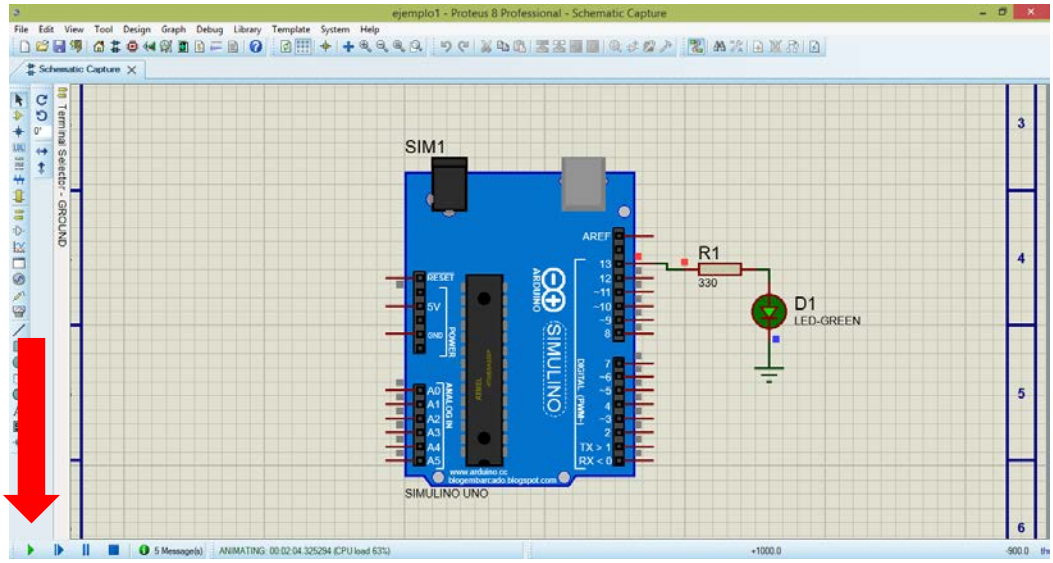


Esto generara los archivos necesarios para el simulador en la carpeta donde hemos guardado nuestro programa.

Volviendo a Proteus haciendo doble click sobre el Arduino Uno se abre la ventana de propiedades del mismo donde podremos seleccionar el archivo que contenga el programa a cargarle (como dijimos este se encontrara en la carpeta donde guardamos el programa Arduino, y será el archivo con la extensión .hex):



Para probar el programa solo resta darle click al botón de simulación en la parte baja de la pantalla, si todo se realizó correctamente el led debería parpadear:

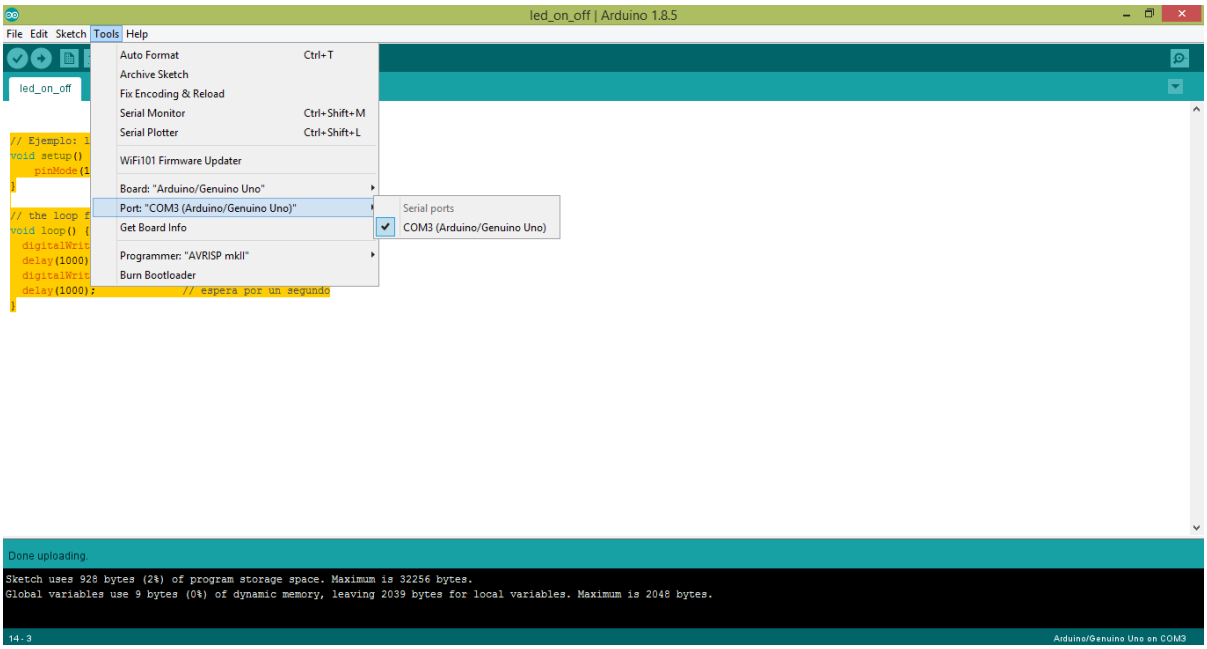


Prueba en circuito real:

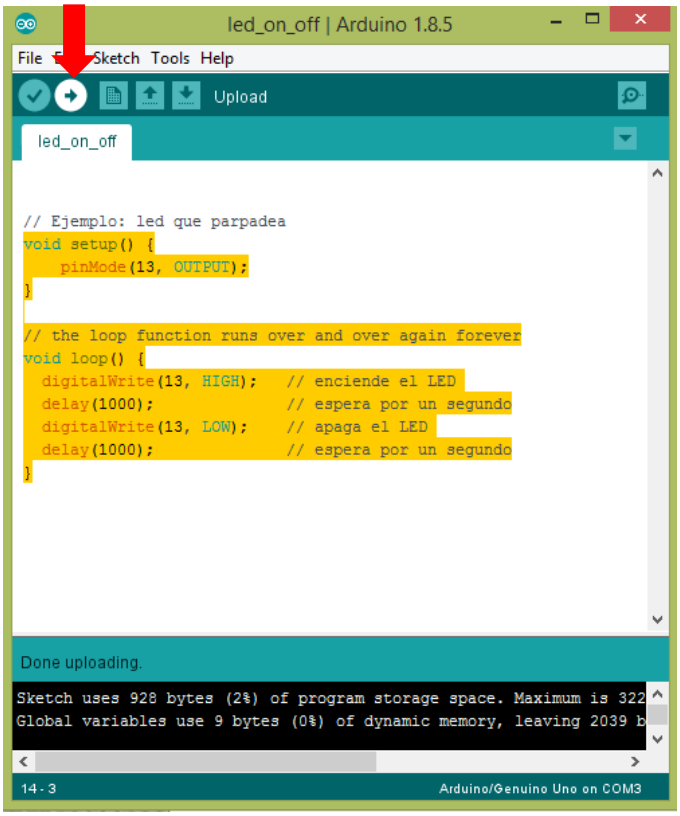
En el caso que contamos con la placa Arduino, la conectamos a la PC mediante una conexión USB. El software de Arduino contiene los drivers para estas plaquetas por lo que al conectarla el sistema detectara a estas como un puerto serie.

En en el entorno Arduino deberemos seleccionar en que puerto se encuentra conectado nuestro Arduino, para esto vamos al menú **Tools->Port** y seleccionamos el que corresponda (para el caso de la figura COM3):

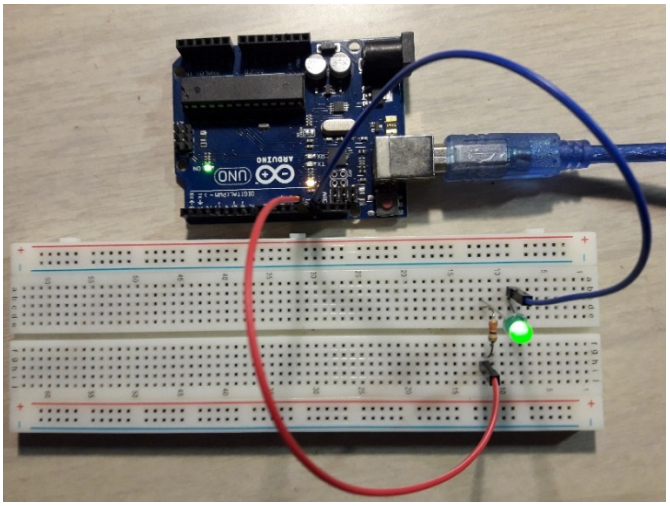
- **Si tenemos mas de un dispositivo conectado el entorno arduino mostrara varios COM para conectar, para concer cual corresponde a nuestra placa deberemos acceder a la configuracion de nuestro sistema operativo y buscar la opcion DEVICES, alli encontraremos el listado de dispositivos con sus correpondientes puertos.**
- **Aunque en general el entorno arduino detecta automaticamente las placas, existen ciertos clones ARDUINO los cuales pueden necesitar la instalcion de drivers adicionales el mas comun es el driver CH341**



Una vez configurado el IDE de Arduino y conectada la placa, procedemos a cargar el programa en la misma. Para esto le damos click al botón flecha en la barra superior:



Luego de un instante cargara el programa en el Arduino y este comenzara a ejecutarlo:

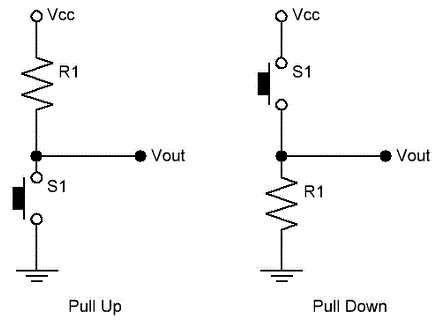


Cuidado!!! Cuando se pasa el programa al Arduino, inmediatamente terminada la carga el programa se ejecutará.

Otras función de interés:

valor = digitalRead(pin);
Lee el valor digital del "pin" definido previamente como INPUT, Esta función devolverá un 1 o un cero, el cual es posible guardar en una variable de tipo int (aquí valor).

- **Al utilizar pines digitales como entradas, debemos asegurarnos que estas entradas no queden sin conexión flotando, ya que el microcontrolador podria detectar cualquier estado. Para forzarle un estado logico se utilizan los siguientes circuitos:**



El circuito de Pull Up fuerza a que la salida quede en "1" (al valor de Vcc) al no precionar el boton, cuando el boton es precionado este pone en "0" la entrada digital (conectada en el esquema a Vout). El circuito Pull Down es equivalente pero fuerza el estado "0" al no estar precionado el boton.

Actividad

1) **Realice y simule un programa donde se le conecten 6 diodos led con sus correspondientes resistencias a un Arduino Uno. El objetivo del programa es hacer que de a un diodo led por vez se valla encendiendo secuencialmente.**
2) **Realice y simule un programa donde a un Arduino Uno se le conecten 2 leds de distinto color, y 2 pulsadores uno con conexión Pull Up, y otro con conexión Pull Dow. El programa debe encender un led con cada pulsador, y en caso de estar ambos estén pulsados no debe encender ninguno.**
3) **Al problema 1) agréguele un pulsador en la configuración que desee. Al tocar este pulsador el sentido secuencial en el que los leds se venían encendiendo debe invertirse.**
4) **Realice el programa para una prensa industrial. Una prensa industrial es un elemento de operación delicada para un operario. Por esto se desea que al realizarse la operación de prensado el operario debe encontrarse apretando dos pulsadores de acción, garantizando que sus manos no se encontraran en la zona de trabajo. (simule la operación de prensado con el encendido de un led).**

- **Muchas de las estructuras del Lenguaje C son similares en Arduino, para ver una lista completa de las funciones y estructuras de control de Arduino vea la pagina de referencias dentro de la pagina web de arduino.**
  https://www.arduino.cc/reference/en/