Cursillo de Introducción

Para GFM

A.Navas 2018

Arduino Programa

- Pequeña historia.
- Qué es Arduino?
- Diferentes placas de Arduino.
- Arduino, microprocesador.
- Instalación del IDE Arduino en ordenador.
- El Harware Arduino.
- Arduino Schield.
- Alimentación eléctrica del Arduino.
- Sofware de Arduino.
- Memorias.
- Programando Arduino, un led. Circuito teórico y práctico.
- Progrmación en C++, primeros pasos.
- Programando. Control de flujo.
- Programando Arduino, varios leds.. Circuito teórico y práctico.

Pequeña historia I

- **Arduino**
- Arduino se inició en el año 2005 como un proyecto para estudiantes en el Instituto IVREA en Ivrea (Italia). En ese tiempo, los estudiantes usaban el microcontrolador BASIC Stamp, cuyo costo era de 100 dólares estadounidenses, el cual consideraban demasiado costoso para su propósito. Por aquella época, uno de los fundadores de Arduino, Massimo Banzi, daba clases en Ivrea.

Historia II

- El nombre del proyecto viene del nombre del *Bar di Re Arduino* (Bar del Rey Arduino), donde Massimo Banzi solía pasar algunas horas. El "rey Arduino" fue rey de Italia entre los años 1002 y 1014.
- Una vez concluida dicha plataforma, los investigadores trabajaron para hacerlo más ligero, más económico y de mayor alcance a la comunidad de hardware y código abierto.

Historia III

- **Arduino** es una compañía que diseña y manufactura placas de desarrollo de hardware para construir dispositivos digitales y dispositivos interactivos que puedan **sensar** y **controlar** objetos del mundo real.
- El IDE (entorno de desarrollo) libre y un lenguaje de programación propio, simplificado de arduino, es compatible con múltiples sistemas operativos, como Windows, Linux o Mac.

Qué es Arduino

- **Arduino** es una plataforma de electrónica "open-source"o de código abierto cuyos principios son contar con software y hardware fáciles de usar. Es un microcontrolador.
- Un **microcontrolador** (abreviado **µC**, **UC** o **MCU**) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: **unidad central de procesamiento**, **memoria** y **periféricos de entrada/s**alida.
- **Arduino** es utilizado como un microcontrolador, cuando tiene un programa descargado desde un ordenador y funciona de forma independiente de éste, y controla y alimenta determinados dispositivos y toma decisiones de acuerdo al programa descargado e interactúa con el mundo físico gracias a sensores y actuadores.

Arduino UNO I

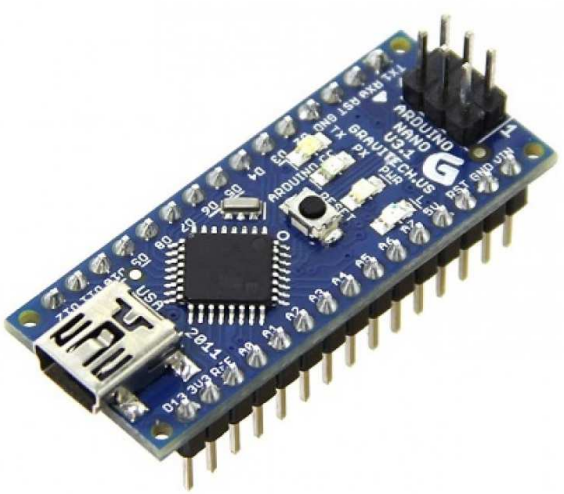
Microcontrolador en formato DIP

Arduino UNO Compatible

Microcontrolador con formato SMD

Otros Arduinos

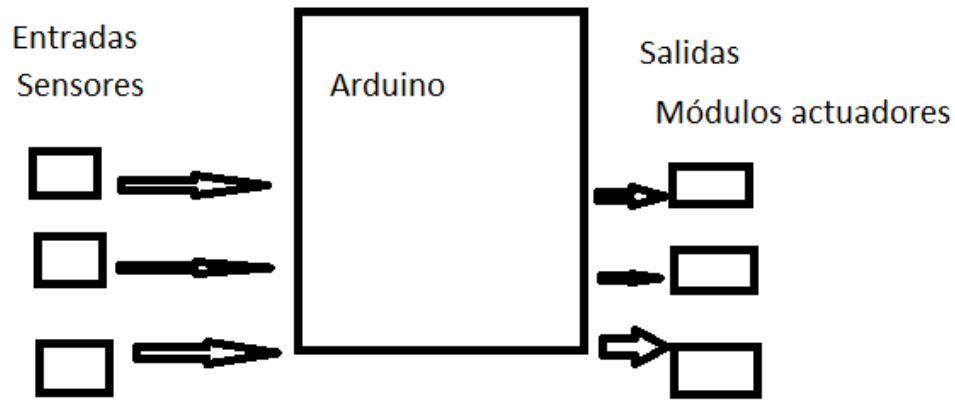
Arduino NANO



Arduino MEGA

Arduino - Microprocesador

Luz
Temperatura
Distancia
Presión
Humedad
Electricidad
Sonido
Gases
Recon. De voz
Ethernet
Wifi
Bluetoots
Inclinación
Un Interruptor
etc.



Infinitas posibilidades.

Variables físicas

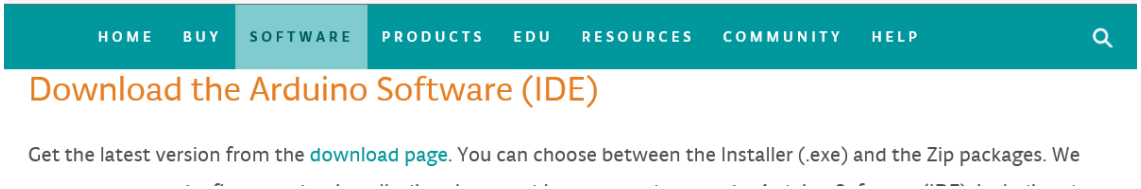
Leds
Relés
Transistores
C. Integrados
Motores
Pantallas LCD
Tarjetas SD
Zumbador
Paso a paso
Display 7 Seg.
Impresora 3D
Pantalla SPI
TFT
Etc.

Instalación de sofware I

Install the Arduino Software (IDE) on Windows

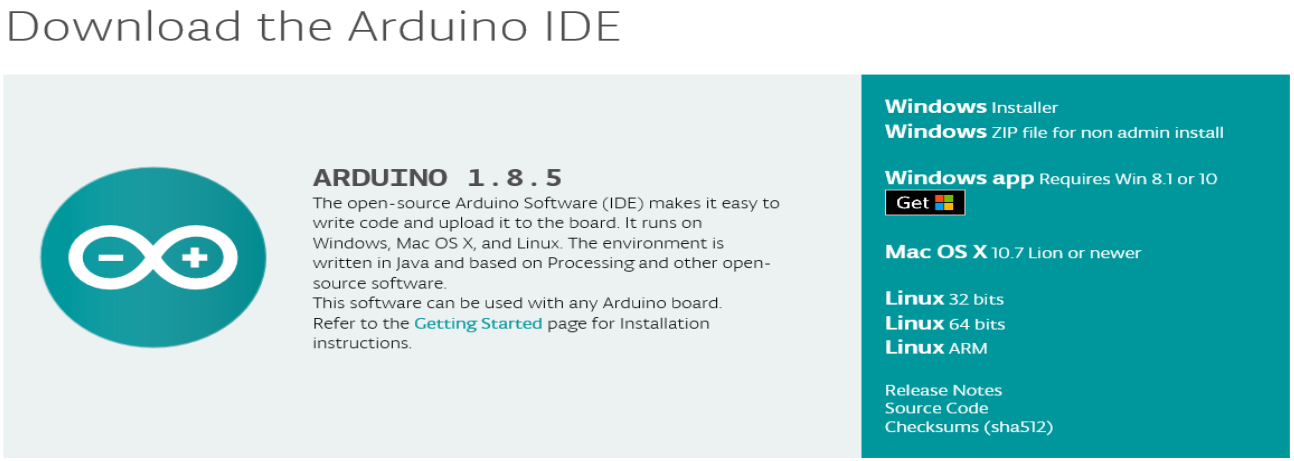
https://www.arduino.cc/en/Guide/Windows ▾ Traducir esta página

9 ago. 2016 - Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the **Arduino** Software (IDE), including the drivers. With the Zip package you need to **install** the drivers manually.

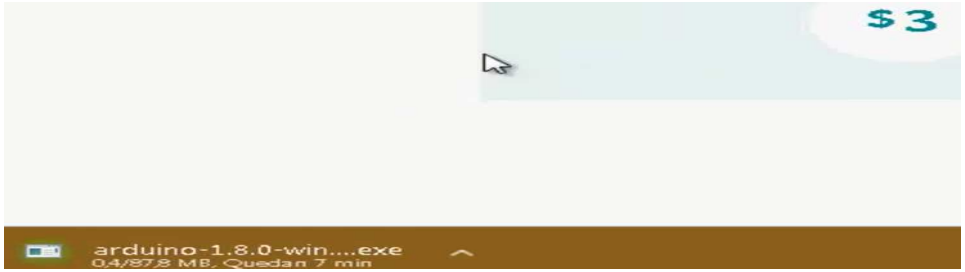


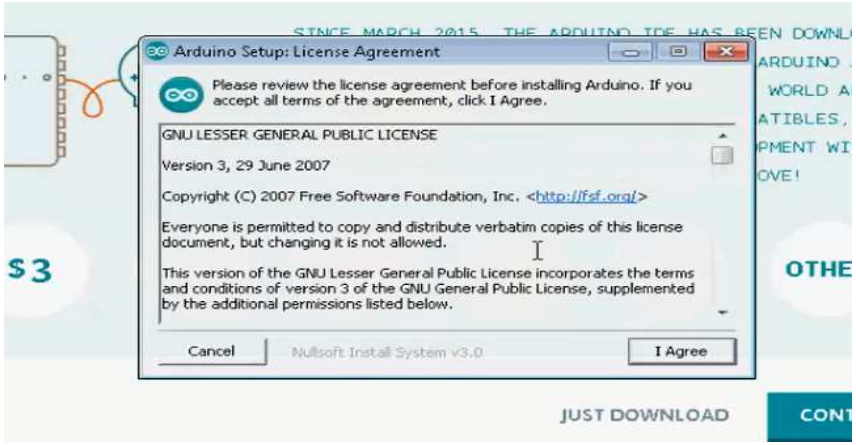
Instalación de sofware II

Para Windows 8 y 10

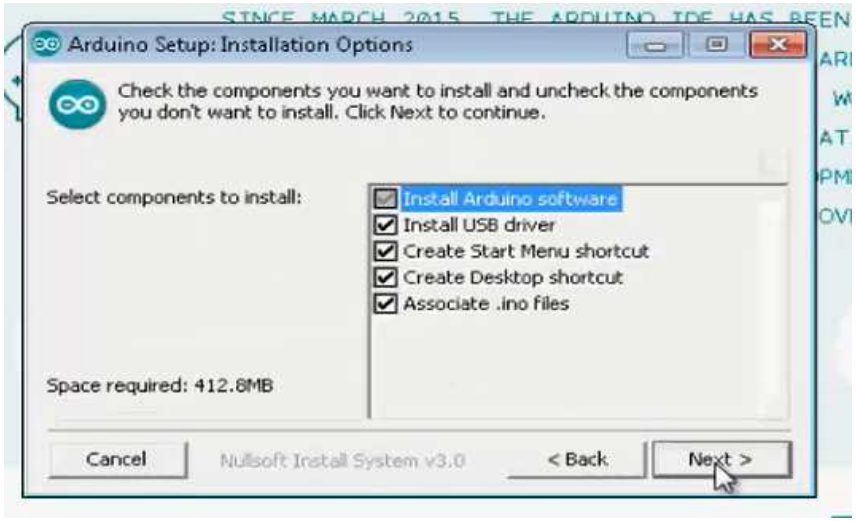


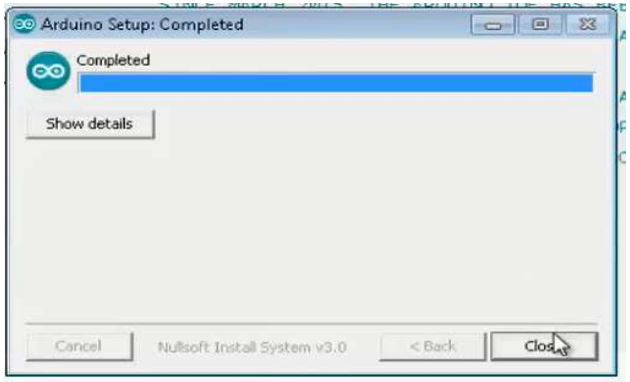
Instalación Arduino III





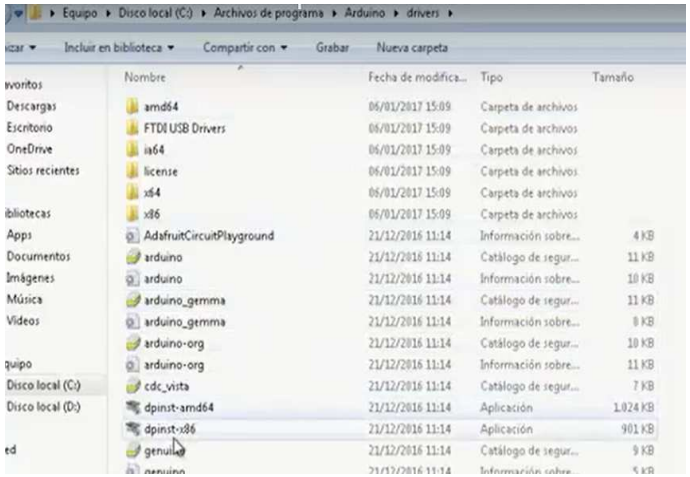
Instalación Arduino IV

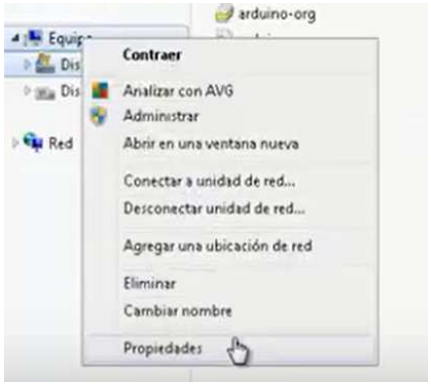




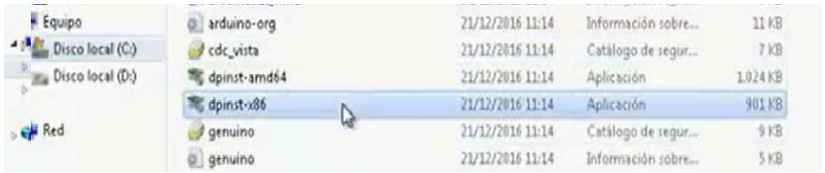
Instalación Arduino V

- Por si no reconoce Arduino.
- Instalaciòn de Drivers:

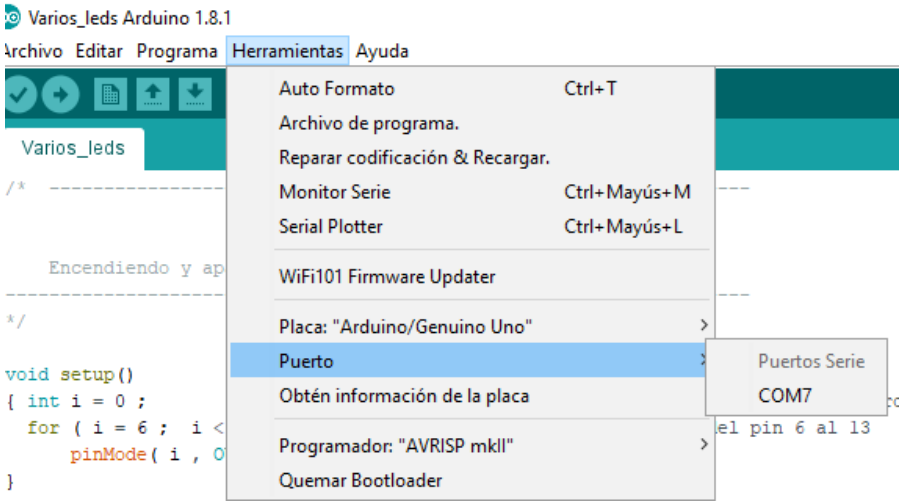


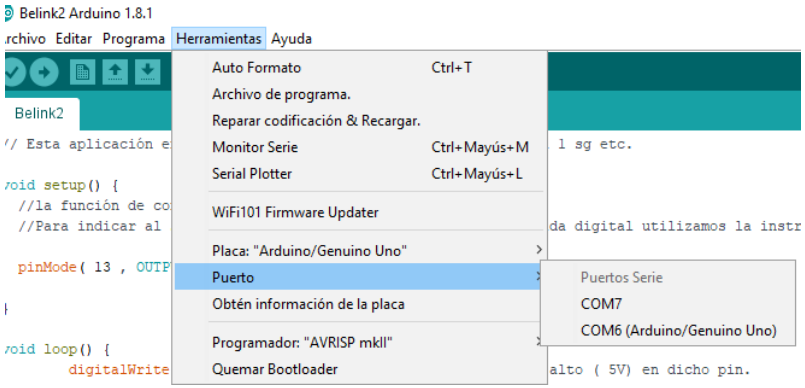


Instalación Arduino VI



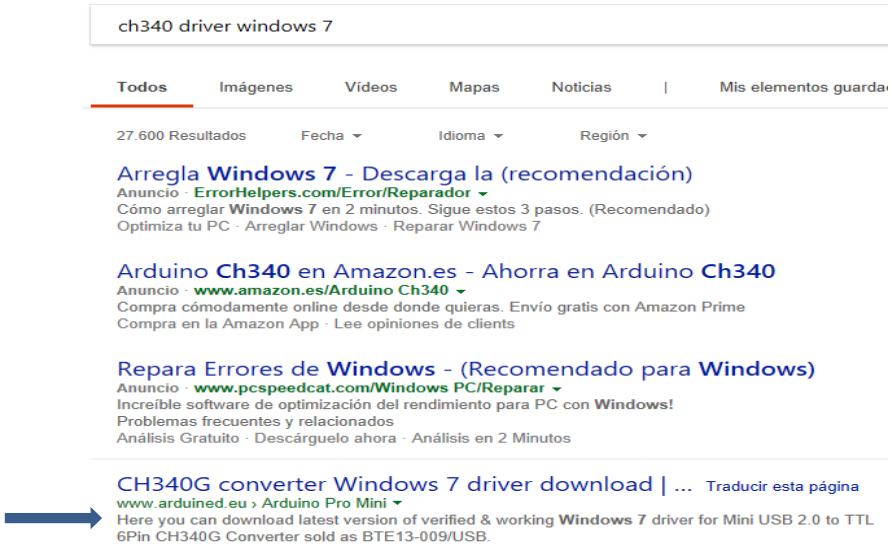
Instalación Arduino VII

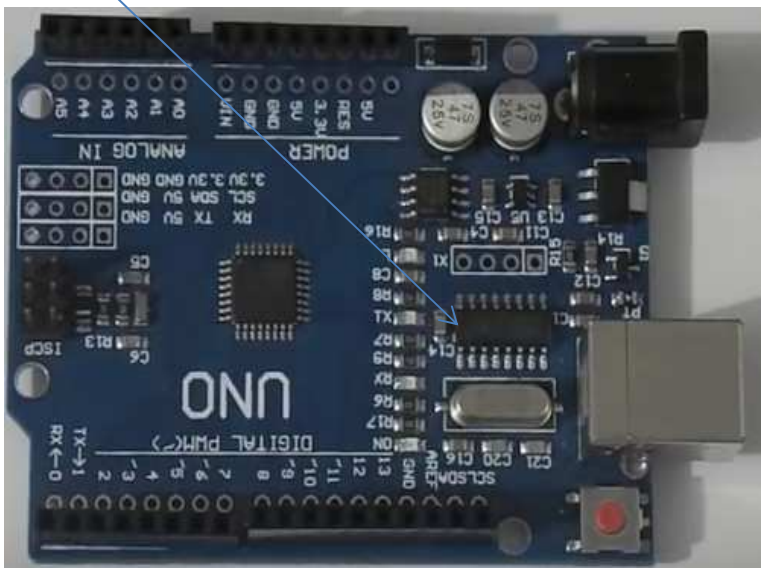




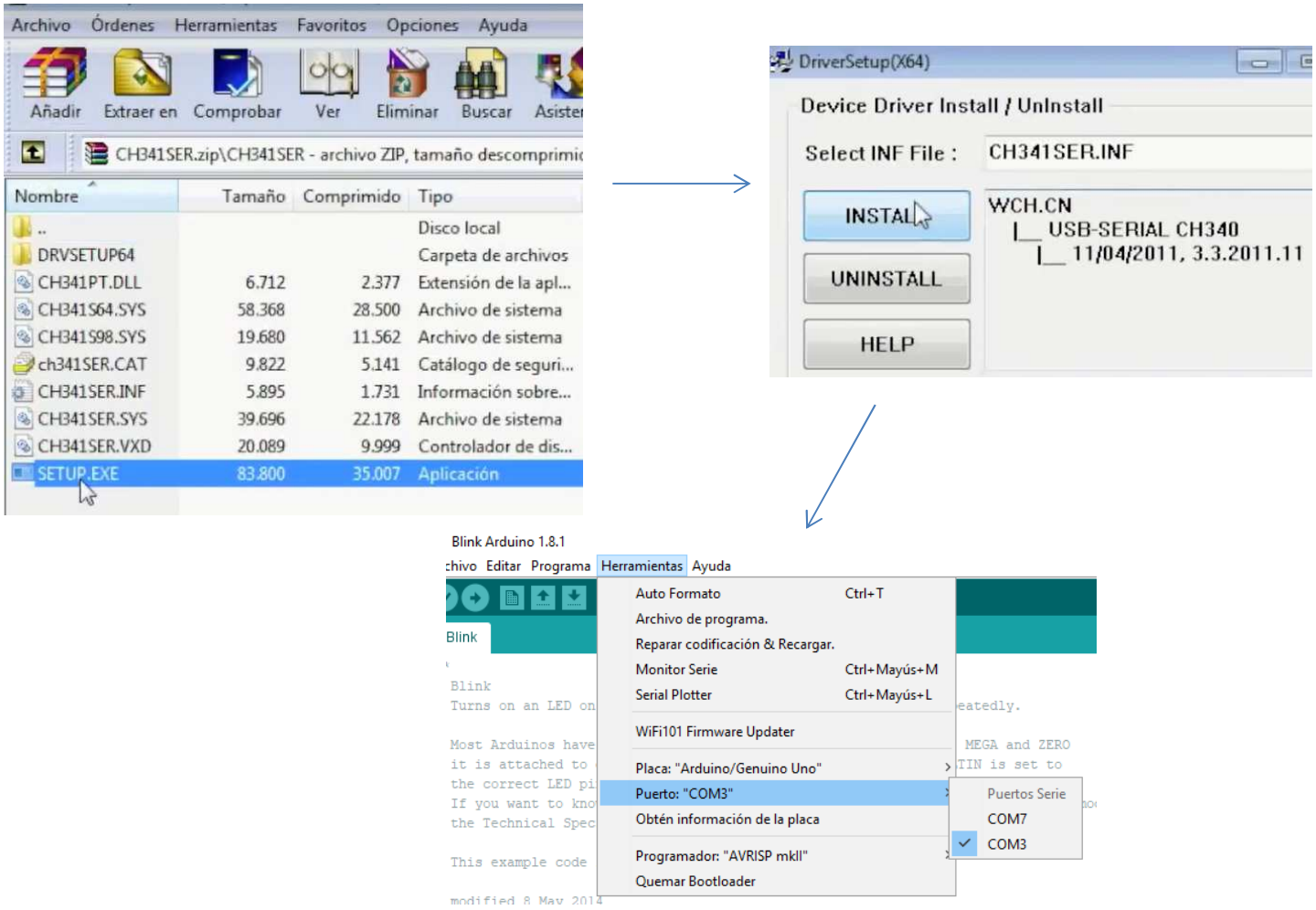
Instalación Arduino (Compatible) I

Algunos Arduinos chinos necesitan el drivers CH340.





Instalación Arduino (Compatible) II



Familia Arduino

Arduino mini
Pro mini
Nano
Fio
Lilypad
Leonardo
Pro
Uno
Ethernet
Mega 328
Mega 1280
Mega ADK

https://www.robotshop.com/blog/en/arduino-microcontroller-feature-comparison-2-3631

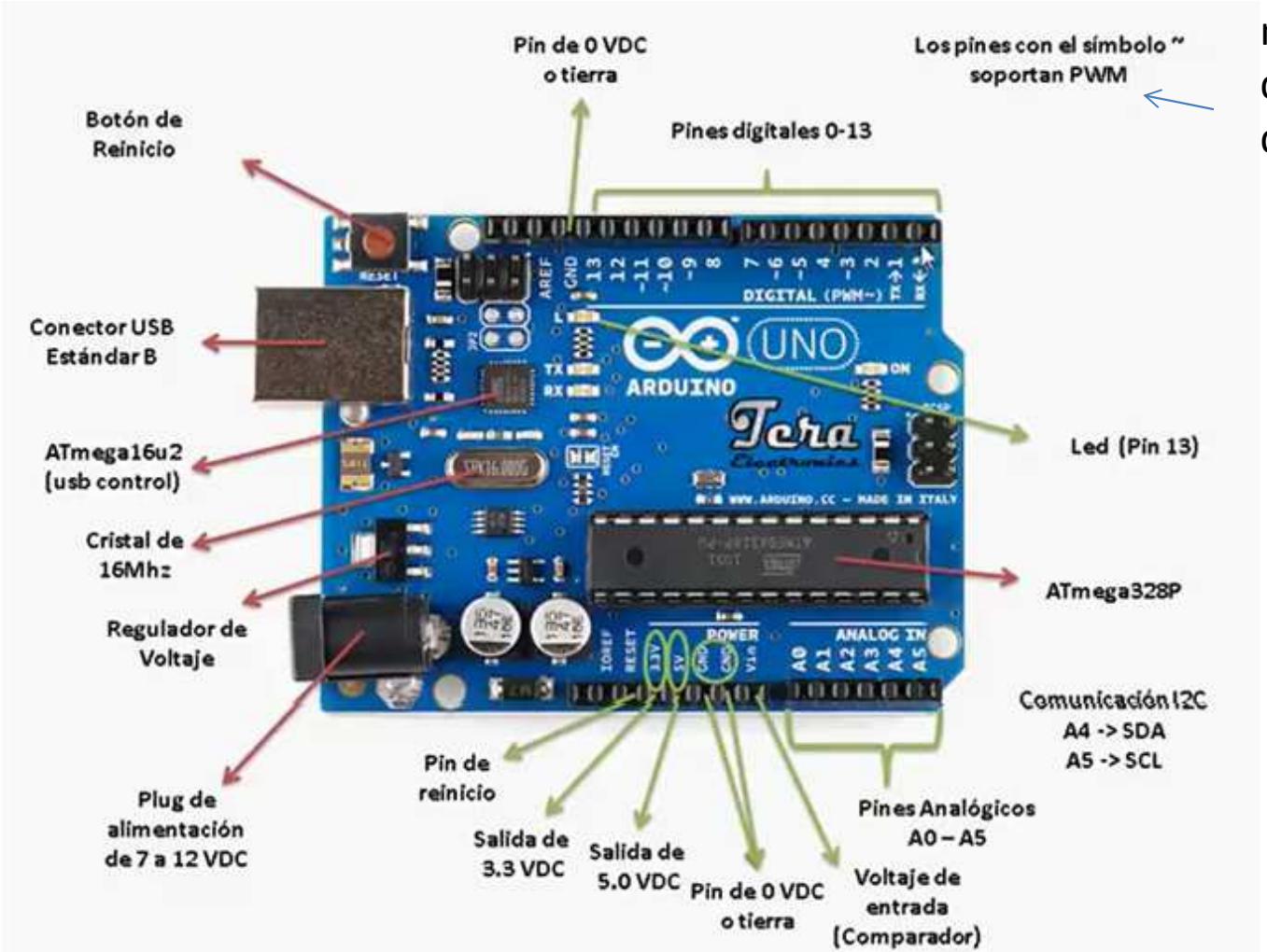
Arduino Hardware

- **Arduino es tanto software como hardware.**
- **El hardware Arduino** más sencillo consiste en una **placa con un microcontrolador y una serie de puertos de entrada y salida**. Los microcontroladores AVR más usados son el Atmega168, **Atmega328**, **Atmega128**, y **Atmega2560** por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños.
- http://www.microchip.com/wwwproducts/en/ATmega328p
- http://www.alldatasheet.com/view.jsp?Searchword=Datasheet%20atmega328&gclid=EAIaIQobChMIoYGIrPD_2QIVDGYbCh0bYQPnEAAYAiAAEgIJ1fD_BwE

Arduino Hardware

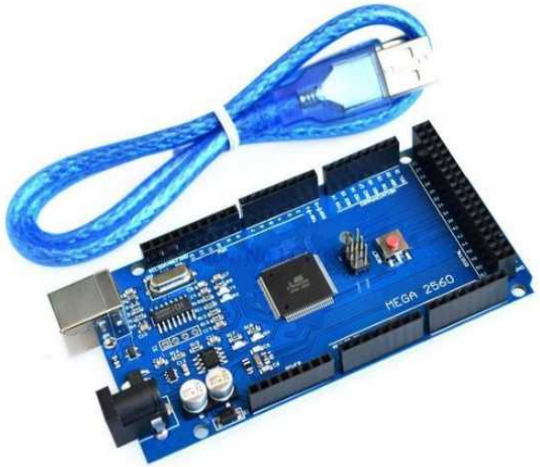
MODELO	CARACTERÍSTICAS
ARDUINO UNO 	- **Microcontrolador:** ATmega328 - **Voltaje de funcionamiento:** 5 V - **Pines I/O digitales:** 14 (de los cuales 6 proveen salida PWM) - **Pines de entradas análogas:** 6 - **Corriente DC por cada pin I/O:** 40 mA - **Corriente DC en el pin de 3.3 V:** 50 mA - **Memoria Flash:** 32 KB (ATmega328) de los cuales 0.5 KB son utilizados por el bootloader - **SRAM:** 2 KB (ATmega328) - **EEPROM:** 1 KB (ATmega328) - **Velocidad de reloj:** 16 MHz

Arduino UNO Hadware



6 salidas modulación de ancho de pulsos.

Arduino MEGA Hardware



Arduino Mega posee las siguientes especificaciones:

- **Microcontrolador:** ATmega2560
- **Voltaje Operativo:** 5V
- **Voltaje de Entrada:** 7-12V
- **Voltaje de Entrada(límites)**: 6-20V
- **Pines digitales de Entrada/Salida:** 54 (de los cuales 15 proveen salida PWM)
- **Pines análogos de entrada:** 16
- **Corriente DC por cada Pin Entrada/Salida:** 40 mA
- **Corriente DC entregada en el Pin 3.3V:** 50 mA
- **Memoria Flash:** 256 KB (8KB usados por el bootloader)
- **SRAM:** 8KB
- **EEPROM:** 4KB
- **Clock Speed:** 16 MHz

Arduino NANO

ATmega168 a 16Mhz

necesita de un cable mini-USB y no posee conector de alimentación externa.

con 14 pines digitales (6 PWM) y 8 analógicos.

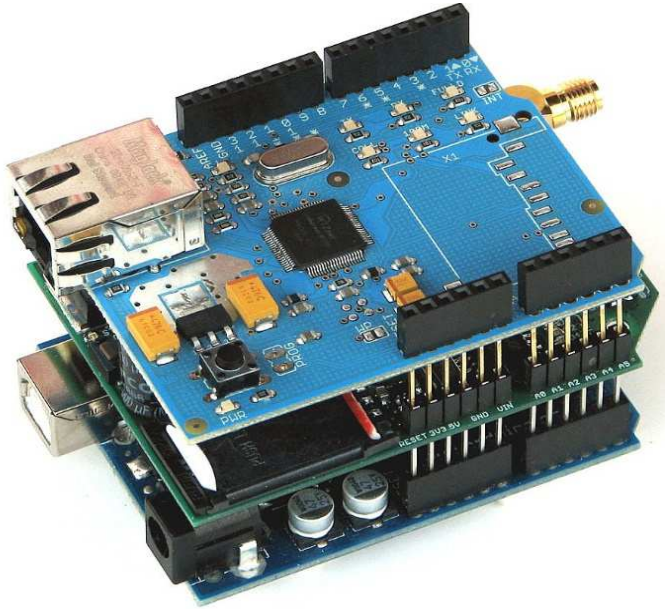
16KB de flash (2 reservados al cargador de arranque), de 2 a 1KB de SRAM y de 1KB a 512 bytes de EEPROM.

Su reducido tamaño no le quitan la posibilidad de ser una placa completa

Arduino Shields

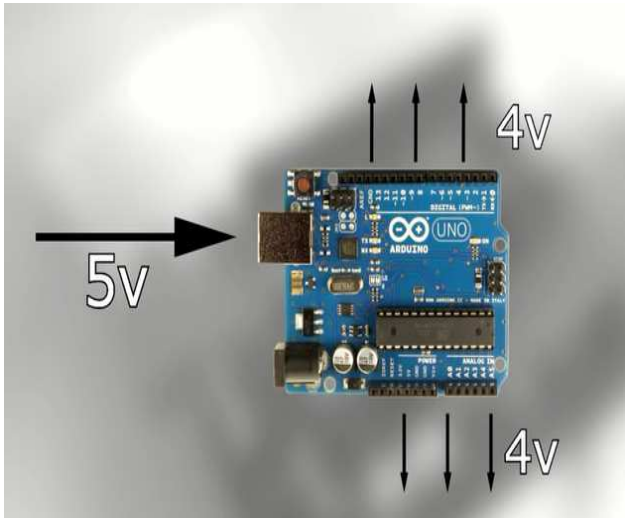
-Las shields son placas de circuitos modulares que se montan unas encima de otras para dar funcionalidad extra a un Arduino. Esta Shields son apilables.

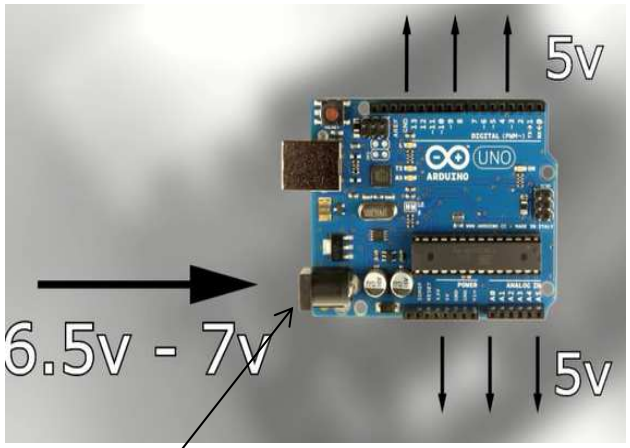
- Un shield en Arduino es un placa que se apila sobre el arduino o sobre otro shield, de forma que nos permite ampliar el hardware/capacidades de Arduino. (Wifi, Bluetooth, Relay Driver, Arduino Motor, GSM, etc.).



Las shields se pueden comunicar con el arduino bien por algunos de los pines digitales o analógicos o bien por algún bus como el SPI, I2C o puerto serie, así como usar algunos pines como interrupción. Además estas shields se alimenta generalmente a través del Arduino mediante los pines de 5V y GND.

Alimentación Arduino UNO



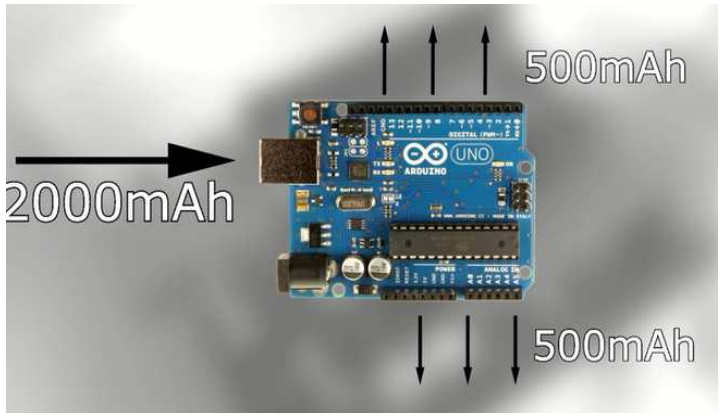


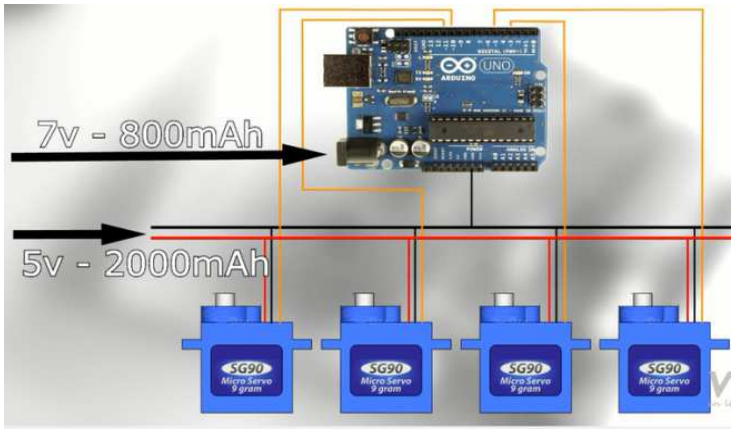
- Se puede alimentar desde 5v. Hasta 20 v.

-Rango recomendado de 7v. hasta 12v.
- Ideal de 7 a 9 v. El exceso de 5v calienta la placa.

Consumos Arduino I

Sólo hay un fusible en la placa arduino. Hay un Fusible Polyfuse en el puerto USB que limita la corriente de partida en 500mA. Este fusible es eficaz sólo cuando se alimenta por USB y sólo cuando el consumo total en los 5V es más de 500mA
Si está alimentado por USB la corriente que el arduino puede entregar al exterior son 500 ma. Es decir, no importa si la corriente de entada es de 2000 mah, ya que solo puede dar 500 m





Por lo que es conveniente alimentar los módulos, etc. con una fuente de alimentación externa.

Ojo¡ importante unir los GND, el del arduino y el de la otra fuente.

Consumos Arduino II

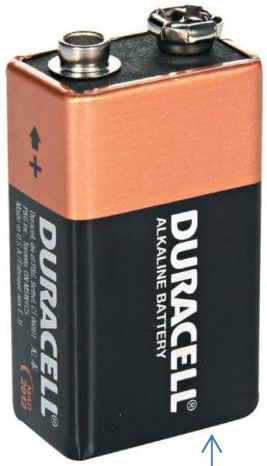
Calculando la duracion de la bateria

Modelo	Consumo en mA	Duracion de una bateria de 1200 mAh
Arduino UNO	46	26 horas
Arduino MEGA	93	Casi 13 horas
Arduino DUE	75	16 horas
Arduino Nano	15	80 horas

Corriente DC por Pin e/s **: 40.0 mA.**

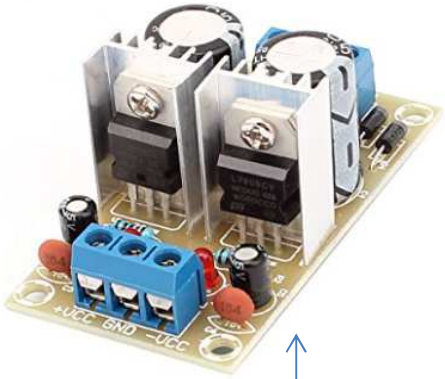
Corriente DC por Pines de VCC y GND : **200 mA.**

Alimentación por el Plug

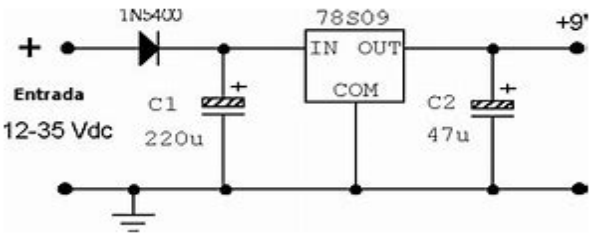


Batería de 9V.

Alimentador
Cargador 9V.



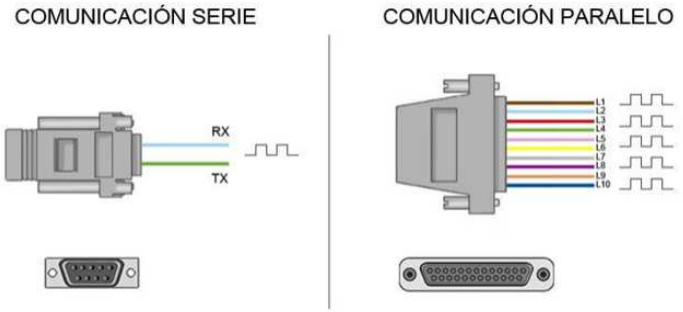
Regulador 9V.



Limitador L7809CV

Puertos de Arduino

Puertos o interface físico o virtual que permiten la comunicación entre dos ordenadores o dispositivos



Tipos de puertos serie, como por ejemplo el **I2C**, **SPI**, **Serial**, Ethernet o FireWire, entre otros.

Algunos Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos de serie, lo que simplifica el proceso de conexión con un ordenador. Principalmente para transferir el código compilado del ordenador al arduino.

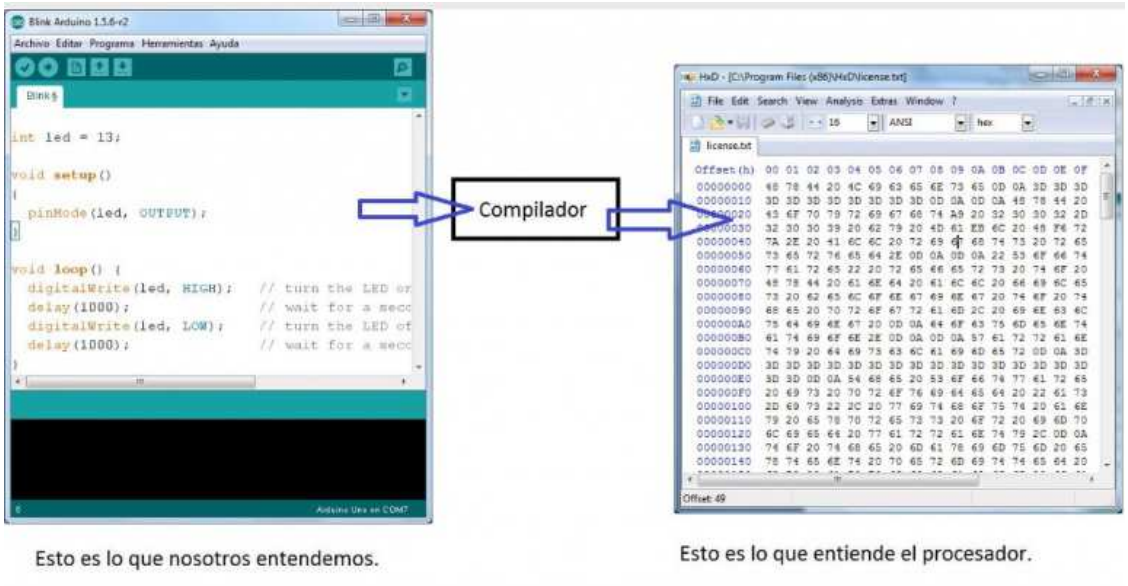
- Pines **Tx/Rx** Para señales TTL.
- Pines **10**, **11**, **12** y **13** Comunicaciones **SPI** Ful duplex maestro/esclavo.

Sofware Arduino I

- Arduino se programa en una variante de C++ , que es un lenguaje muy extendido por sus características, aunque no es un lenguaje sencillo. C++, que fija reglas estrictas de cómo escribir estas instrucciones.
- Un programa es una serie de instrucciones que se ejecutan en secuencia (salvo que indiquemos expresamente condiciones precisas en las que esta secuencia se altera).
- Un programa interno comprueba que la sintaxis de nuestro programa es acorde a la norma de C++, y si hay cualquier cosa que no le convence dará un error y finalizará la comprobación obligándonos a revisar lo que hemos escrito.
- Cuando el comprobador acepta nuestro programa, invoca otro programa que traduce lo que hemos escrito a instrucciones comprensibles para el procesador de nuestro Arduino. A este nuevo programa se le llama compilador.

Software Arduino II

Cuando el comprobador acepta nuestro programa, invoca otro programa que traduce lo que hemos escrito a instrucciones comprensibles para el procesador de nuestro Arduino. A este nuevo programa se le llama compilador.



Funcion del compilador

Unidades de memoria

Unidad	Descripción
Bit (Dígito binario)	Un dígito binario es lógico 0 y 1 que representa un pasivo o un estado activo de un componente en un circuito eléctrico.
Nibble	Un grupo de 4 bits se llama nibble.
Byte	Un grupo de 8 bits se llama byte. Un byte es la unidad más pequeña que puede representar un elemento de datos o un personaje.
Palabra	Una palabra de computadora, como un byte, es un grupo de número fijo de bits procesados como una unidad que varía de PC a PC pero es fijo para cada ordenador. La longitud de una palabra de computadora se llama tamaño de palabra o palabra de longitud y puede ser tan pequeño como 8 bits o puede ser tan largo como 96 bits. Una computadora almacena la información en forma de palabras de computadora.

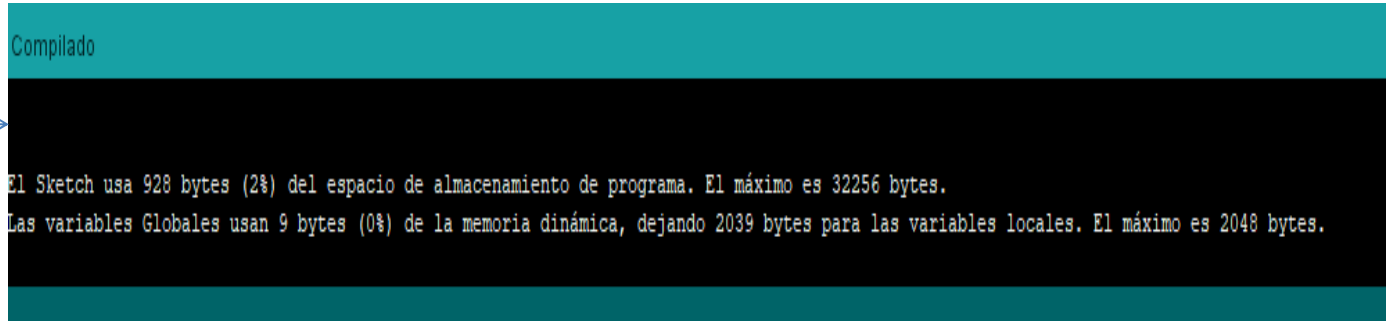
Unidad	Descripción
Kilobyte (KB)	1 KB = 1024 Bytes
Megabyte (MB)	1 MB = 1024 KB
GigaByte (GB)	1 GB = 1024 MB
TeraByte (TB	1 TB = 1024 GB
PetaByte (PB)	1 PB = 1024 TB

Memorias de Arduino I

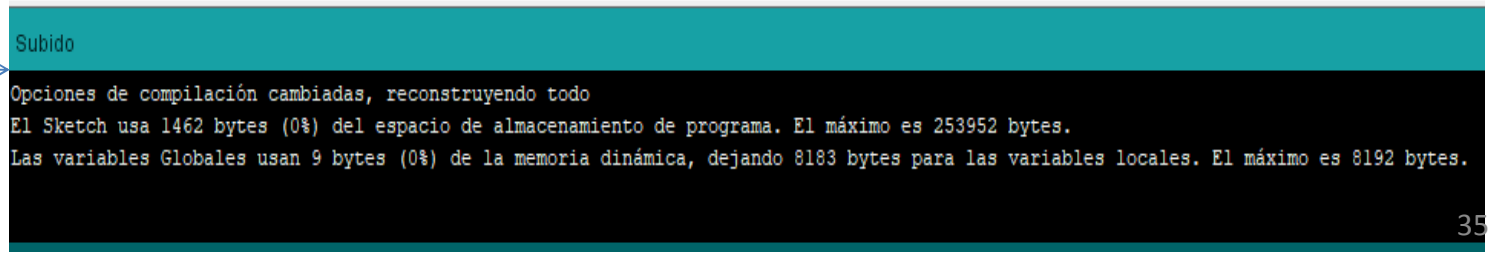
Arduino y todos los microcontroladores tienen varios tipos de memoria integradas en el mismo chip, en el caso de Arduino usa tres tipos de memoria:

Arduino	Processor	Flash	SRAM	EEPROM
UNO, Uno Ethernet, Menta, Boarduino	Atmega328	32K	2K	1K
Leonardo, Micro, Flora, 32U4 Breakout, Teensy, Esplora	Atmega 32U4	32K	2.5K	1K
Mega, MegaADK	Atmega2560	256K	8K	4K

UNO →



MEGA →



Memorias de Arduino II

- **SRAM** (static random access memory): Variables locales, datos parciales. Usualmente se trata como banco de registros y memoria volátil. Es la zona de memoria donde el sketch, debemos supervisar su uso para evitar agotarlo. En el Arduino UNO = 2K --> 2 x 1.024 bytes = 2.048 bytes.

- **EEPROM**: Memoria no volátil para mantener datos después de un reset. Se puede grabar desde el programa del microcontrolador, usualmente, constantes de programa. es decir, una memoria no volátil (como la de un disco duro) en la que puedes almacenar datos sin preocuparte de perderlos cuando le quites la alimentación a tu Arduino. Las EEPROMs tienen un número limitado de lecturas/escrituras, tener en cuenta a la hora de usarla. La vida útil de la EEPROM es de unos 100.000 ciclos de escritura.

- **Flash**: Memoria de programa. Usualmente desde 1 Kb a 4 Mb (controladores de familias grandes). Es donde se guarda el sketch ya compilado. Sería el equivalente al disco duro de un ordenador. En la memoria flash también se almacena del bootloader.
  La memoria flash usa la misma tecnología que las tarjetas SD, los pendrives o algunos tipos de SSD, esta memoria tiene una vida útil de unos 100.000 ciclos de escritura, así que cargando 10 programas al día durante 27 años podríamos dañar la memoria flash.

  En el Arduino UNO = 32 K --> 32 x 1.024 = 32.768 bytes

  32.768 - 32.256 = 512 bytes que se reserva para el Bootloader. (es el nombre en inglés para el gestor de arranque del dispositivo. Es el primer programa que se ejecuta en el micro controlador cuando se enciende).

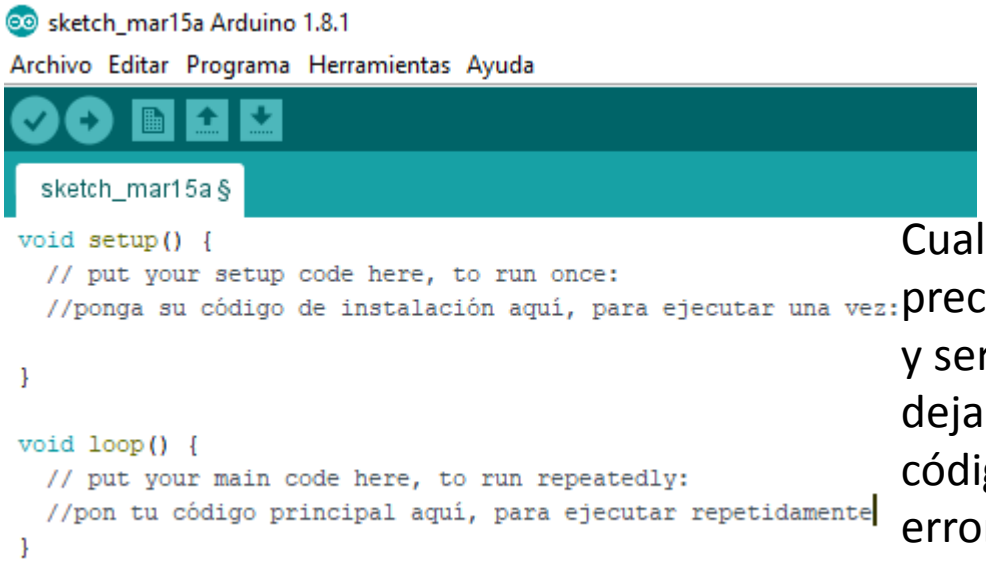
Tipos de memoria de Arduino

Tipos de Datos	Memoria que ocupa	Rango de valores
boolean	1 byte	0 o 1 (True o False)
byte / unsigned char	1 byte	0 – 255
char	1 byte	-128 – 127
int	2 bytes	-32.768 – 32.767
word / unsigned int	2 bytes	0 – 65.535
long	2 bytes	-2.147.483.648 – 2.147.483.647
unsigned long	4 bytes	0 – 4.294.967.295
float / double	4 bytes	-3,4028235E+38 - 3,4028235E+38
string	1 byte + x	Array de caracteres
array	1 byte + x	Colección de variables

Programando Arduino I

- Un programa o **sketch** de Arduino consiste en **dos** secciones o funciones básicas:
- ***Setup****: Sus instrucciones se ejecutan solo una vez, cuando se arranca el programa al encender Arduino o cuando pulsamos el botón de reset.*
- ***Loop****: Sus instrucciones se van ejecutando en secuencia hasta el final…. Y cuando acaba, vuelve a empezar desde el principio haciendo un ciclo sin fin.*

Programando Arduino II



Cualquier cosa que escribamos precedido por " // " son comentarios, y serán ignorados. Es decir podemos dejarnos mensajes dentro del código,(que de otro modo darían errores). El compilador ignorará cualquier cosa entre // y el fin de línea.

- El principio de cada función es indicado por la apertura de llave " **{** " y el fin de la misma corresponde al símbolo de cerrar llaves " **}** ".

- El conjunto de instrucciones contenidas entre una apertura y cierre de llaves se llama **bloque.**

Encender y apagar un led externo

- Pediremos a Arduino que active su pin 13 como de salida digital y después encenderemos y apagaremos esta señal lo que hará que el LED que tiene conectado de serie se encienda o apague al ritmo que marquemos.
- Para indicar al sistema que deseamos usar el pin 13 como salida digital utilizamos la instrucción:

pinMode (13, OUTPUT) ;

El primer parámetro indica el pin a usar y "OUTPUT" es para usarlo como salida en la función setup()

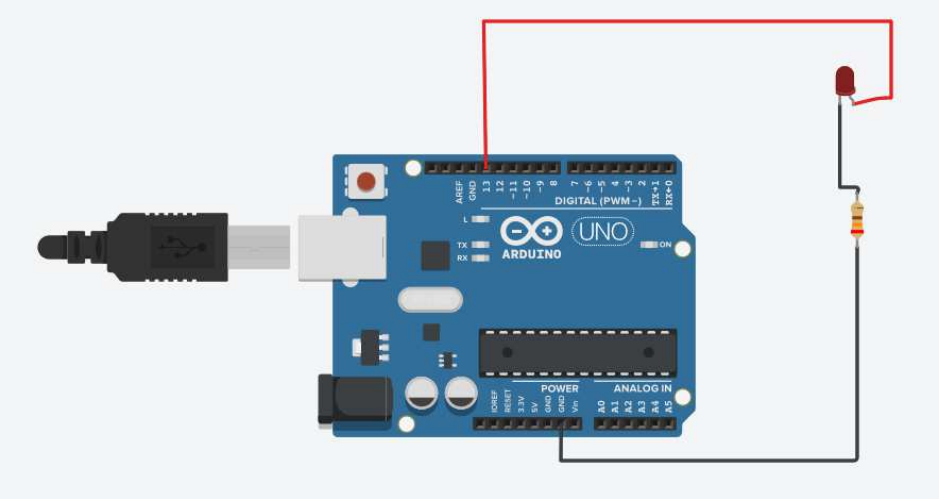
La instrucción finaliza en " **;**" . C++ obliga a acabar las instrucciones con un punto y coma que delimite la orden. Si se omite generará un error.

*Para encender el LED usaremos la instrucción: **digitalWrite(13 , HIGH) ;**

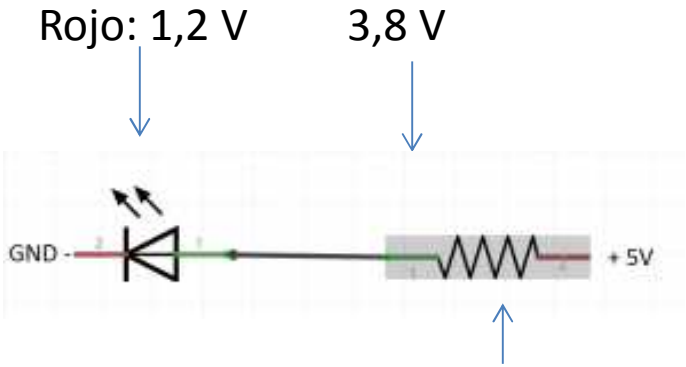
*Y otra instrucción similar que le ordena apagarlo: **digitalWrite(13 , LOW) ;**

delay(1000) ; **//** delay(n) "congela" Arduino n milisegundos

Circuit Teòric



Tensión Total : 5V

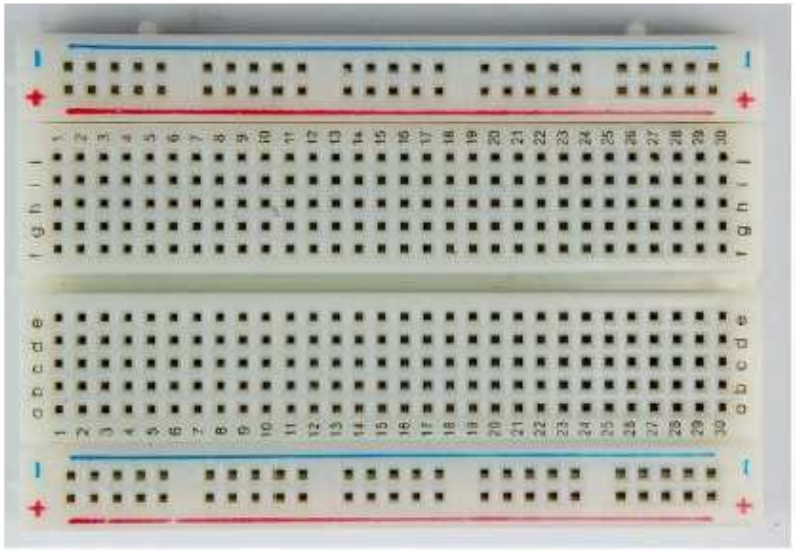


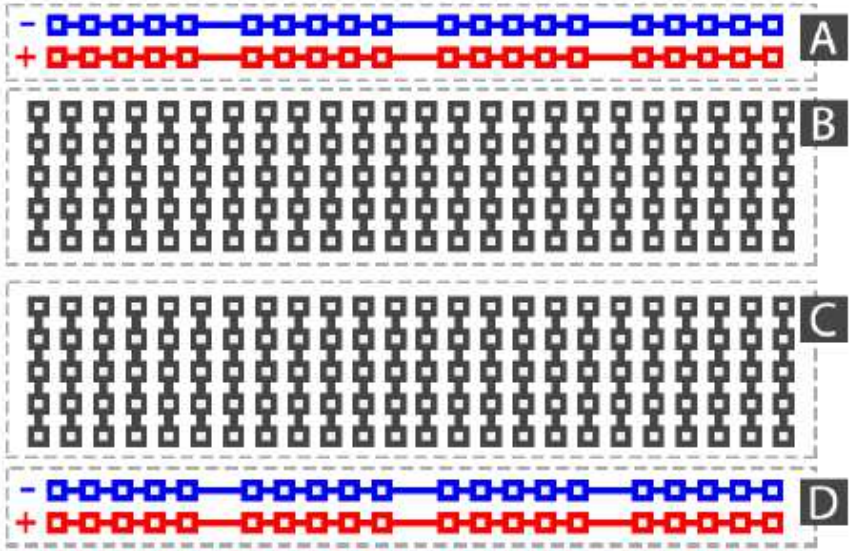
Ley de Ohm : I = V/R --> R = V / I

Resistencia eléctrica (Ohm) = Tensión (Voltios / Intensidad (Amperios)

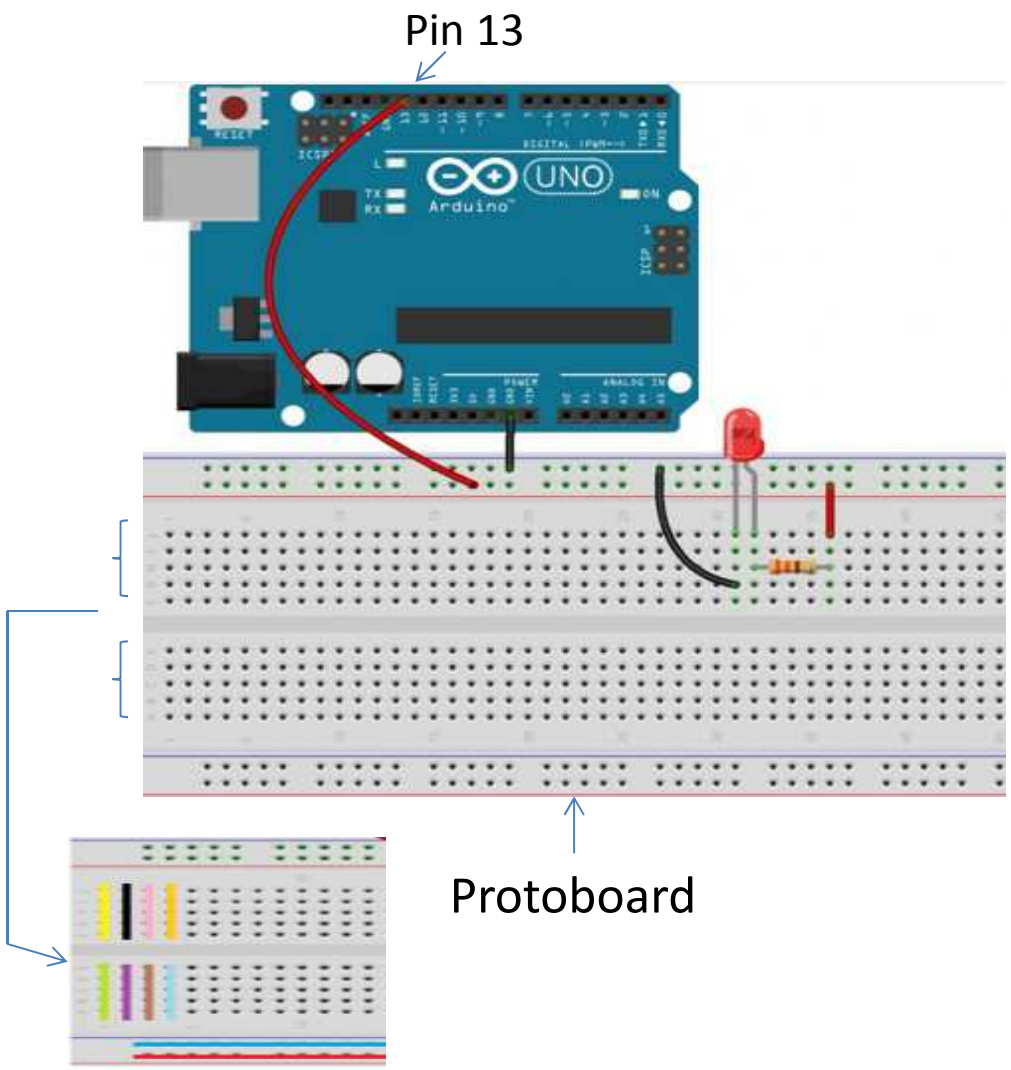
En este caso : **R** = 3,8 V / 20 mA (0,02 A) = 380 / 2 = **160 Ohm. (230)**

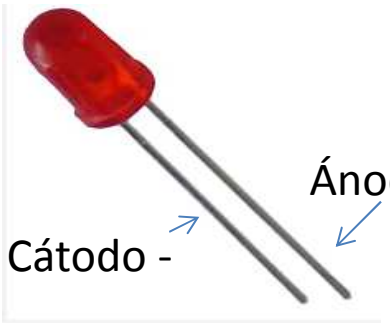
Protoboard – Connexió interna





Circuit Pràctic





Alimentación: 5V			
tipo de led	Vled	corriente	resistencia
azul / blanco alta luminosidad	3,7V	20 mA	(calculado: 65 ohm) **68 ohm**
rojo alta luminosidad	1,2V	20 mA	(calculado: 190 ohm) **180 ohm**
rojo tipo indicatore	1,2V	5 mA	(calculado: 760 ohm) **680 ohm**
verde / amarillo tipo indicatore	1,6V	5 mA	(calculado: 680 ohm) **680 ohm**

Codi

```
1  void setup()
2  {
3    pinMode(13, OUTPUT);
4  }
5
6  void loop()
7  {
8    digitalWrite(13, HIGH);
9    delay(1000); // Wait for 1000 millisecond(s)
10   digitalWrite(13, LOW);
11   delay(1000); // Wait for 1000 millisecond(s)
12 }
```

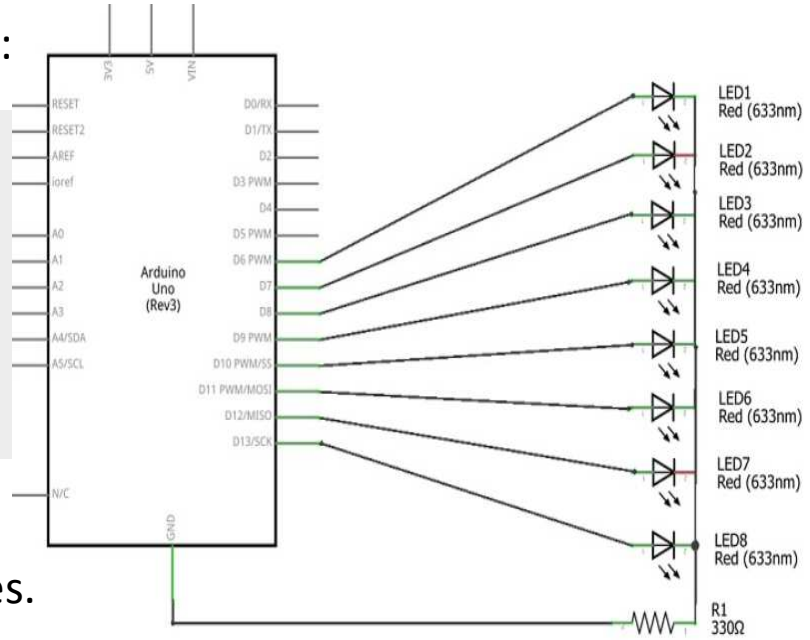
Muntatge amb diversos leds I

- **Objetivos**: - Conocer la instrucción for.
- - Utilizar variables.
- - Encender y apagar varios leds conectados a distintos pines.
- 1 - Asignación elemental de pines digitales.
- 2 - Asignación de pines mediante for.

Muntatge amb diversos leds II

Asignación de pines a salidas uno a uno:

```
void setup()
   {
          // initialize the digital pins as an output
         pinMode( 13, OUTPUT) ;
         pinMode( 12, OUTPUT) ;
         pinMode( 11, OUTPUT) ;
         ----------------------
         pinMode( 6, OUTPUT) ;
   }
```



- Para utilizar un **for** necesitamos variables.

- Una variable es un contenedor que puede tomar varios valores, en nuestro caso aceptará todos los valores entre 6 y 13.

C++ nos exige declarar el **tipo** de las variables antes de usarlas. En nuestro caso usaremos el tipo entero que se escribe **int** para indicar que esta variables es **numérica** y entera, sin decimales. En el **void setup()**
int i = 0 ; // Inicializamos la variable i como un entero (-32.768 a 32.767)

Aritmética C++

: Operadores aritméticas:

x = x +2;

y = y -7;

I = i * 5;

r = r/5;

Asignaciones compuestas:

x++; // lo mismo que x = x + 1

Y--; // lo mismo que y = y - 1

Operaciones de comparación: (en declaraciones **if)**

x == y; // Si x es igual a y

X != y; // Si x no es igual a y

X < y ; // Si x es menor que y

X > y ; // Si x es mayor que y

Control de flux I

- Función **if**
- Las sentencias if comprueban si cierta condición ha sido alcanzada y ejecutan todas las sentencias dentro de las llaves si la declaración es cierta. Si es falsa el programa ignora la sentencia.
- If (variable == 10)

```
{
  Enciende un led;
  }
```

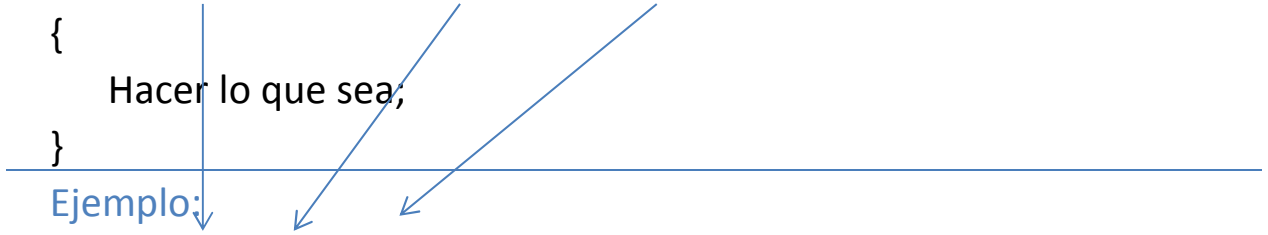
If…else .- Permite tomar decisiones “este o este”.

```
if (variable ==10)
 {
  Enciende el led rojo;
  }
  else
{
  Enciende el led verde;
  }
```

Control de flux II

- For

- La sentencia **for** se usa para repetir un bloque de declaraciones encerradas entre llaves un número específico de veces. Un contador de incremento se usa a menudo para incrementar y terminar el bucle. Hay tres partes separadas por punto y coma(;), en la cabecera del bucle.
- **For (inicialización; condición; expresión)**
- {
- Hacer lo que sea;
- }
- Ejemplo:



```
For (int i=0; i<20; i++)          //declara i, comprueba si es menor
{                                 // que 20, incrementa i en 1
    digitalWrite (13, HIGH);      // activa el pin 13
    Delay (250);                  // pausa por un ¼ de segundo
    digitalWrite (13, LOW);       // desactiva el pin 13
    Delay (250);                  // pausa por un ¼ de segundo
}
```

Muntatge amb diversos leds IV

- Supongamos que queremos encender 8 leds y que los vamos a conectar desde el pin digital 6 al 13. Si los asignamos en el
- void setup() { // uno a uno:
  pinMode (6, OUTPUT);
- pinMde(7,OUTPUT); } //… así sucesivamente, es muy pesado. Utilizando el for se asignaría automáticamente:

- /* ---
- Encendiendo y apagando 8 LEDs simultaneamente
- ---
- */
-
- void setup()
- { int i = 0 ; // Inicializamos la variable i como un entero
- for (i = 6 ; i < 14 ; i++) // asigna automáticamente del pin 6 al 13
- pinMode(i , OUTPUT) ;
- }

Muntatge amb diversos leds V

- La segunda parte, el void loop también se puede emplear el for:

```
void loop()                    // Programa 4.1
{   int i = 0 ;                // Definimos una variable tipo int
    for ( i = 6 ;  i < 14 ; i++)
     {
        digitalWrite( i , HIGH) ;   //Enciende el led i
        delay (500) ;               // Espera ½ segundo
        digitalWrite( i , LOW);     //Apaga el led i
        delay (500) ;               // Espera ½ segundo
     }

}
```

Muntatge amb diversos leds VI

Muntatge pràctic