

# Introducción a Arduino

Grupo Halley

Universidad Industrial de Santander



## ¿Qué es arduino?

Arduino es una plataforma de código libre diseñada para facilitar proyectos de electrónica. Posee un entorno gráfico de desarrollo que usa un lenguaje de programación processing/wiring y un gestor de arranque; en lo que respecta al hardware está compuesta por un microcontrolador y puertos de entrada y salida.

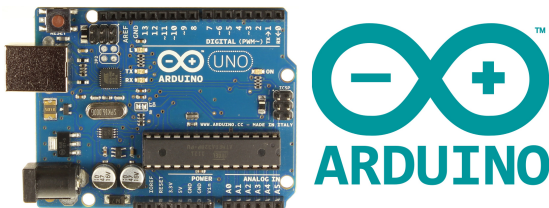


Figura 1: Targeta Y Logo Arduino

# Historia

Arduino inició como un proyecto de estudiantes del instituto IVREA(Italia), en el cual fue participe el colombiano Hernando Barragán quien propuso como tesis de grado la plataforma de programación wiring con la cual se programa el microcontrolador.

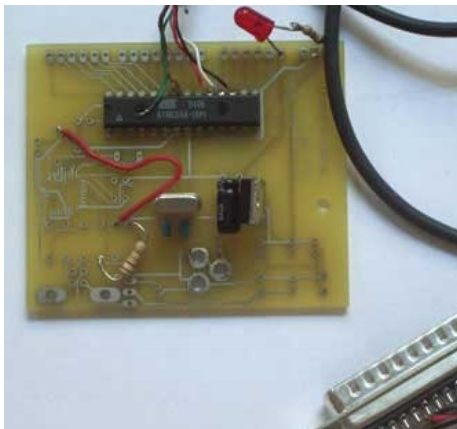


Figura 2: Prototipo

# Partes de Arduino

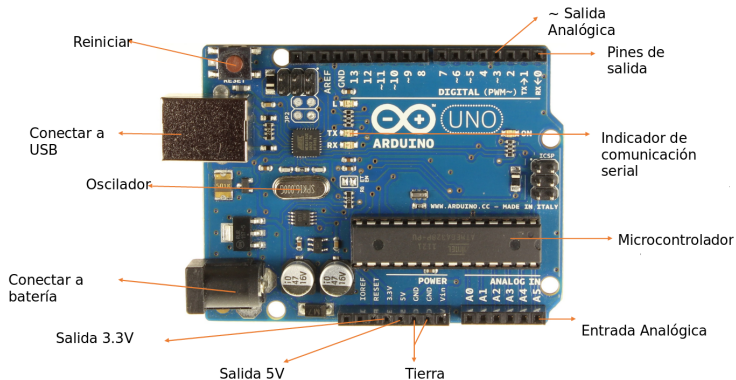


Figura 3: La Targeta Arduino

# Ventana principal

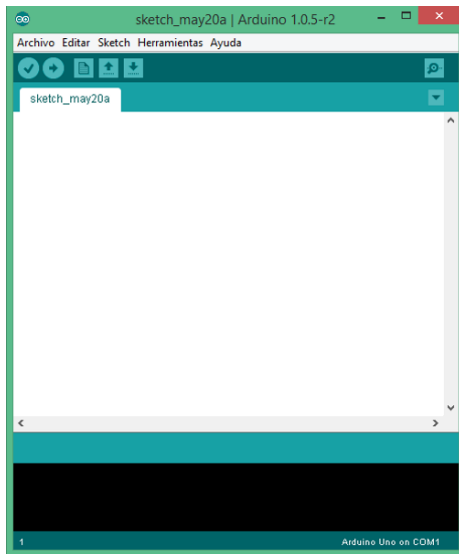


Figura 4: Ventana principal

# Partes de la ventana principal

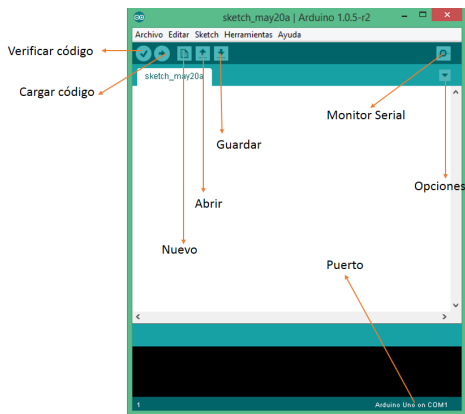


Figura 5: Botones principales

# Pestaña archivo

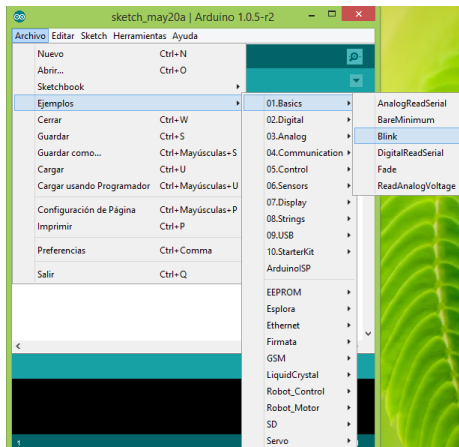
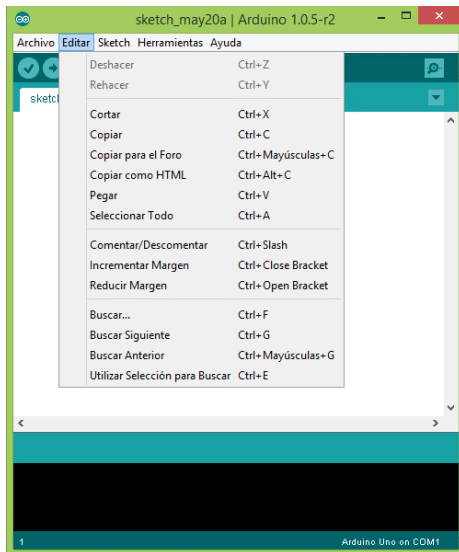


Figura 6: Pestaña archivo

# Pestaña editar





# Pestaña sketch

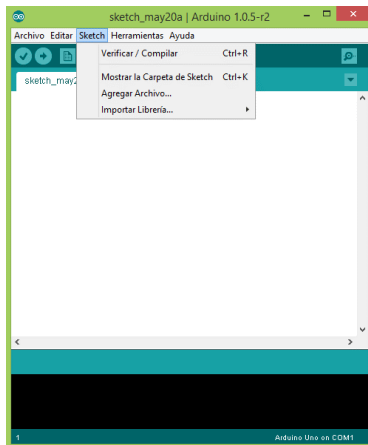


Figura 8: Pestaña sketch

# Pestaña herramientas

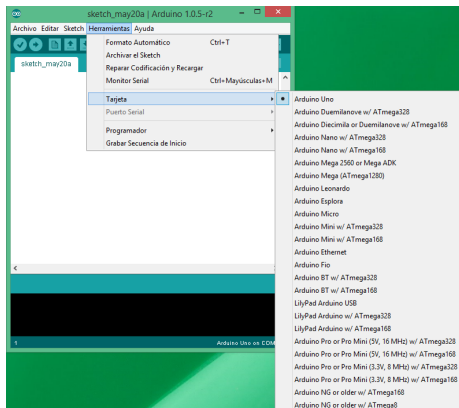


Figura 9: Pestaña herramientas - *Tarjeta*

# Pestaña herramientas

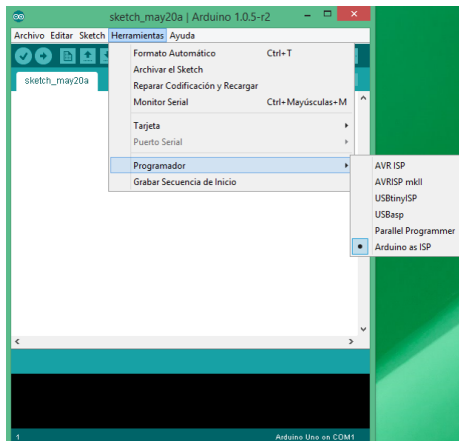
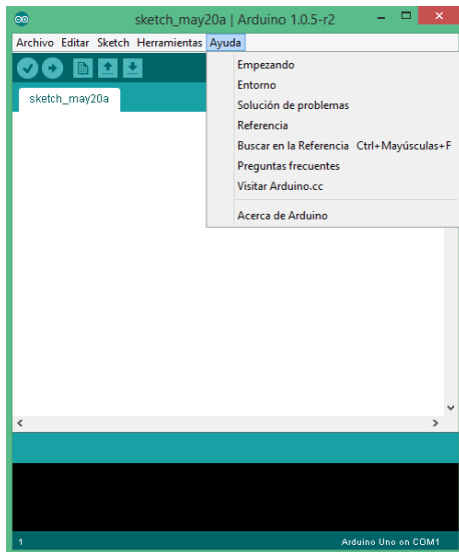


Figura 10: Pestaña herramientas - *Programador*

# Pestaña ayuda



# Estructura de un código

```
/*  
Encabezado  
Autor  
Año  
Descripción  
Derechos de CopyRight  
*/  
#include<Librería.h> //Inclusión de librerías  
int variable; //Declaración de variables  
void setup() { //Función void setup()  
}  
void loop() { // Función void loop()  
}  
/*Es muy importante colocar al final  
de cada línea punto y coma (;)  
*/
```

Figura 12: Estructura de un código



## Operador Lógico Y (&&)

Se usan para especificar una condición.

```
if(A<54 && B>54){ //¿A es menor de 54 y
    Camino A      //B mayor que 54?
}
/*Si se cumple la condición
toma el camino A
*/
```

Figura 13: Operador lógico Y (&&)

# Operador Lógico O (||)

```
if(A<54 || B>54){ //¿A es menor de 54 o
    Camino A      //B mayor que 54?
}
/*Si se cumple una de las
dos toma el camino A
*/
```

Figura 14: Operador Lógico O (||)

# Adición, sustracción, multiplicación, división, módulo, igualdad

## Los operadores matemáticos son:

- Adición +
- Sustracción -
- Multiplicación \*
- División /
- Módulo %
- Igualdad =





## Operador De Igualdad (==)

Son usados principalmente como parte de condicionales.

```
if(A==B){ //¿A es igual a B?  
    Camino A  
}
```

Figura 15: Operador De Igualdad (==)

# Operador De Desigualdad ( $\neq$ )

```
if(A!=B){ //¿A es diferente a B?  
    Camino A  
}
```

Figura 16: Operador De Desigualdad ( $\neq$ )

## Mayor O Menor (<>)

```
if(A>B){      //¿A es mayor a B?  
    Camino A  
}  
  
if(A<B){      //¿A es menor a B?  
    Camino A  
}
```

Figura 17: Mayor O Menor (<>)

## Menor, Mayor O Igual ( $\leq$ o $\geq$ )

```
if(A>=B){    //¿A es mayor o igual a B?
    Camino A
}

if(A<=B){    //¿A es menor o igual a B?
    Camino A
}
```

Figura 18: Menor, Mayor O Igual ( $\leq$  o  $\geq$ )

# Variables

Declarar una variable es simplemente asignar un valor. Se hace de la siguiente manera.

## Variable int

```
int led = 13;
int variable;
/* Usamos int para
declarar variables
enteras
Toma valores entre
-32,768 Y 32,767
*/
```

Figura 19: Variable int



## Variable float

```
float variable;  
/* Usamos float para  
declarar variables de  
números en coma flotante  
(Número decimal)  
Toma valores entre  
-3.4028235E+38 y  
3.4028235E+38  
*/
```

Figura 20: Variable float



## Variable long

```
long variable;  
/* Usamos long para  
declarar variables de  
números grandes.  
Toma valores entre  
-2,147,483,648 a  
2,147,483,647.  
*/
```

Figura 21: Variable long



## Variable double

```
double variable;  
/*Se usa para números  
con punto decimal.  
Puede tomar valores entre  
1.7E-308 y 1.7E+308.  
*/
```

Figura 22: Variable double



# Variable String

```
String variable;  
/* No permite guardar valores  
alfanuméricos. Toma valores desde  
0 hasta  $2 \cdot 10^9$  caracteres  
*/
```

Figura 23: Variable String

# Variable char

```
char variable;  
/*  
char es un carácter, por  
lo tanto, el contenido  
de una variable char debe  
ser un carácter  
*/
```

Figura 24: Variable char

# Condicionales

Las estructuras condicionales comparan una variable contra otro valor. Se usan de la siguiente manera.

## Condicionales If / Else

```
if(Condición 1){  
    Camino A  
}  
else(){  
}  
/* Si la condición 1 se  
cumple, entonces tomaremos  
el camino A, pero si no  
se cumple toma el camino B.
```

Figura 25: Condicional If / Else



## Ciclo for

```
for(int i=0; i<200; i++){  
}  
/* Éste ciclo se repite  
las veces que deseemos.  
*/
```

Figura 26: Condicional for

Está estructurado de a siguiente manera:

*for(condición de inicio; condición de parada; aumento)*



## Ciclo while

```
while(numero<100){  
}  
/*Es un ciclo muy útil,  
similarmente al for  
permite ejecutar una orden  
mientras se cumpla la  
condición*/
```

Figura 27: Condicional while

Su estructura consiste en:

*while(condición)*

# Switch

```
switch(Variable){ //Variable
  case 1:
    Tomar camino A //Si coincide, toma este camino.
    break
  case 2:
    Tomar camino B //Si coincide, toma este camino.
    break
}
/*Permitir especificar un código distinto que
debe ser ejecutado en varias condiciones.
En particular, una sentencia switch compara
el valor de una variable a los valores especificados
en las sentencias case.
Cuando se encuentra una sentencia case cuyo valor
coincide con el de la variable, se ejecuta el
código en esa sentencia case.
*/
```

Figura 28: Switch

## ¿Qué es una función?

Una función es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor. Están compuestas de la siguiente manera:

*variable de salida - nombre de la función (variable de entrada) )*

Es posible usar la función void como parámetro de salida puesto que es una función que no retorna datos.

Es característico de una función que no hay espacio entre los nombres de funciones. Para llamar una función basta con escribir el nombre de la función seguido de paréntesis.

*Nombredelafunción()*



## Función void setup()

La función *void setup()* siempre debe ser invocada, se ejecuta una sola vez, cuando comienza a correr el código o si reiniciamos nuestro Arduino. Esta se utiliza para inicializar y configurar puertos, variables y comunicaciones.

```
void setup() {  
}
```

Figura 29: Función void setup()





## Función void loop()

Esta función tiene la característica que siempre está iterando. Dentro de ésta se editará el código que iterará de manera infinita.

```
void loop() {  
  }  
}
```

Figura 30: Función void loop()

# Referencias

-  <http://www.arduino.cc>
-  <http://arduino.cc/en/Reference/HomePage>