

# MÓDULO 4: ECOSISTEMA “MAKER”

---

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS



# Entorno de Programación

Arduino IDE es el entorno de desarrollo oficial para Arduino.

Puede descargarse libremente desde:

- <https://www.arduino.cc/en/Main/Software>

Permite:

- Desarrollo de código
- Conexión con Arduino
  - Compilación y envío a Arduino
  - Monitor Serie
- Descarga librerías



# LENGUAJE C++

---

El lenguaje de programación de Arduino es C++.

- No es un C++ puro sino que es una adaptación para su uso con los microcontroladores AVR de Atmel y cuenta con muchas utilidades específicas.

Principales características:

- Lenguaje de programación de propósito general.
- Lenguajes híbrido: desde orientación a objetos, hasta manejo de bits y direcciones de memoria.
- Posee una gran portabilidad

# LENGUAJE C++: SINTAXIS

---

Principales reglas de sintaxis:

- **{}** entre llaves
  - Las llaves sirven para definir el principio y el final de un bloque de instrucciones. Se utilizan para los bloques de programación como `setup()`, `loop()`, `if`, `for`, `while`, etc.
- **;** punto y coma
  - Se utiliza para separar instrucciones. También se utiliza para separar elementos en una instrucción de tipo “bucle for”.

# LENGUAJE C++: SINTAXIS

---

- `/*...*/` bloque de comentarios
  - Los bloques de comentarios, o comentarios multi-línea son áreas de texto ignorados por el programa que se utilizan para las descripciones del código o comentarios que ayudan a comprender el programa.
- `//` línea de comentarios
  - Una línea de comentario empieza con `//` y terminan con la siguiente línea de código.

# LENGUAJE C++: GUÍA DE ESTILO

---

## Recomendaciones:

- Documentar al máximo
- Usar esquemas
- Predominar la facilidad de lectura sobre la eficiencia del código
- Poner el *setup()* y *loop()* al principio del programa
- Usar variables descriptivas
- Explicar el código al principio
- Usar indentación

# ESTRUCTURA BÁSICA DE UN SKETCH

---

Estructura básica de un sketch de Arduino:

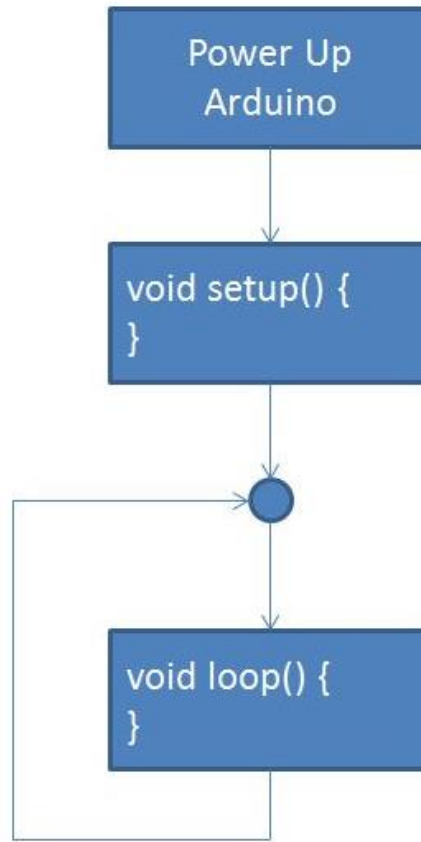
```
1  void setup() {  
2    // put your setup code here, to run once:  
3  
4  }  
5  
6  void loop() {  
7    // put your main code here, to run repeatedly:  
8  
9  }
```

- No es necesario que un sketch esté en un único fichero, pero si es imprescindible que todos los ficheros estén dentro del mismo directorio que el fichero principal.
- No es necesario que el fichero principal (el que tiene el mismo nombre que el directorio que lo contiene) tenga obligatoriamente las funciones *setup()* y *loop()*.

# ESTRUCTURA BÁSICA DE UN SKETCH

---

Diagrama de flujo:





# TIPOS DE DATOS

---

Un tipo de dato informático es un atributo de los datos que indica al ordenador (y/o al programador) sobre la clase de datos que se va a trabajar.

Esto incluye imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar.

- Tipos de datos comunes son: números enteros, números con signo (negativos), números de coma flotante (decimales), cadenas alfanuméricas, estados (booleano), etc.

# TIPOS DE DATOS

---

Data Types	Size in Bytes	Can contain:
boolean	1	true (1) or false (0)
char	1	ASCII character or signed value between -128 and 127
unsigned char, byte, uint8_t	1	ASCII character or unsigned value between 0 and 255
int, short	2	signed value between -32,768 and 32,767
unsigned int, word, uint16_t	2	unsigned value between 0 and 65,535
long	4	signed value between -2,147,483,648 and 2,147,483,647
unsigned long, uint32_t	4	unsigned value between 0 and 4,294,967,295
float, double	4	floating point value between -3.4028235E+38 and 3.4028235E+38 (Note that double is the same as a float on this platform.)

# ALGEBRA DE BOOLE

Es una estructura algebraica que esquematiza las operaciones lógicas.

Permite la realización de comparaciones en los programas de ordenador.

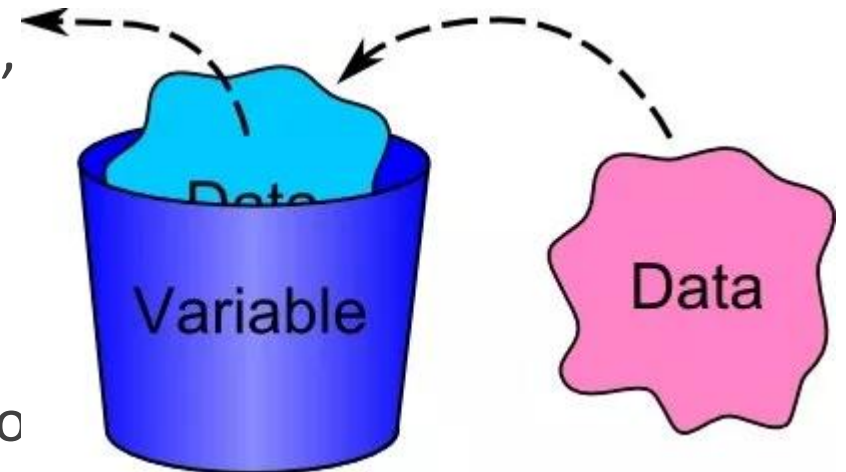
También puede usarse a nivel electrónico.

La operación AND o Y		
$0 \cdot 0 = 0$		$0 \cdot 0 = 0$
$0 \cdot 1 = 0$		$0 \cdot A = 0$
$1 \cdot 0 = 0$		$A \cdot 0 = 0$
$1 \cdot 1 = 1$		$A \cdot A = A$
La operación OR o O		
$0 + 0 = 0$		$A + 0 = A$
$0 + 1 = 1$		$A + 1 = 1$
$1 + 0 = 1$		$A + A = A$
$1 + 1 = 1$		$A + A = 1$
La operación NOT o No		
$\overline{0} = 1$		$A'' = A$
$\overline{1} = 0$		<b>Nota:</b> $A' = \overline{A}$

# VARIABLES DE UN PROGRAMA

Una variable es un lugar donde almacenar un dato, tiene un nombre, un valor y un tipo.

- **IMPORTANTE:** en C++ todas las variables tienen que ser declaradas antes de su uso.
  - Sólo se declara una vez.
- Una variable tiene un nombre, un valor y un tipo
  - Al declarar una variable se debe indicar primero el tipo de variable y luego su nombre, opcionalmente se le puede dar un valor.



Ejemplos de declaración:

- `int i,j;`
- `float pi=3.1416;`
- `float x,pi;`
- `unsigned long contador=0;`

# VARIABLES DE UN PROGRAMA

---

## AMBITO DE UNA VARIABLE

Una variable puede ser declarada en múltiples sitios: al inicio del programa antes *setup()*; a nivel local dentro de las funciones; dentro de un bloque, como por ejemplo *if*, *for*, etc.

En función del lugar de declaración de la variable así se determinará el ámbito de aplicación:

- **Variable global:** es aquella que puede ser vista y utilizada por cualquier función y estamento de un programa.
- **Variable local:** es aquella que se define dentro de una función o como parte de un bucle. Sólo es visible y sólo puede utilizarse dentro de la función en la que se declaró.

# VARIABLES DE UN PROGRAMA

---

## AMBITO DE UNA VARIABLE

```
int x=10;      // Global x
void main()
{
  ① [ int x=20;      // X Local to Block 1
    -- --
    -- --
    ② [ {
      int x=30; // X Local to Block 2
      -- --
      -- --
      -- --
    }
  ]
}

void funct()
{
  ③ [ int x=40;      // X Local to Block 3
    -- --
    -- --
    -- --
  ]
}
```

# VARIABLES DE UN PROGRAMA

---

## NORMAS DE NOMBRES

Deben empezar por una letra O ‘\_’ (esto último no siempre recomendable).

- Los nombres de variables pueden tener letras, números y el símbolo ‘\_’.
- ¡Minúsculas y mayúsculas no son iguales!
- Las palabras reservadas (*if*, *else*, *for*, etc.) no pueden usarse como nombres.

Recomendación: Usa las mismas reglas dentro del código para el nombramiento de variables (primera letra palabra en mayúscula, separación con ‘\_’, etc.)

# CONSTANTES DE UN PROGRAMA

---

Una constante es un valor que no puede ser alterado/modificado durante la ejecución de un programa, únicamente puede ser leído.

Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos. Por ejemplo el valor de  $\pi = 3.1416$ .

Las constantes pueden ser definidas a nivel de módulo antes de compilar, de forma que no ocupan memoria:

- Se usa la palabra clave *#define* → Ej. `#define MOTOR 2` (¡sin punto y coma!)

Constantes predefinidas Arduino: <http://arduino.cc/en/Reference/Constants>



# OPERADORES

---

## ARITMÉTICOS

Realizan operaciones aritméticas básicas:

- Asignación: = (<http://arduino.cc/en/Reference/Assignment>)
- Operaciones: +, -, \*, / (<http://arduino.cc/en/Reference/Arithmetic>)
- Módulo: % (<http://arduino.cc/en/Reference/Modulo>)

## COMPUESTOS

Combinan una operación aritmética con una variable asignada

- ++, -- (<http://arduino.cc/en/Reference/Increment>)
- +=, -=, \*=, /= (<http://arduino.cc/en/Reference/IncrementCompound>)

# OPERADORES

---

## COMPARADORES

Comprueban si una condición es verdadera. Se utilizan en las estructura *if*, *while*, etc.

- **==, !=, <, >, <=, >=** (<http://arduino.cc/en/Reference/If>)

## BOOLEANOS

Comparan dos expresiones y devuelve un VERDADERO o FALSO dependiendo del operador:

- **AND (&&), OR (||)** y **NOT (!)**

# ESTRUCTURAS DE CONTROL

---

## IF()

Es un estamento que se utiliza para probar si una determinada condición se ha alcanzado. Su estructura es:

```
If (condición_1){  
    instrucción;  
}  
if... else (condición_2){  
    instrucciones;  
}  
else{  
    instrucciones  
}
```

# ESTRUCTURAS DE CONTROL

---

## FOR()

Se usa para repetir un bloque de sentencias encerradas entre llaves un número determinado de veces. Su estructura es:

```
for (inicialización; condición; incremento) {  
    instrucciones;  
}
```

- Ejemplo:

```
for(int i=0; i<10; i++){  
    Serial.print("Repetición número "); Serial.println(i+1)  
}
```

# ESTRUCTURAS DE CONTROL

---

## WHILE()

Bucle de ejecución continua mientras se cumpla su condición. Su estructura es:

```
while(condición){  
    instrucción;  
}
```

- Ejemplo:

```
while(Interruptor==HIGH){  
    Serial.print("Repetición número "); Serial.println(i+1)  
    i++;  
    delay(1000);  
}
```

# COMUNICACIÓN CON EL USUARIO

---

## SERIAL.PRINT() Y SERIAL.PRINTLN()

Imprime texto a través del monitor Serial en un formato legible por el ser humano (texto ASCII).

- *Serial.print()*: imprime el texto o variable sin salto de línea al final.
- *Serial.println()*: ídem, incluyendo un salto de línea al final.