Variables

Tipos de datos y constantes de Arduino.

Constantes

Las constantes son expresiones predefinidas en el lenguaje Arduino. Se utilizan para facilitar la lectura de los programas.

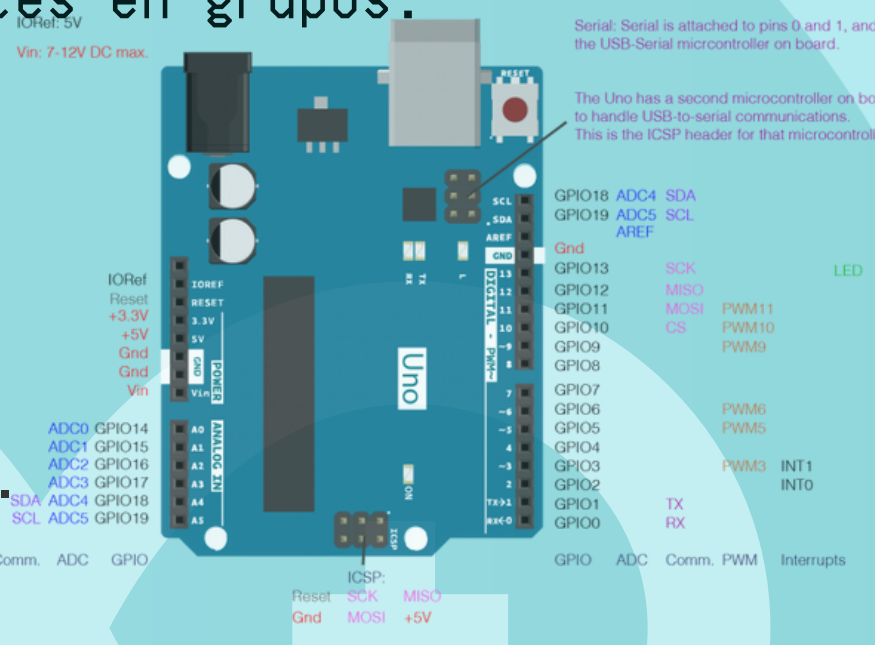
Clasificamos las constantes en grupos:

true, false.

HIGH,LOW.

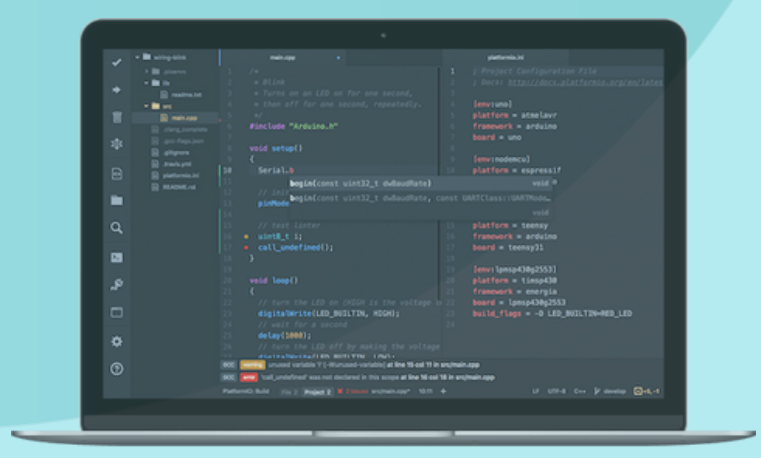
INPUT,INPUT_PULLUP,OUTPUT.

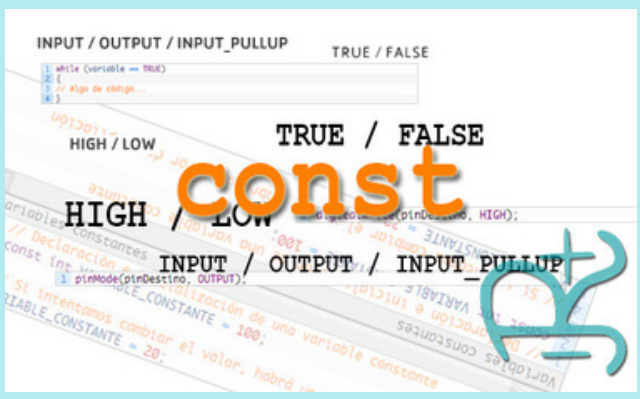
LED_BUILTIN.

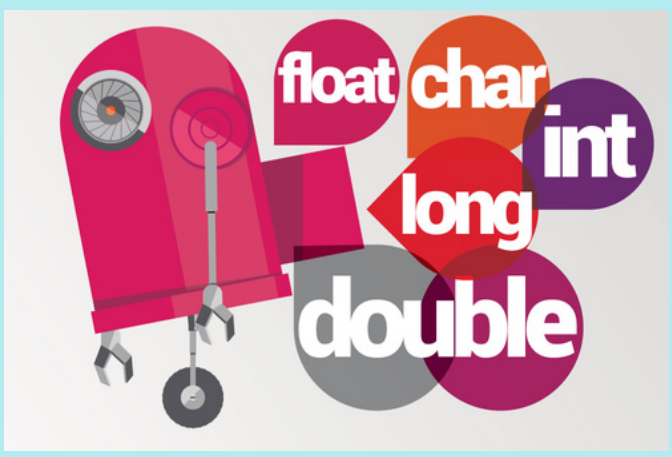


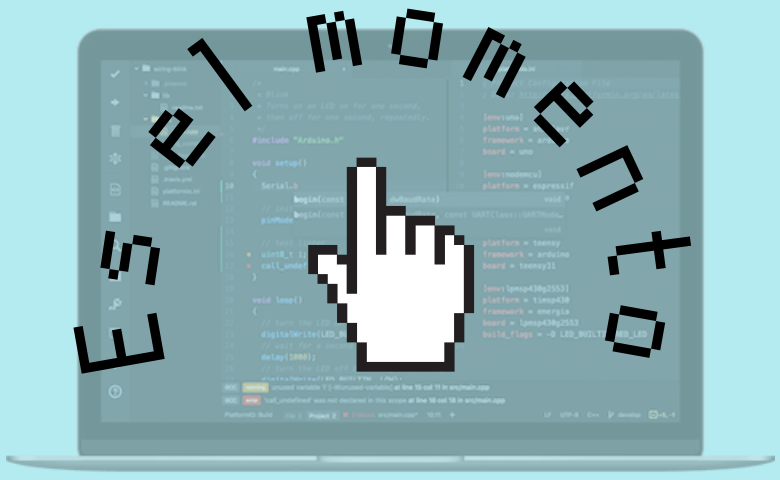
Tipos de datos

array
bool
boolean
byte
char
double
float
int
string
String()
void









Constantes

True y False

Hay dos constantes que se utilizan para representar la verdad y la falsedad en el lenguaje Arduino: true, y false.

False
false - Es el más fácil de los dos para definir. falso se define como 0 (cero).

True
true a menudo se dice que se define como 1.
Tenga en cuenta que las constantes true y false se escriben en minusculas a diferencia HIGH, LOW, INPUTy OUTPUT.

HIGH y LOW

Al leer o escribir en un pin digital, solo hay dos valores posibles que un pin puede tomar o configurar: HIGHy LOW.

HIGH
El significado de HIGH (en referencia a un pin) es algo diferente dependiendo de si un pin está configurado como INPUT o OUTPUT.

LOW
El significado de LOW también tiene un significado diferente dependiendo de si un pin está configurado en INPUT o OUTPUT.

INPUT, INPUT_PULLUP , OUTPUT y LED_BUILTIN.

Los pines digitales se pueden usar como INPUT, INPUT_PULLUPo OUTPUT. Cambiar un pin pinMode()cambia el comportamiento eléctrico del pin.

Pines configurados como INPUT

Si tiene su pin configurado como INPUT y está leyendo un interruptor, cuando el interruptor está en estado abierto, el pin de entrada estará "flotando", lo que dará como resultado resultados impredecibles.

Pines configurados como INPUT_PULLUP

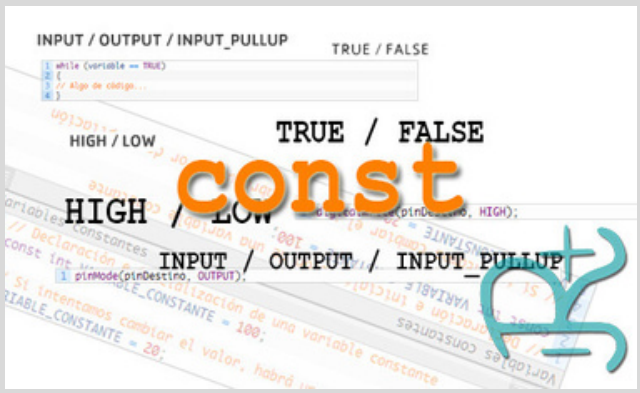
El microcontrolador ATmega en el Arduino tiene resistencias pull-up internas (resistencias que se conectan internamente a la alimentación) a las que puede acceder.

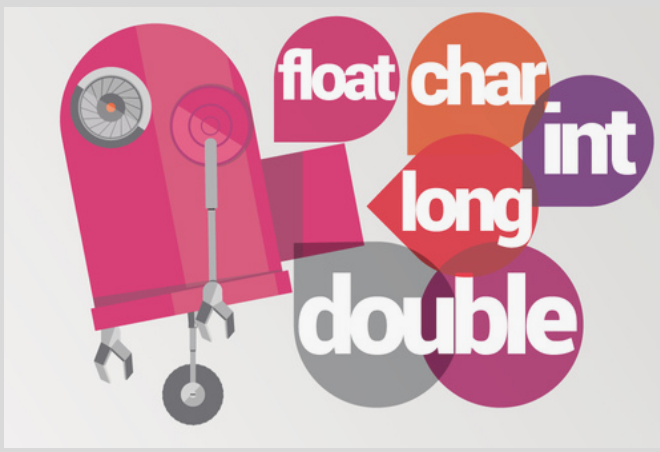
Pines configurados como OUTPUT

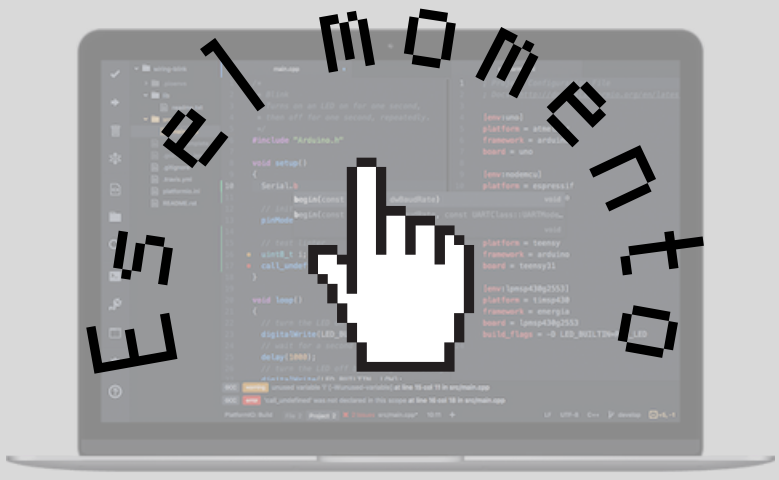
Se dice que los pines configurados OUTPUT con pinMode()están en un estado de baja impedancia, Esto significa que pueden proporcionar una cantidad sustancial de corriente a otros circuitos.

Definición de incorporados: LED_BUILTIN

La mayoría de las placas Arduino tienen un pin conectado a un LED integrado en serie con una resistencia. La constante LED_BUILTIN es el número del pin al que está conectado el LED integrado. La mayoría de las placas tienen este LED conectado al pin digital 13.







Tipos de datos

Array

Es una colección de variables a las que se accede con un número de índice.

Crear (declarar) una matriZ.

Todos los métodos a continuación son formas válidas de crear (declarar) una matriz.

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[5] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

Acceso a una matriz

Las matrices están indexadas a cero, es decir, en referencia a la inicialización de la matriz anterior, el primer elemento de la matriz está en el índice 0, por lo tanto

```
mySensVals[0] == 2, mySensVals[1] == 4,Etcétera.
```

Para asignar un valor a una matriz:

```
mySensVals[0] = 10;
```

Para recuperar un valor de una matriz:

```
x = mySensVals[4];
```

Bool

Descripción

A bool contiene uno de dos valores, true o false. (Cada boolvariable ocupa un byte de memoria).

Sintaxis

bool var = val;

Parámetros

var: Nombre de la variable.

val: El valor a asignar a esa variable.

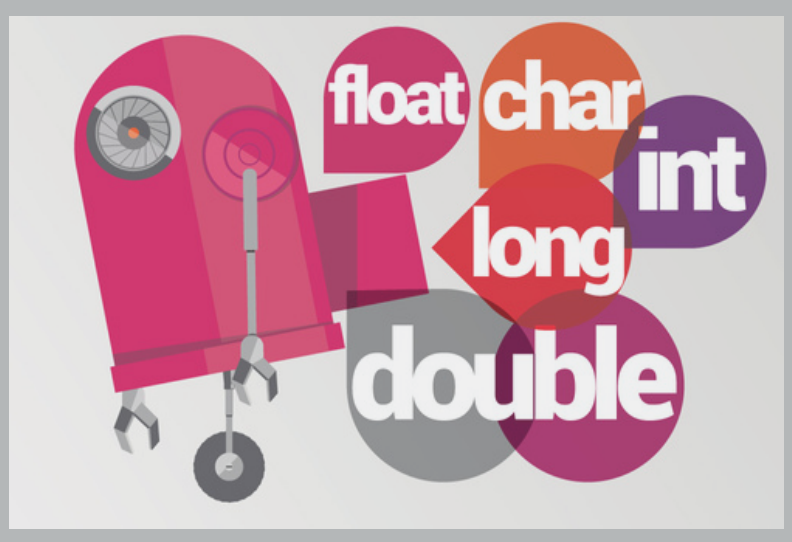
Código de ejemplo

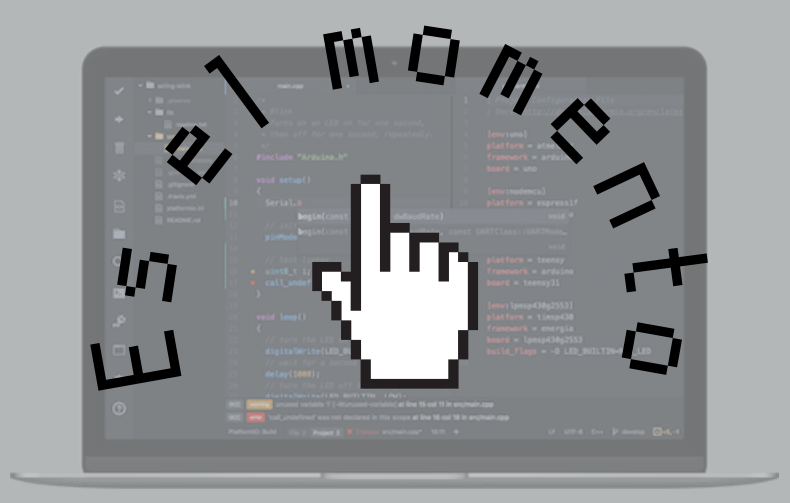
```
int LEDpin = 5;     // LED en pin 5
int switchPin = 13; // Interruptor momentáneo en 13, otro lado conectado a tierra

bool running = false;

void setup() {
  pinMode(LEDpin, OUTPUT);
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH);  // Encienda la resistencia pull-up
}
void loop() {
  if (digitalRead(switchPin) == LOW) {
    // el interruptor está presionado - pullup mantiene el pin alto normalmente
    delay(100);                     // Retardo para interruptor de rebote
    running = !running;             // Alternar la variable en ejecución
    digitalWrite(LEDpin, running);  // indicar mediante LED
  }
}
```

Boolean

Descripción

boolean es un alias de tipo no estándar bool definido por Arduino. En su lugar, se recomienda utilizar el tipo estándar bool, que es idéntico.





Tipos de datos

Char

Descripción

Un tipo de datos utilizado para almacenar un valor de carácter. Los caracteres literales se escriben entre comillas simples, así: 'A' (para varios caracteres, cadenas, use comillas dobles: "ABC").

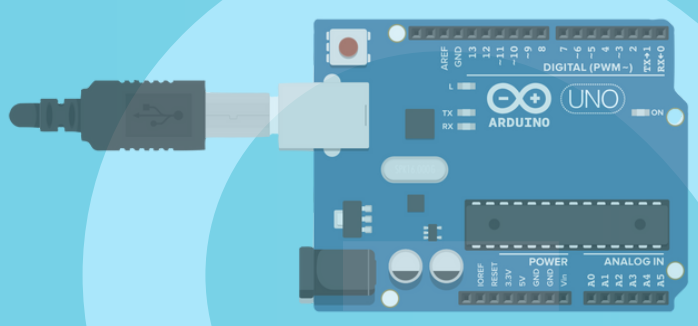
Sintaxis

char var = val;

Parámetros

var: Nombre de la variable.
val: El valor a asignar a esa variable

```
char myChar = 'A';
char myChar = 65; // Ambos son equivalentes
```



Byte

Descripción

Un byte almacena un número sin signo de 8 bits, de 0 a 255.

Sintaxis

byte var = val;

Parámetros

var: Nombre de la variable.
val: El valor a asignar a esa variable.

Double

Descripción

Número de coma flotante de doble precisión. En Uno y otras placas basadas en ATMEGA, ocupa 4 bytes. Es decir, la implementación doble es exactamente igual que la flotante, sin ganar en precisión.

Sintaxis

double var = val;

Parámetros

var: Nombre de la variable.

val: El valor a asignar a esa variable.

Float

Descripción

Tipo de datos para números de punto flotante, un número que tiene un punto decimal. Los números de coma flotante a menudo se usan para aproximar valores analógicos y continuos porque tienen una resolución mayor que los números enteros.

Sintaxis

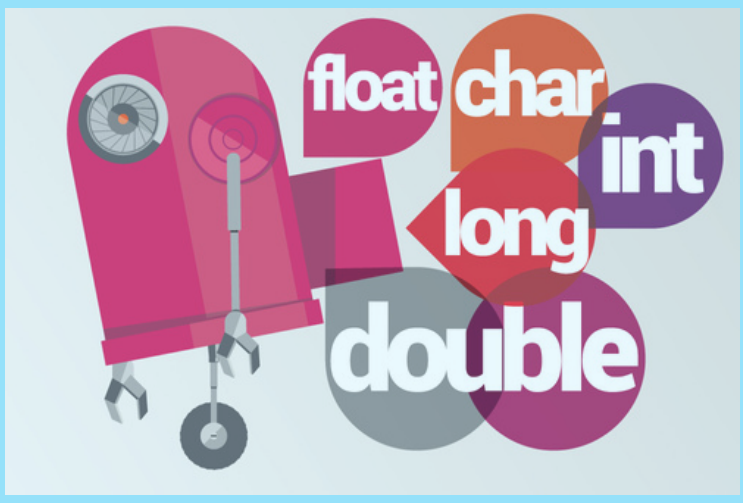
float var = val;

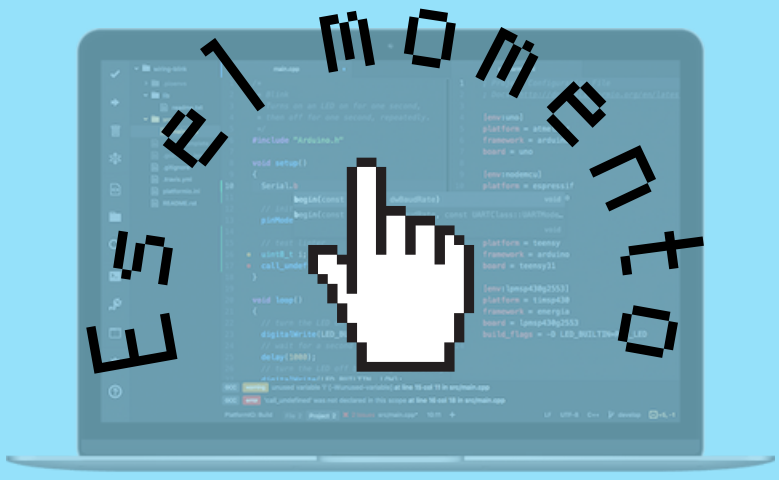
Parámetros

var: Nombre de la variable.
val: El valor que asignas a esa variable.

Código de ejemplo

```
float myfloat;
float sensorCalbrate = 1.117;

int x;
int y;
float z;

x = 1;
y = x / 2;          // y ahora contiene 0, ints no puede contener fracciones.
z = (float)x / 2.0; // z ahora contiene .5 (tienes que usar 2.0, no 2).
```





Tipos de datos

Int

Descripción

Los números enteros son su tipo de datos principal para el almacenamiento de números.

Sintaxis

```
int var = val;
```

Parámetros

var: Nombre de la variable.
val: El valor que asignas a esa variable.

Código de ejemplo

```
int countUp = 0;             //Crea un entero variable llamado 'countUp'

void setup() {
  Serial.begin(9600);        // Utilice el puerto serie para imprimir el número
}

void loop() {
  countUp++;                 //Asuma 1 al int countUp en cada ciclo
  Serial.println(countUp);   // Imprime el estado actual de countUp
  delay(1000);
}
```

String

Descripción

Las cadenas de texto se pueden representar de dos maneras. puede usar el tipo de datos String, que es parte del núcleo a partir de la versión 0019, o puede hacer una cadena a partir de una matriz de tipo char y terminarla en cero.

Sintaxis

Todas las siguientes son declaraciones válidas para cadenas.

```
char Str1[15];
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[] = "arduino";
char Str5[8] = "arduino";
char Str6[15] = "arduino";
```

Código de ejemplo

```
char *myStrings[] = {"This is string 1", "This is string 2", "This is string 3",
                     "This is string 4", "This is string 5", "This is string 6"
                    };

void setup() {
  Serial.begin(9600);
}

void loop() {
  for (int i = 0; i < 6; i++) {
    Serial.println(myStrings[i]);
    delay(500);
  }
}
```

String()

Descripción

Construye una instancia de la clase String. Hay varias versiones que construyen cadenas a partir de diferentes tipos de datos (es decir, les dan formato como secuencias de caracteres).

Sintaxis

```
String(val)
String(val, base)
String(val, decimalPlaces)
```

Parámetros

val: una variable para formatear como una cadena.

base: (opcional) la base en la que dar formato a un valor integral.

decimalPlaces: solo si val es float o double . Los lugares decimales deseados.

Código de ejemplo

```
String stringOne = "Hello String";                    // Usando una cadena constante
String stringOne = String('a');                       // Convertir un carácter constante en una cadena
String stringTwo = String("This is a string");        // Convertir una cadena constante en un objeto String
String stringOne = String(stringTwo + " with more"); // Concatenar dos cadenas
String stringOne = String(13);                        // Utilizando un entero constante
String stringOne = String(analogRead(0), DEC);        // Usando un int y una base
String stringOne = String(45, HEX);                   // Usando un int y una base (hexadecimal)
String stringOne = String(255, BIN);                  // Usando un int y una base (binario)
String stringOne = String(millis(), DEC);             // Usando una larga y una base
String stringOne = String(5.698, 3);                  // Usando un flotador y los lugares decimales
```