Elementos que debemos conocer

Antes de comenzar a programar, es importante que conozcamos una serie de elementos que nos van a acompañar durante este curso. Ahora puede abrumarnos y parecernos mucha información, pero este apartado y el siguiente que trata sobre estructuras está pensado para que puedas volver a consultarlo a lo largo del curso.

1. Comentarios...

En algunos de nuestros algoritmos/programas, puede que se vea muy claro qué hace cada parte, pero en otras ocasiones no es tan fácil saberlo. Por ello, los comentarios van a ayudar a que otros programadores que tengan acceso a nuestro código puedan comprenderlo más fácilmente, e incluso nosotros mismos. Imagina que has escrito un algoritmo en 2007, te olvidas de él y lo vuelves a abrir en 2032. Probablemente, en un primer momento no sepas de qué va, ni para qué lo escribiste, pero si en 2007 escribiste comentarios, te habrá ahorrado mucho tiempo al abrirlo 25 años después.

Estos comentarios contienen **información dirigida a las personas**, son notas que tomaréis para vosotros mismos relacionadas con el funcionamiento del programa que estéis construyendo, o notas para facilitar a otras personas la comprensión del algoritmo que habéis creado. Son muy útiles para escribir **recordatorios o explicaciones** durante el proceso de trabajo y para **etiquetar** las diferentes partes de las estructuras que hagáis.

...en Arduino

Vamos a echarle un vistazo a la siguiente imagen:

Copyright 2026

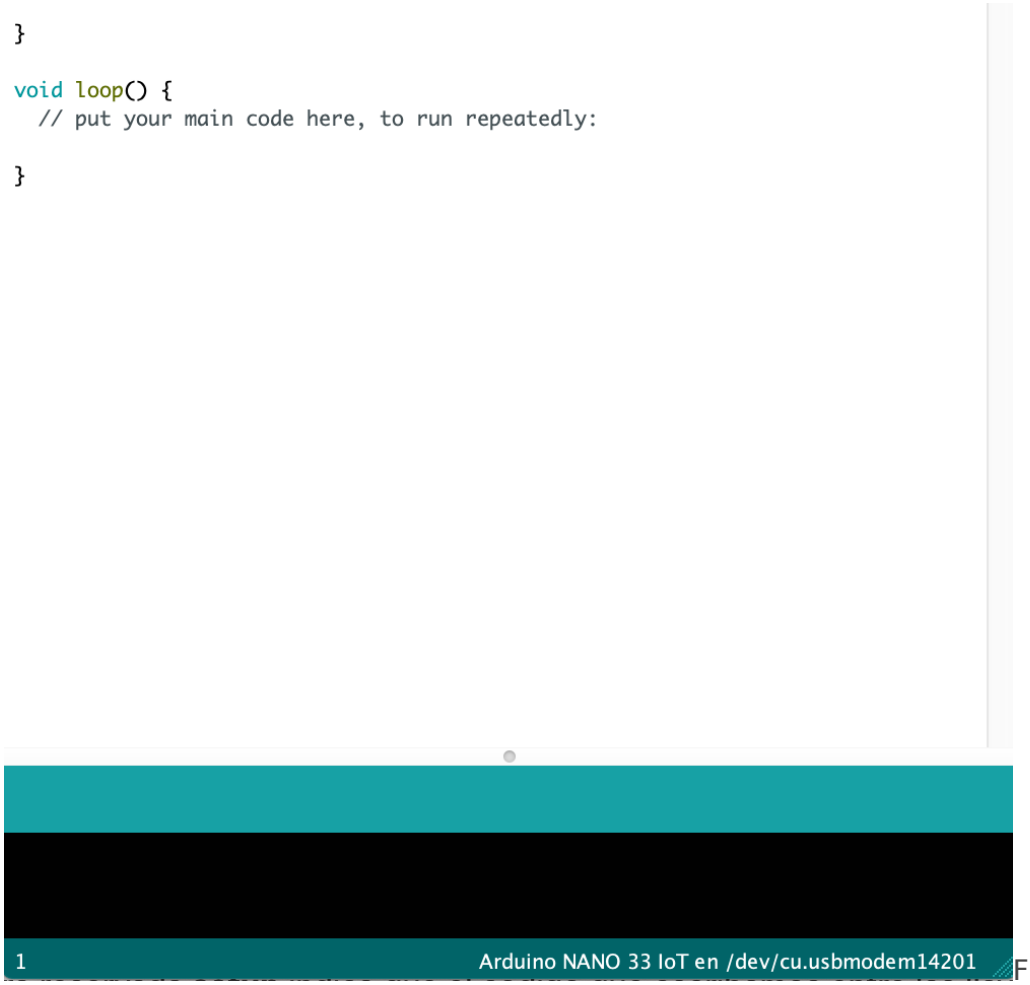


Figura 1. Imagen de un sketch de Arduino

Esto es lo primero que veremos cuando tengamos instalado Arduino. Por defecto, al abrir la aplicación aparecen ciertas palabras escritas. En azul y verde vemos **void setup()** y **void loop().** De ambas nos ocuparemos en el siguiente apartado sobre funciones.

Ahora, lo que nos interesa son las líneas que aparecen // entre los corchetes de ambas funciones. ¿Qué es importante aquí? Que esas indicaciones no van destinadas a nuestro ordenador, sino al programador. Se trata de **comentarios.**

Cuando nuestros comentarios solamente ocupen una línea los escribiremos tras //. Como puedes comprobar, su color es **gris claro**, eso nos indica que **Arduino hará caso omiso de ellos.** Habrá momentos en los que necesitemos escribir comentarios más extensos y para ellos los escribiremos

Copyright 2026

entre /**/.

Ejemplo: **/*Esto es un comentario que ocupa más de una línea. */**

...en Pure Data

En Pure data vamos a tener cajitas para las distintas funciones con entradas y salidas para los datos. Los comentarios visualmente no aparecen en ninguna cajita ni tienen entradas/inputs ni salidas/outputs. El texto introducido en los comentarios es información que no sera interpretada por la maquina como elementos transformadores del programa/patch que estamos construyendo.



Figura 2. Un comentario en Pure data cuando esta recién creado.

2. Funciones...

Una **función** es un fragmento de código, un **subalgoritmo** dentro de nuestro algoritmo, que tiene como propósito resolver una tarea determinada.

Por ejemplo, si nuestro **algoritmo** es hacer un **bizcocho**, que consta de una serie de **pasos** como: comprar los ingredientes, batir los huevos, añadir el azucar, etc. Una **función** podría ser la que se encarga del paso **'batir los huevos'** y la **descompondríamos** en : 1. coger los huevos, 2.romperlos y echarlos en un plato, 3. coger un tenedor, 4. batirlos.

... en Arduino

Como hemos visto en el apartado anterior, a parte de los comentarios había unas palabras a color: **void setup()** y **void loop()**

Bueno, la estructura sería esta.

```
void setup() {
// aquí escribimos los comandos a ejecutar
}
```

> Es muy importante que no olvidemos ningún**paréntesis** ni **corchete**, o el programa no funcionará.

Ambas son **funciones**. Y, ¿qué es una función dentro del mundo de Arduino? Como ya hemos dicho, una función es un fragmento de código, un **subalgoritmo** dentro de nuestro algoritmo, que tiene como propósito resolver una tarea específica.

La palabra **void**, significa que esta función no nos devuelve ningún valor (sea un número, un caracter, un string, etc.) y la palabra reservada **setup** indica que el código que escribamos entre las llaves **{}** se ejecutará **una sola vez** al inicio del programa. Tanto setup como loop son **palabras reservadas**, pero eso lo veremos un poco más abajo.

En Arduino existen muchas funciones ya programadas, pero a parte de eso, podemos programar nuestras propias funciones.

> Tienes una lista completa con las funciones que podemos encontrar en Arduino, aquí.

... en Pure Data

En pure data las funciones serán los **objetos**, que son cajitas en las que escribiremos el nombre de la función. Cada objeto va a realizar una función específica. La cajita con "+13" que vemos debajo cumple la función de sumar 13 a cualquier numero que entre en la cajita.



Una metáfora de una función que ya hemos visto en la Práctica 0: Programaci... | Librería CATEDU sería la cajita exprimidor.

https://giphy.com/embed/3o7TKPdUkkbCAVqWk0

Figura 3. La cajita exprimidor.

3. Variables y palabras reservadas

... en Arduino

No podemos avanzar mucho más sin hablar de ellas, las **variables**. Ellas nos van a permitir almacenar valores en nuestros programas, para ello debemos que saber que podemos almacenar datos de distintos tipos.

Palabras como **int, for, void loop()** o **if** se consideran **palabras reservadas** porque poseen un significado especial dentro del lenguaje de programación que estamos empleando. Una manera de identificar a estas palabras especiales es porque **aparecen con un color diferente**.

Palabras reservadas para tipos de datos

Estas palabras tienen diferentes objetivos. Uno de ellos es para denotar los tipos de datos que podemos usar en nuestros programas. A estas palabras se les conoce como **tipos predefinidos**, las escribimos delante de aquellas variables que vamos a emplear en nuestro programa y algunas de ellas son:

int: almacena un número entero de un tamaño máximo de 16 bits.
float: almacena un número decimal de 32 bits.
char: almacena un caracter.
boolean: almacena el valor verdadero (TRUE) o falso (FALSE).
long: almacena el valor de un número entero de 32 bits.

Por ejemplo podemos crear una variable que almacene el valor numérico que obtengamos con un sensor. Podríamos crearla así:

```
int valorSensorTemperatura = 0;
```

Lo que haría que **el valor de esa variable inicialmente sea cero**, siendo modificado cuando lo asignemos al pin de nuestro Arduino al que hemos conectado el sensor. Como vemos, la palabra reservada sería **int,** mientras que **valorSensorTemperatura** es el nombre de la variable. Ese nombre lo usaremos en nuestro programa cada vez que queramos usar o modificar su valor.

Copyright 2026

Palabras reservadas para funciones

Como ya hemos visto, existe una serie de funciones que ya vienen preprogramadas y que podemos utilizar directamente. Las dos fundamentales son **void setup()** y **void loop()**, pero existen muchas otras.

Algunas de ellas son:
delay(): detiene el programa durante el tiempo indicado (en milisegundos).
random(): genera números pseudoaleatorios.
float(): convierte un número, por ejemplo del tipo int, a un número decimal.

... en Pure Data

Las palabras reservadas en Pure Data serán aquellas que generalmente vamos a escribir dentro de una cajita de tipo objeto y que van **determinar la función de esa cajita**. Como por ejemplo la palabra "metro" que crea un objeto que envía señales periódicas cada X tiempo o la palabra "delay" que detendrá momentáneamente o retrasara el flujo que pase por ella.





4. Consola

La consola nos va a permitir saber **el estado** de nuestro algoritmo. En ella vamos a poder leer información referente a los diferentes pasos que nuestro algoritmo siga mientras es procesado. Es muy útil para detectar errores (bugs).

... en Arduino

de las conexiones con nuestro Arduino.
si ha sido subido correctamente a
s ver rodeada en color rojo en la

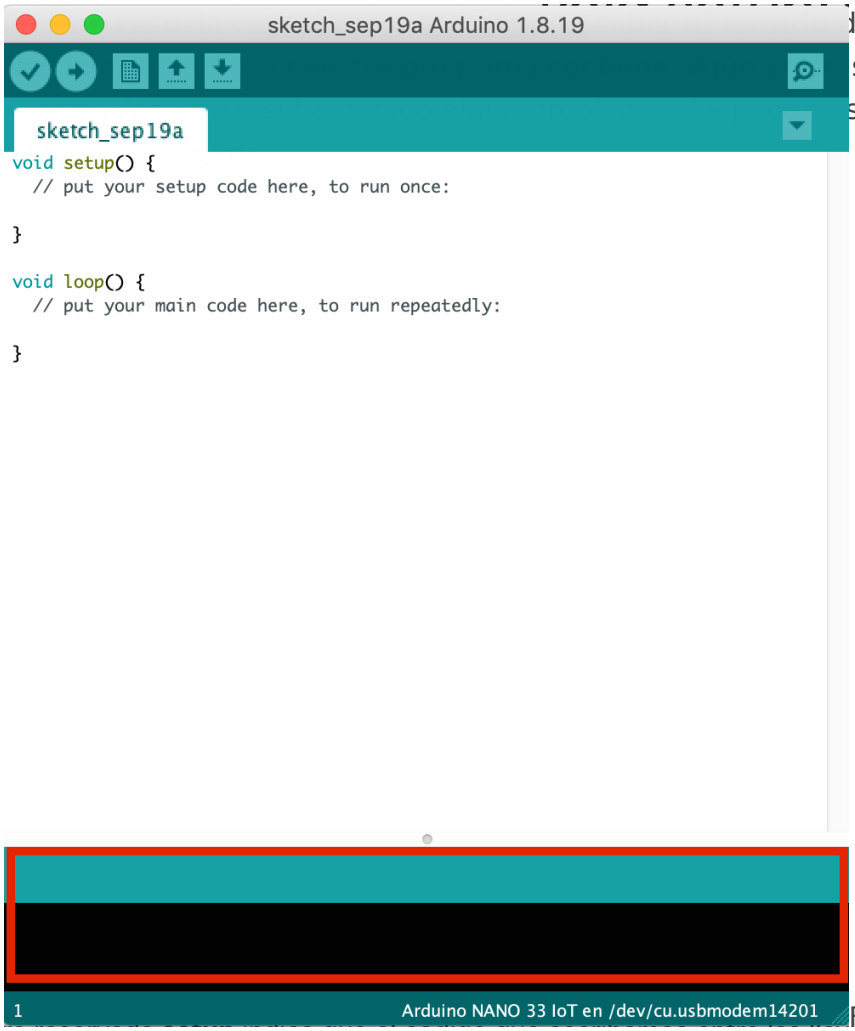


Figura 4. Sketch de Arduino con la zona correspondiente a la consola marcada en rojo

...en Pure Data

La ventana principal se abre cuando ejecutamos Pd y tiene esta apariencia:

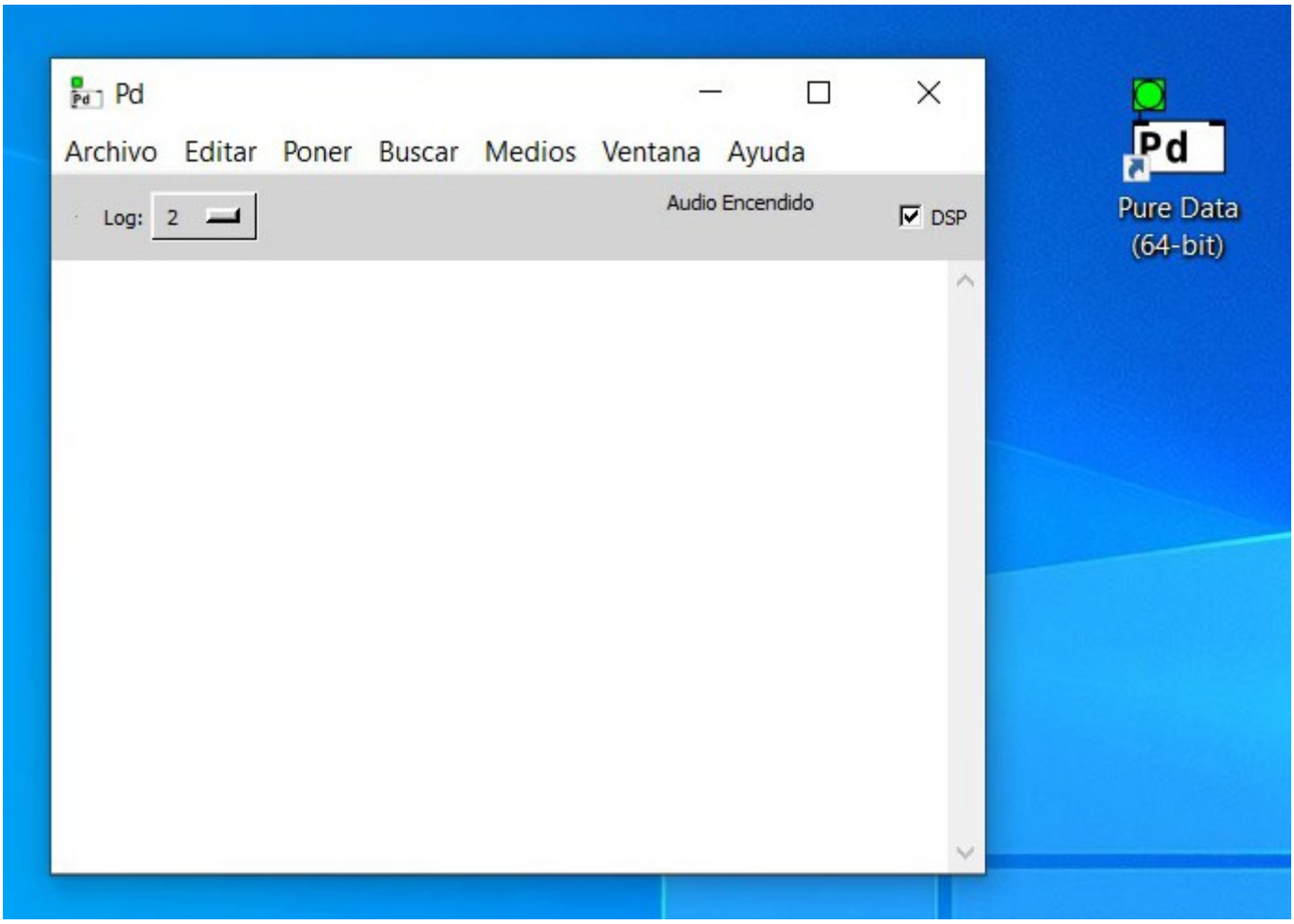


Figura 5. Ventana principal de Pure Data.

En la parte inferior tenemos un **espacio para la impresión** de contenido desde patches de Pd y/o mensajes del propio Pd, esta es la **consola**. Para imprimir desde un patch utilizaremos principalmente el objeto "**print**".
Este objeto es útil para visualizar el estado de los datos en diferentes partes de la estructura/programa que creemos, nos va a ayudar a encontrar errores y a entender mejor cómo funciona nuestro programa.

Los mensajes que Pd envíe aparecen en la consola y nos dan información acerca de errores y procesos que realiza el programa. El botón de **"Log"** nos permite filtrar la cantidad de mensajes enviados por Pd que queremos ver en ese espacio. Siendo "0 fatal" la mínima cantidad de información mostrada y "4 todo" la máxima. Inicialmente vamos a tener el Log en "2 normal" y ampliaremos la información mostrada en función de nuestras necesidades a la hora de resolver posibles problemas.

Copyright 2026

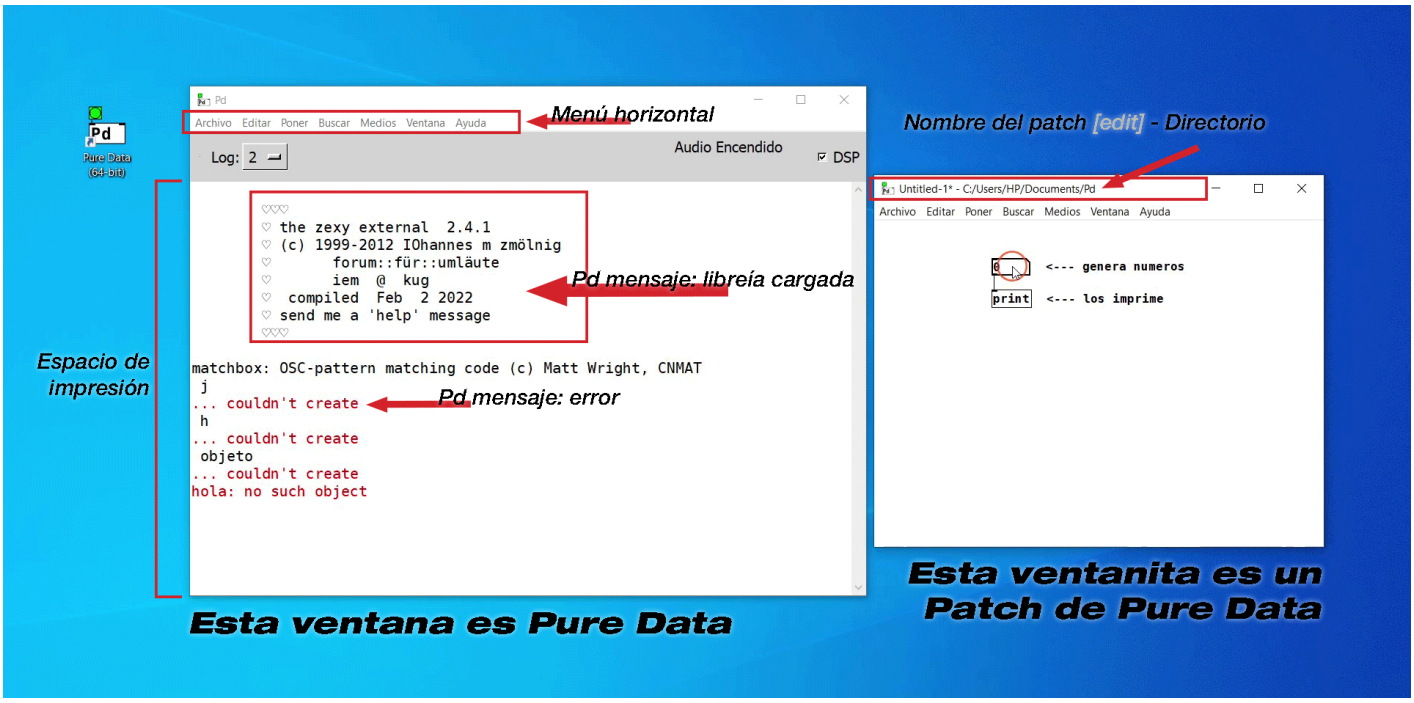


Figura 6. Ventana principal de Pure data (izquierda) y un patch/programa de Pure data (derecha).

REFERENCIAS:

Cerrada Somolinos, J. A., & Collado Machuca, M. E. (2015). *Fundamentos de programación*. Editorial Universitaria Ramón Areces : UNED.

Figuras:

Figura 1. Captura de pantalla de la IDE de Arduino

Figura 2. Un comentario en Pure data cuando esta recién creado.

Figura 3. La cajita exprimidor. https://giphy.com/gifs/hulu-snl-saturday-night-live-nbc-3o7TKPdUkkbCAVqWk0

Figura 4. Captura de pantalla de la IDE de Arduino

Figura 5. Ventana principal de Pure Data.

Figura 6. Ventana principal de Pure data (izquierda) y un patch/programa de Pure data (derecha).

Copyright 2026

Revision #33
Created 2022-09-17 14:39:24 CEST by Marta P. Campos
Updated 2023-12-05 12:43:03 CET by Julia del Río

Copyright 2026