Práctica 1.1: Descargar la IDE de Arduino y encender un LED

> Esta práctica es obligatoria, ya que realizándola nos aseguramos poder continuar el taller correctamente. Para su entrega, se especificarán los detalles en Moodle.

Y es que después de la teoría tenía que llegar la práctica. Ahora que ya conocemos las partes de nuestro 33IoT, es el momento de ver qué podemos hacer con él, comenzando por lo más sencillo: **encender un LED**.

Para evitar tener que realizar conexiones, vamos a encender el LED con el que cuenta nuestra placa. A este LED se le conoce con el nombre ***LED_BUILTIN***, y esa es exactamente (no vale cambiar las mayúsculas por minúsculas, añadir espacios o escribir los guiones mal) la palabra que utilzaremos para encenderlo y apagarlo.

Instalar la IDE de Arduino

Lo primero que tenemos que hacer es instalar la IDE (Entorno de Desarrollo Integrado) de Arduino en nuestro ordenador. Dependiendo del sistema operativo que empleemos, tendremos que seguir unas instrucciones u otras.

A continuación te dejo los enlaces para instalarlo tanto en Windows, Linux o Mac.

Arduino IDE Linux

Arduino IDE macOS

Arduino IDE Windows

La descarga e instalación nos puede llevar de 5 a 20 minutos, dependiendo de la velocidad de nuestra conexión a internet y del sistema operativo que tengamos, ya que el procedimiento varía

Copyright 2026

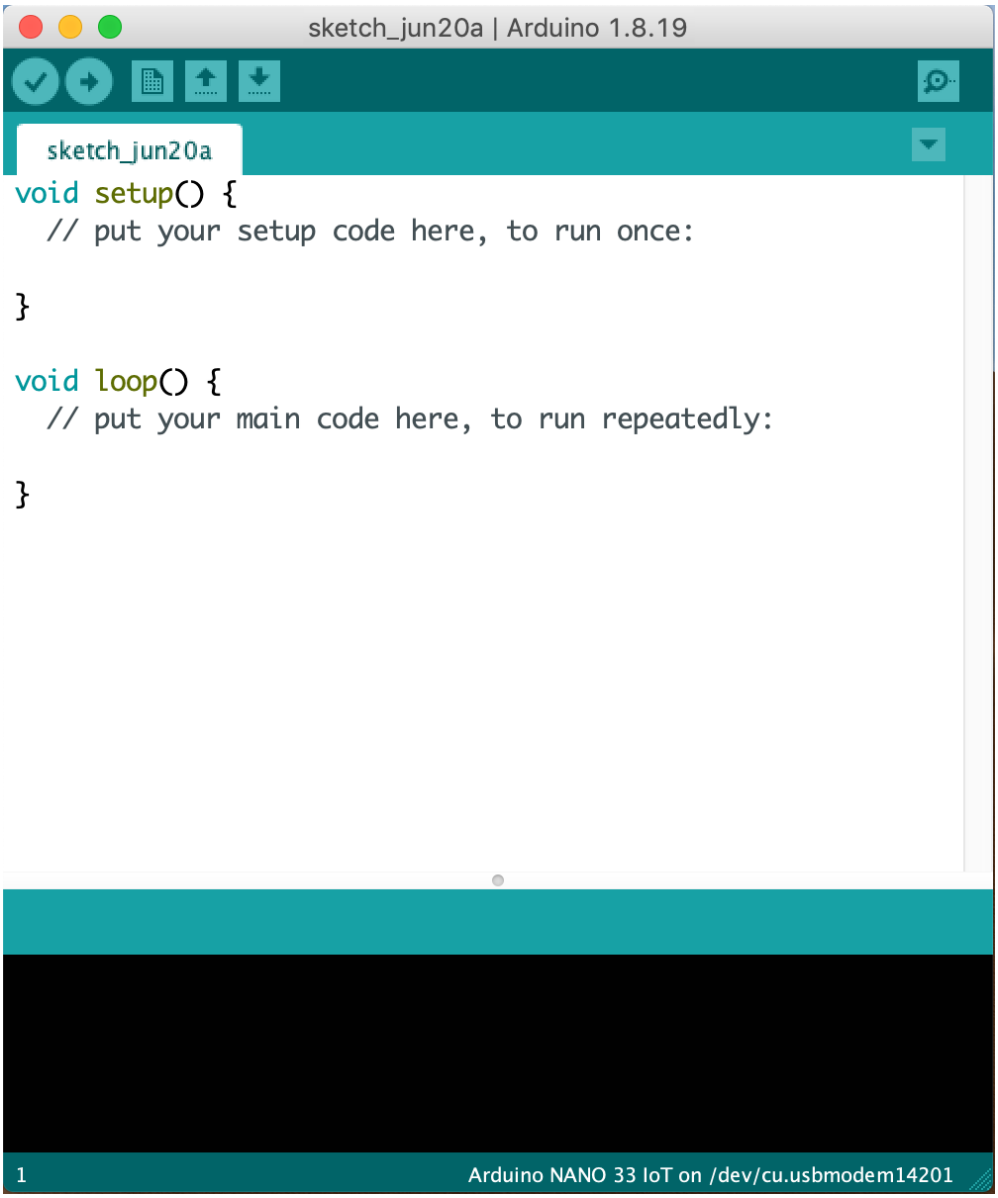
notablemente de uno a otro.

> En este curso, recomendamos instalar la IDE de Arduino en vuestro ordenador, aunque si durante la instalación tuviéseis muchos problemas (que no suele ser común, aunque puede suceder si, por ejemplo el ordenador que utilizáis tiene un sistema operativo muy antiguo) podrías utilizar la versión online. Para ello es necesario crear una cuenta en la web de Arduino y seguir los siguientes pasos.

Abrimos la aplicación

Una vez hayamos instalado la IDE, estaremos listos para abrirla y nos encontraremos con algo muy similar a esto:

Copyright 2026



A través de esta aplicación programaremos nuestro Arduino. Lo primero que vamos a hacer es ver qué podemos hacer con algunos de los botones que utilizaremos con más frecuencia. Comenzaremos de arriba a abajo y de izquierda a derecha.

> Se recomienda abrir la aplicación para poder ir viendo cada botón conforme se explica su función.

- **Símbolo tick (Verificar):** con él, comprobaremos que el código que queremos subir a nuestro Arduino no contiene fallos sintácticos. ¿Dónde nos comunicará la aplicación si nuestro código tiene fallos o no? Lo veremos en el rectángulo negro de la parte inferior. Esa parte se denomina consola. ¿Siempre que nuestro código pase satisfactoriamente el paso de verifiación nos aseguramos de que funciona correctamente? Pues,

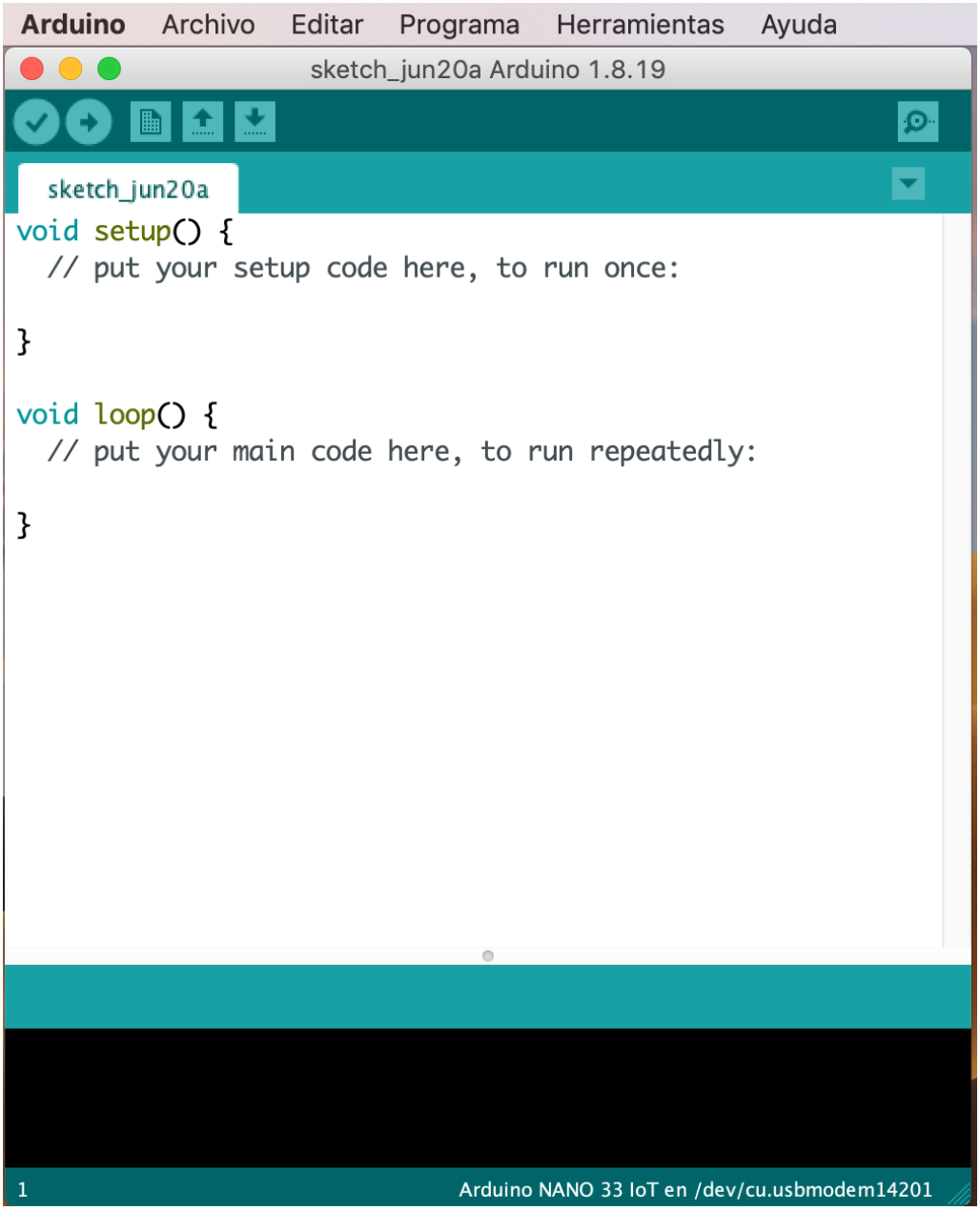
Copyright 2026

desafortunadamente no. Esta verificación nos asegura que no existen fallos sintácticos: palabras mal escritas, expresiones incorrectas... pero no nos asegura que la lógica de nuestro programa sea la correcta.

- **Flecha a la derecha (Cargar):** una vez nos hayamos asegurado de que nuestro código es correcto sintácticamente, al darle a la flecha, comenzaremos a cargarlo en nuestra placa Arduino.
- **Hoja de papel (Nuevo):** este botón abre una pestaña nueva para que podamos escribir código. No la utilizaremos en este taller.
- **Flecha arriba (Abrir):** nos permite abrir un archivo (sketch) existente.
- **Flecha abajo (Guardar):** permite guardar el archivo en el que estamos trabajando actualmente.
- **Lupa (monitor serial):** abre una ventana que nos permite ver la información serie que la placa transmite. Es muy útil para encontrar fallos en el código y ver si el programa y la placa Arduino funcionan correctamente. Lo veremos con más detalle en las partes prácticas del taller.
- **Nombre del sketch:** en esta pestaña encontraremos el nombre de nuestro sketch. Por defecto será algo similar a *sketch_fecha*
- **Área de escritura:** en esta zona escribiremos nuestro programa.
- **Área de mensajes:** en esta parte, el IDE nos indicará si existen errores en el código.
- **Consola de texto:** la consola nos mostrará los mensajes de error completos, es muy útil a la hora de depurar nuestro código.
- **Placa y puerto serie**: se muestra el nombre de la placa (Arduino) que tenemos conectada (o la última que hemos conectado, si no hay ninguna conectada) y el puerto al que está conectada.

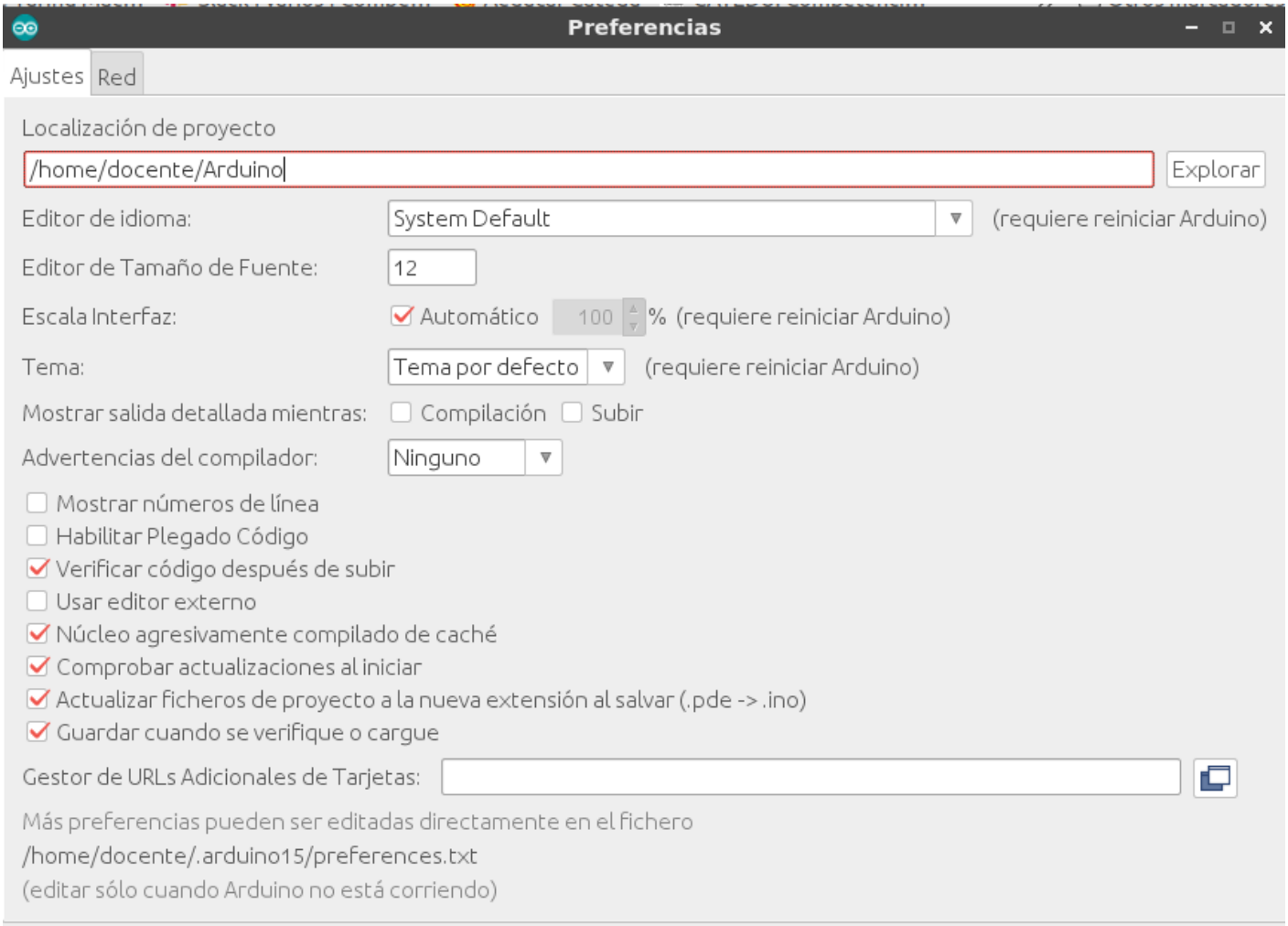
Aparte de estas herramientas, contamos con una barra de herramientas en la parte superior (Arduino, Archivo, Editar...). El aspecto de esta barra de herramientas puede variar ligeramente de un sistema operativo a otro, pero las funciones serán las mismas. Algunas de ellas las veremos a continuación.

Copyright 2026



Cuando Arduino IDE se instala por primera vez en un equipo, crea en el mismo una carpeta llamada **Arduino** donde almacenará las bibliotecas y otras utilidades que necesita, y donde colocará nuestros proyectos si nosotras no le indicamos lo contrario. Para saber en nuestro equipo dónde ha colocado esa carpeta, podemos verlo desde el menú **Archivo/Preferencias.**

Copyright 2026



Por supuesto podemos modificar la ubicación de esta carpeta desde aquí, cambiar el idioma, así como algunas otras opciones que pueden ser interesantes para la **accesibilidad de la herramienta** como el tamaño de letra o el tema a utilizar. Para más opciones sobre cómo configurar la accesibilidad de arduino por ejemplo para personas con deficiencia visual consulta aquí.

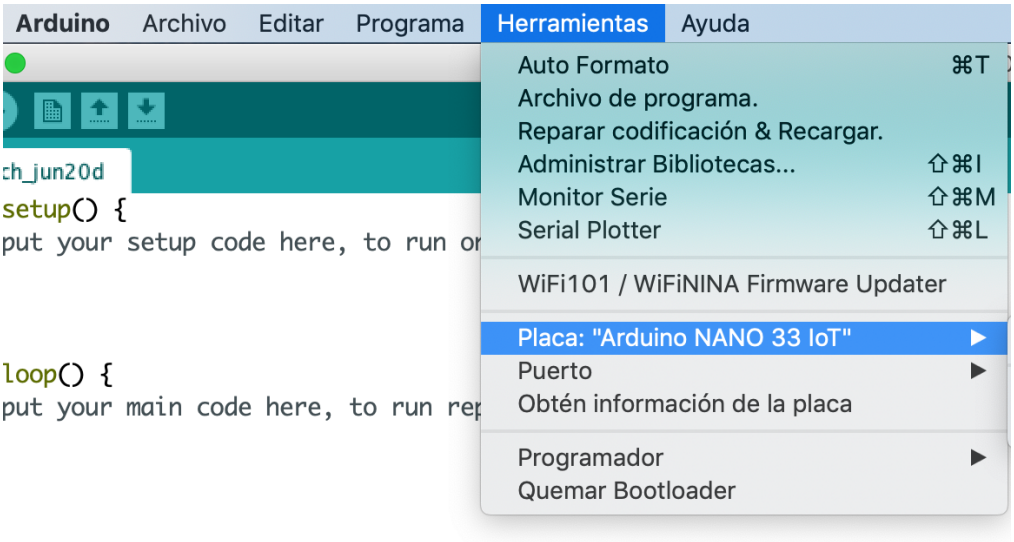
Conectamos nuestro 33 IoT al ordenador

Llegadas a este punto, estamos listas para conectar nuestro Arduino al ordenador. Para ello, necesitaremos nuestro cable USB-microUSB.
Una vez hayamos conectado un extremo a nuestro Arduino y el otro al puerto USB de nuestro ordenador, tendremos que ayudar a la IDE a que reconozca a nuestro Arduino. Lo primero que veremos será un LED amarillo que se enciende y junto al que pone ON. Eso nos indica que la placa está recibiendo corriente desde nuestro ordenador.

Copyright 2026

> Los usuarios de Windows (seguramente) tendrán que instalar una serie de drivers una vezse conecte el Arduino al ordenador.

Lo siguiente que tendremos que hacer es ir a la barra de herramientas superior y precisamente en **Herramientas** tendremos que configurar dos cosas, **la placa y el puerto**:



Configuramos la placa

En la imagen superior, en **Placa** ya nos aparece nuestro 33 IoT, pero cuando lo abras por primera vez.... no tendrás tanta suerte. Lo que tendrás que hacer es descargar lo que se conoce como **SAMD core** y que permitirá que nuestro ordenador reconozca la placa con la que vamos a trabajar. Como puedes comprobar, es una placa un poco especial, que requiere una serie de pasos hasta que podamos conectarla correctamente a nuestro ordenador.

Para ello, tendremos que ir a **Herramientas > Placa > Gestor de tarjetas.**

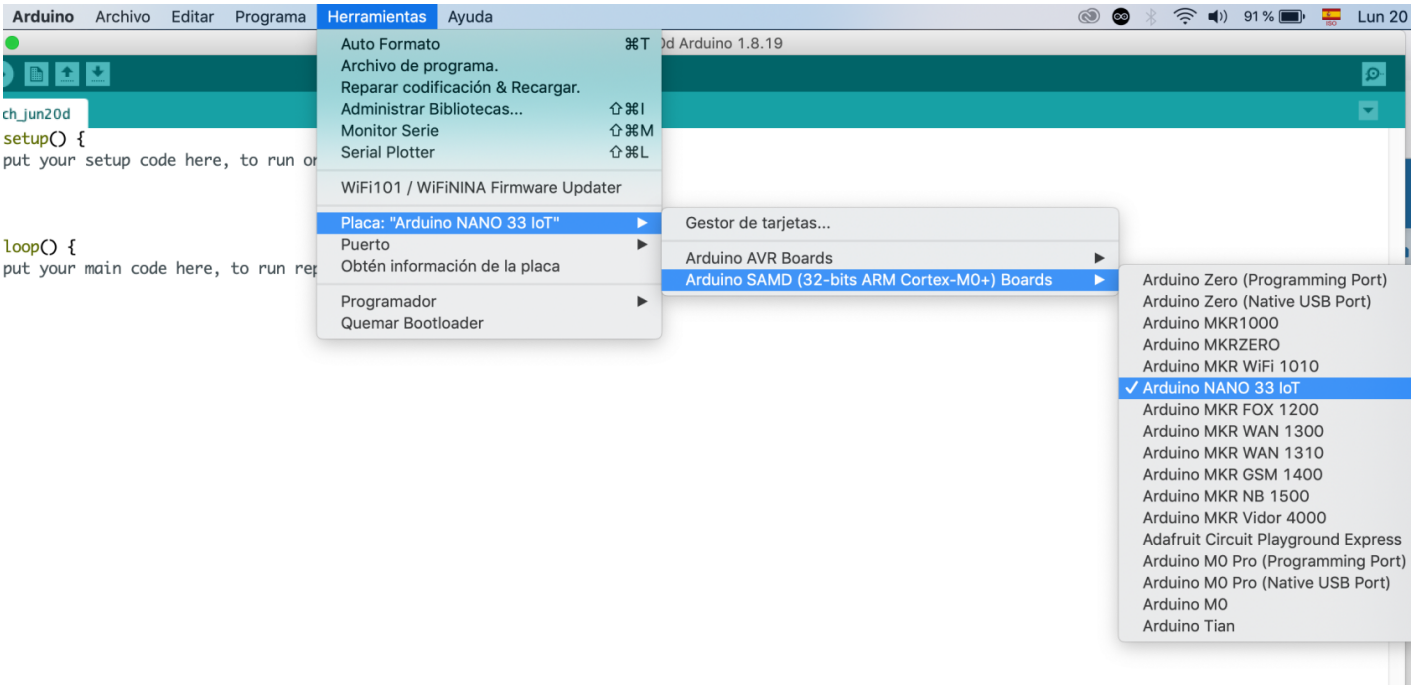
Una vez ahí, nos aparecerán una serie de cores que podemos instalar, pero el que nosotras necesitaremos será el **Arduino SAMD Boards (32-bits ARM Cortex-MO+)**:



Lo podemos buscar en la barra de búsqueda de la derecha o hacer scroll hacia abajo hasta que lo encontremos. Una vez lo hayamos localizado, lo instalaremos haciendo clic en **INSTALAR** (en la imagen superior no se puede instalar, porque ya está instalado).

El proceso de instalación puede tardar algunos minutos, de nuevo depende de la conexión a internet y de la velocidad del ordenador desde el que estemos realizando la instalación.

Una vez lo hayamos instalado, nos aparecerá la palabra **INSTALLED**, como puedes ver en la imagen superior, y a continuación podremos ir a **Herramientas > Placa > Arduino SAMD (32-bits ARM Cortex-MO+) Boards** y seleccionar la nuestra: **Arduino Nano 33 IoT.**



Al seleccionarla, le estamos diciendo a nuestro Arduino "¡Hey! Prepárate, voy a conectarte un Arduino Nano 33 IoT", por lo que si al final conectásemos otro tipo de Arduino, la IDE no la reconocería y no podríamos programarla.

> Si utilizamos otro tipo de Arduino, lo primero que tenemos que hacer es decírselo a Arduino seleccionando la placa correcta en **Placa**, como acabamos de hacer.

Configuramos el puerto

Una vez le hemos dicho la placa que vamos a conectar, nuestro IDE necesita saber en qué puerto USB la hemos conectado. Para ello, tendremos que ir a **Herramientas > Puerto**



Para los usuarios de **Windows** podría tener esta apariencia:

```
<COM29> (Arduino NANO 33 IoT)
```

Para los de **MAC** o **Linux**, la apariencia puede ser parecida a esta:

```
/dev/cu.usbmodem14112 (Arduino NANO 33 IoT)
```

Una vez hayamos seleccionado nuestra placa y puerto, pasaremos a encender el LED.

Y se hizo la luz

Para encender nuestro LED, vamos a utilizar un ejemplo de los que se nos proporcionan desde la propia IDE de Arduino.

> Estos ejemplos nos pueden ser útiles para descubrir lo que podemos hacer con estas placas, por lo que os recomiendo que les echéis un vistazo.

El ejemplo que vamos a utilizar se llama **Blink** y nos va a venir bien para asegurarnos de que la instalación y toda la configuración se han realizado correctamente. Para abrir y subir este ejemplo a nuestro Arduino tenemos que navegar a: **Archivo > Ejemplos > 01.Basics > Blink**.



Una vez hagamos click, se abrirá una nueva ventana de Arduino con el código que permitirá que nuestro LED comience a parpadear. El código que veremos será el siguiente:

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
```

```
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

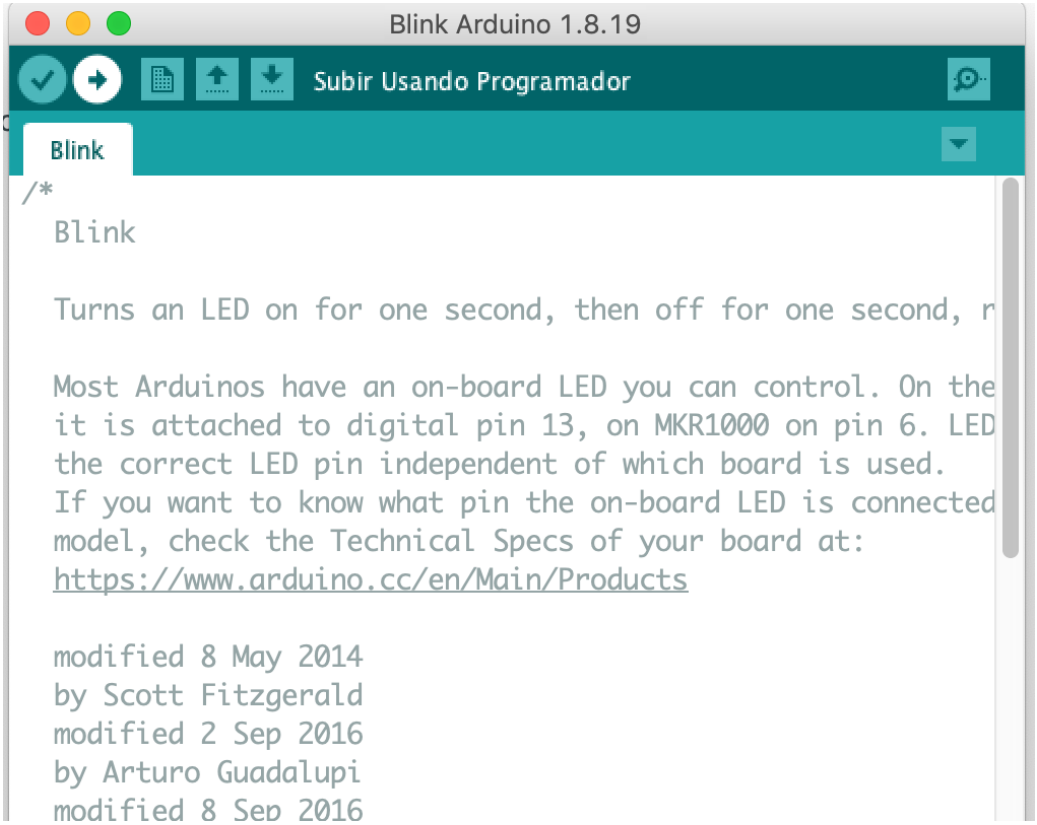
  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

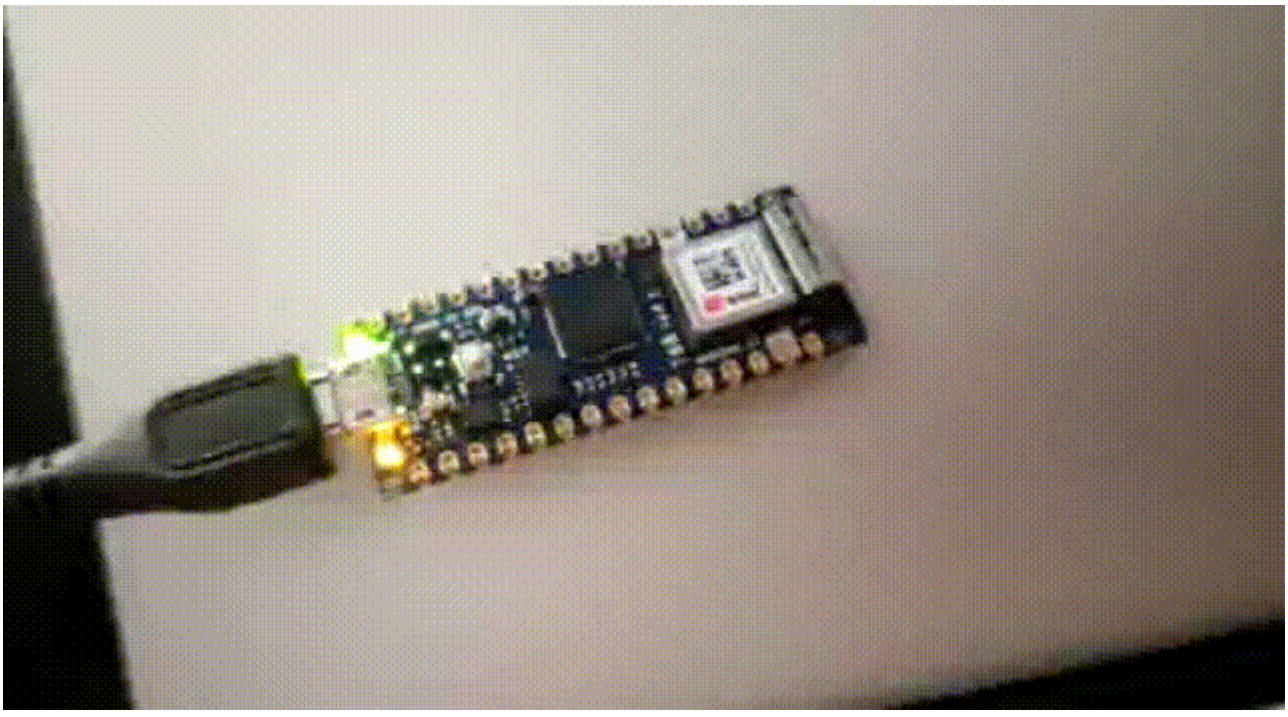
Ahora, no vamos a entrar a explicar qué significa este código, eso lo haremos en la página siguiente. Lo que sí vamos a hacer es darle a la flecha a la derecha para subir ese código a nuestro 33IoT:

Copyright 2026



Y si todo ha ido bien, lo que veremos será un LED naranja, justo en el lado del microUSB en el que no está el LED amarillo, permanecer un segundo encendido, un segundo apagado.

¡Y ya lo tenemos!



Es posible que nos de algún error y que no funcione. Eso no significa que algo esté roto. Como ya avisé en la primera página del taller, no siempre funciona todo a la primera. Si en la consola vemos algún error, tendremos que asegurarnos de que hemos seleccionado el puerto y la placa correctas y que hemos seguido todos los pasos correctamente, sin olvidar ninguno.

Ha habido casos en los que usuarios de Linux se han encontrado con un **problema de permisos**. Puede que el puerto esté bien seleccionado, pero que el usuario no tenga permiso para acceder al puerto. Para obtener permiso ese necesario emplear el comando: "sudo usermod -a -G dialout ". Tras reiniciar debería funcionar.

FUENTES:

Arduino Nano 33 IoT: https://docs.arduino.cc/hardware/nano-33-iot

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
GOBIERNO DE ARAGON

Revision #25
Created 2022-06-20 15:02:38 CEST by Marta P. Campos
Updated 2023-04-03 14:30:17 CEST by Marta P. Campos

Copyright 2026