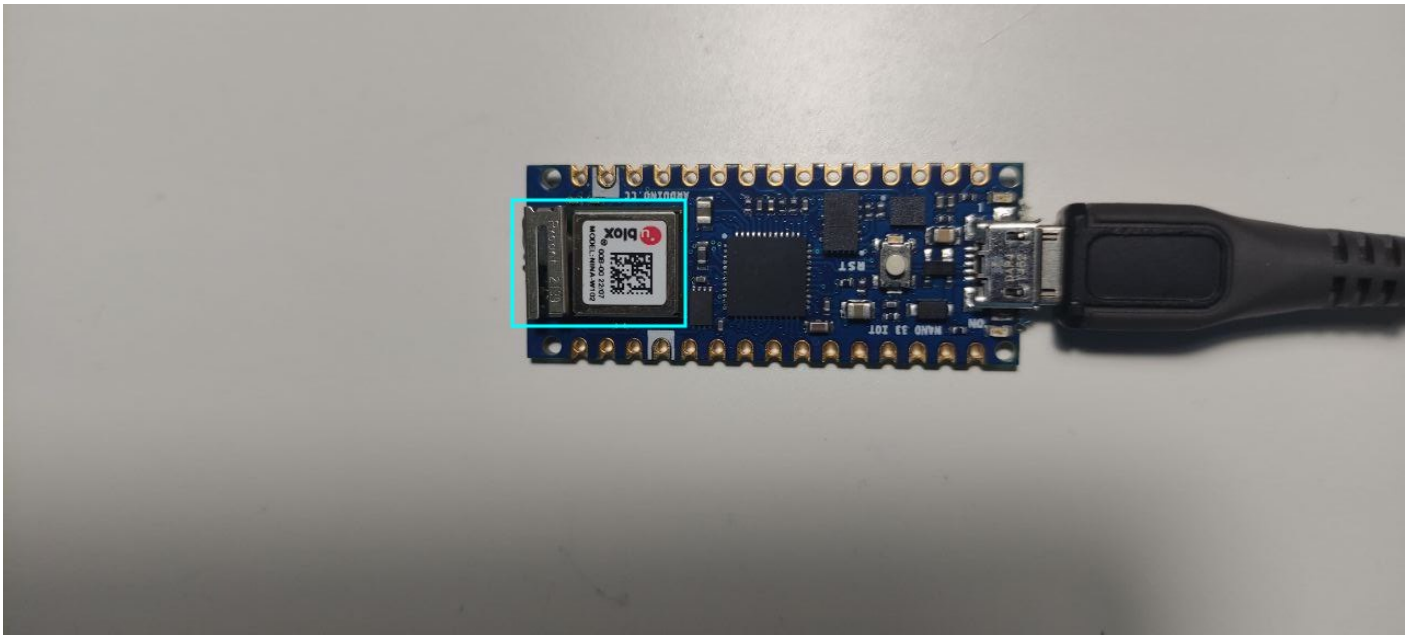
Conectemos nuestro microcontrolador a la red Wifi

Antes de comenzar con la parte práctica sobre el internet de las cosas (IoT), vamos a ver un componente de nuestro Arduino Nano 33 IoT que es el que nos va a permitir la interconexión con redes Wi-Fi.

Como ya hemos visto, nuestro Arduino tiene un LED integrado, un acelerómetro y un giroscopio; pues aparte de eso, también tiene un módulo que permite que nos conectemos a una red Wi-Fi: el **NINA-W102**.

¿Qué es el módulo NINA-W102?



El componente que vemos dentro del recuadro azul claro es el que nos va a permitir la conexión a una red Wi-Fi. Consta de una memoria Flash, un cristal, una antena y otros componentes de adaptación y filtrado que nos permiten conectarnos con seguridad a la red.

Copyright 2026

> La antena es la parte que sobresale un poco, en la imagen es la parte que está más a la izquierda. **Esta antena es bastante delicada**, al igual que el conector USB, así que tenemos que tener cuidado de no golpearla o romperla, porque, si no, el módulo dejará de funcionar

La librería WiFiNINA

Esta librería nos va a permitir utilizar el módulo NINA-W102 sin tener que entrar en programación demasiado complicada. Desde los años 90 hasta ahora, se han desarrollado protocolos de seguridad que permiten conexiones más seguras como el WEP, WPA y WPA2, todos ellos son soportados por nuestra librería WiFiNINA.

> Si quieres saber más sobre estos tipos de cifrado, haz click aquí.

¿Qué vamos a hacer en este proyecto?

En este proyecto, vamos a:

1. Obtener la información de una red Wi-Fi como el SSID, la IP de la placa y la fuerza de la señal. Con esta información podremos hacer que el LED_BUILTIN de la placa parpadee más rápido o más lento en función de la intensidad de la señal de la red.

2. Aprender cómo usar el Arduino Web Editor. En lugar de usar la IDE de Arduino, que ya hemos utilizado para programar nuestro proyecto en las prácticas anteriores, en este caso vamos a programar nuestro proyecto desde el **editor online**. ¿Por qué? Aparte de para que veas que también tienes esta opción, porque nos va a facilitar la vida a la hora de crear una cosa llamada **Secret Tab**.

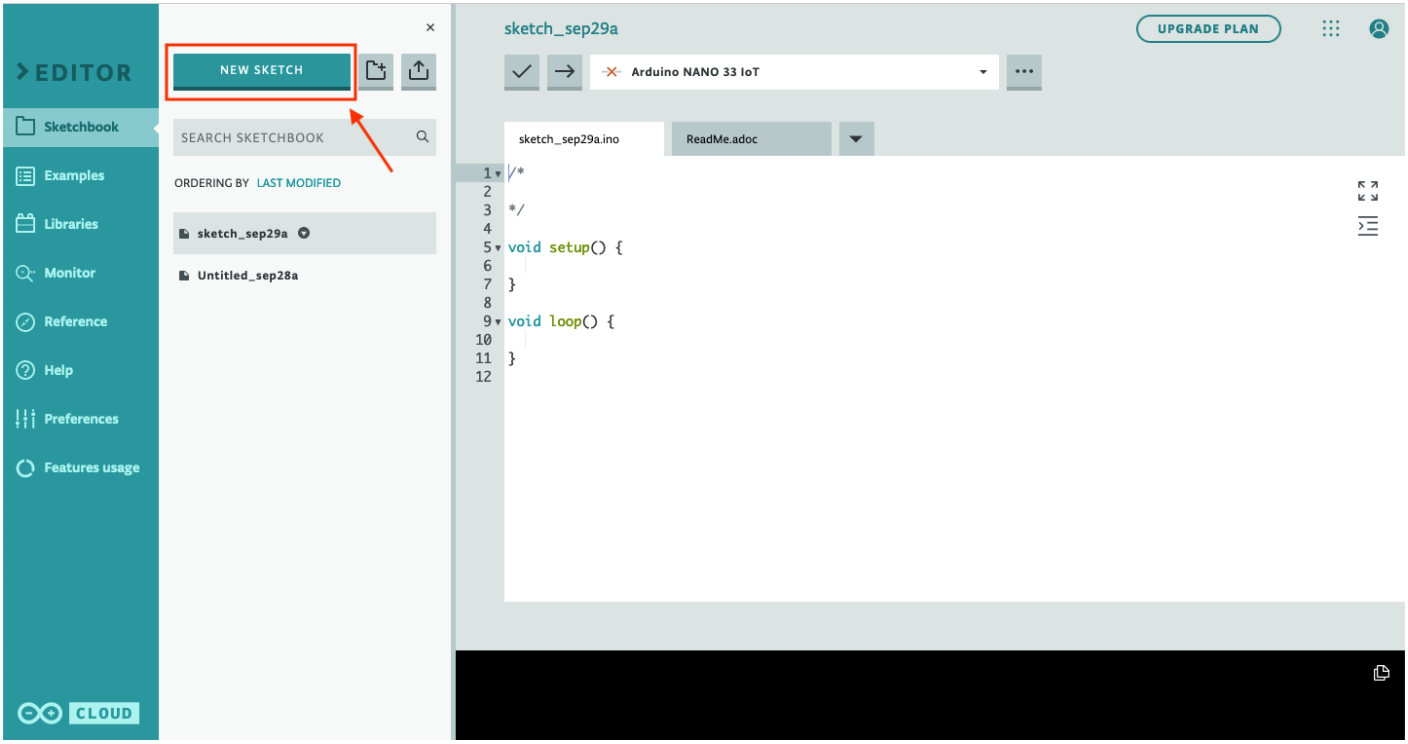
¡Comenzamos!

Para este proyecto vamos a necesitar la cuenta de Arduino que ya hemos creado y también será necesario haber instalado el Arduino Create Agent, que también deberías tener instalado si has seguido en orden este curso.

Lo primero que haremos será acceder a Arduino Web Editor.

Una vez hayamos introducido nuestro usuario y contraseña, veremos una pantalla similar a esta, en la que lo primero que haremos será crear un nuevo sketch:

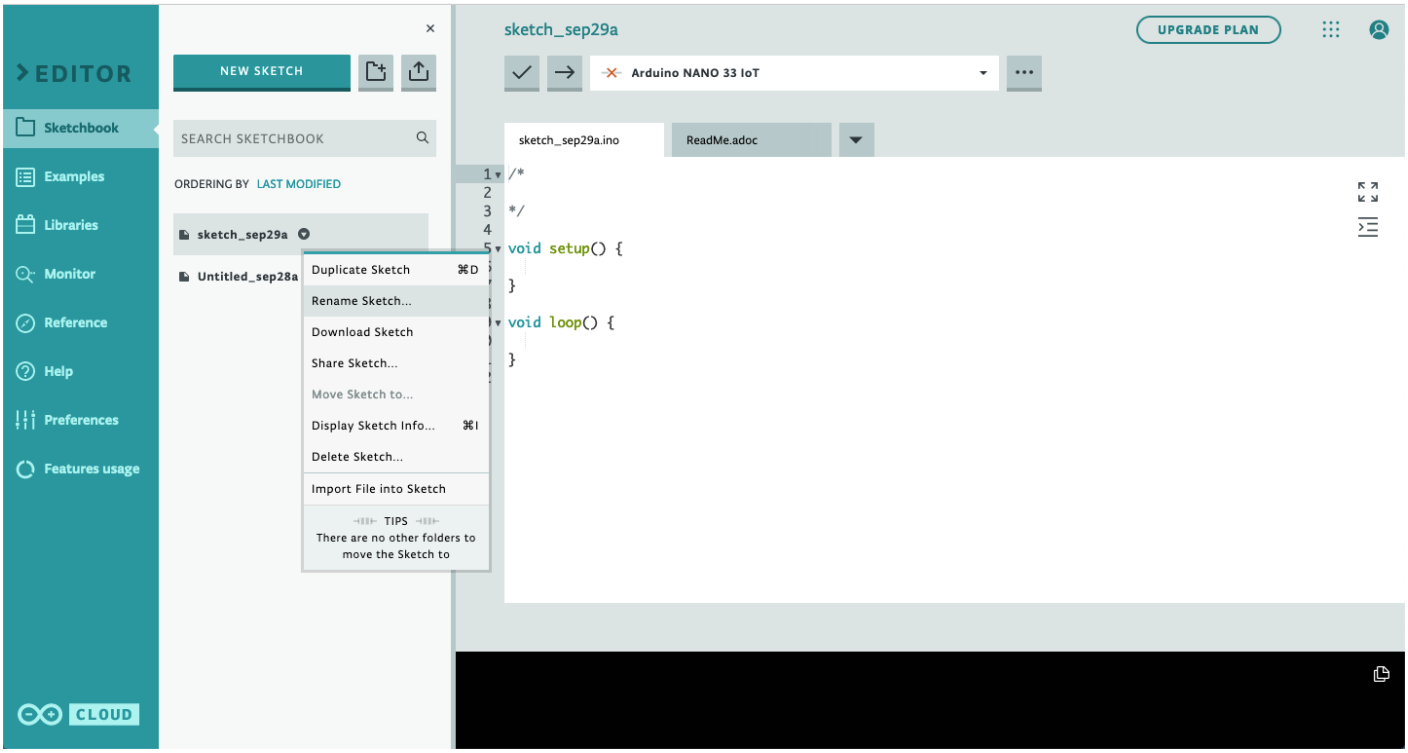
Copyright 2026



Por defecto, se habrá creado con el nombre 'sketch' seguido del mes, el día, y una letra del abecedario. En la imagen superior, puedes ver que el nombre por defecto que me ha creado ha sido **sketch_sep29a**. Así que, a continuación, renombraremos nuestro sketch de una manera que nos podamos acordar fácilmente de qué va el proyecto.

Por ejemplo, yo lo renombraría **Conexion-wifi**. Te aconsejo que no uses tildes ni caracteres especiales. Para renombrarlo, haremos click en la flecha que aparece a la derecha del nombre de nuestro sketch recién creado y seleccionaremos la opción **Rename Sketch...** :

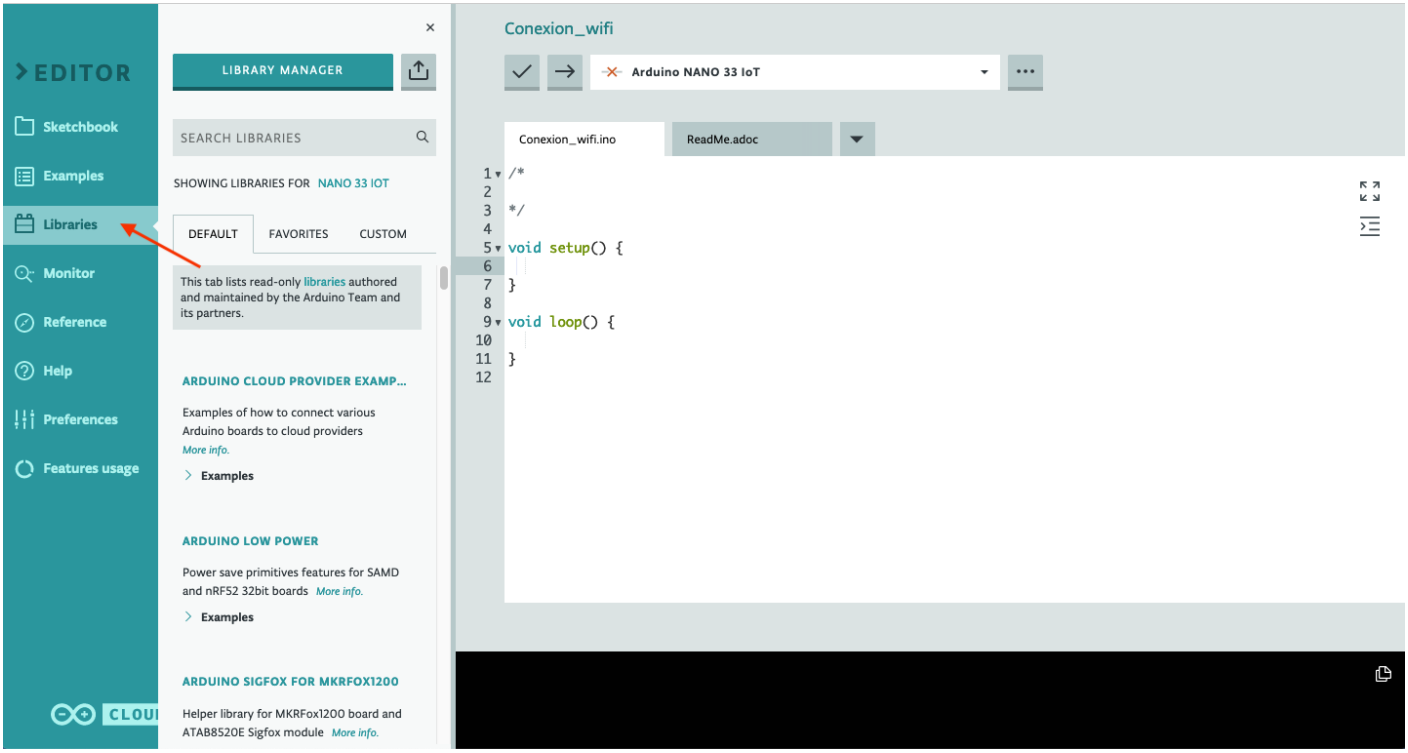
Copyright 2026



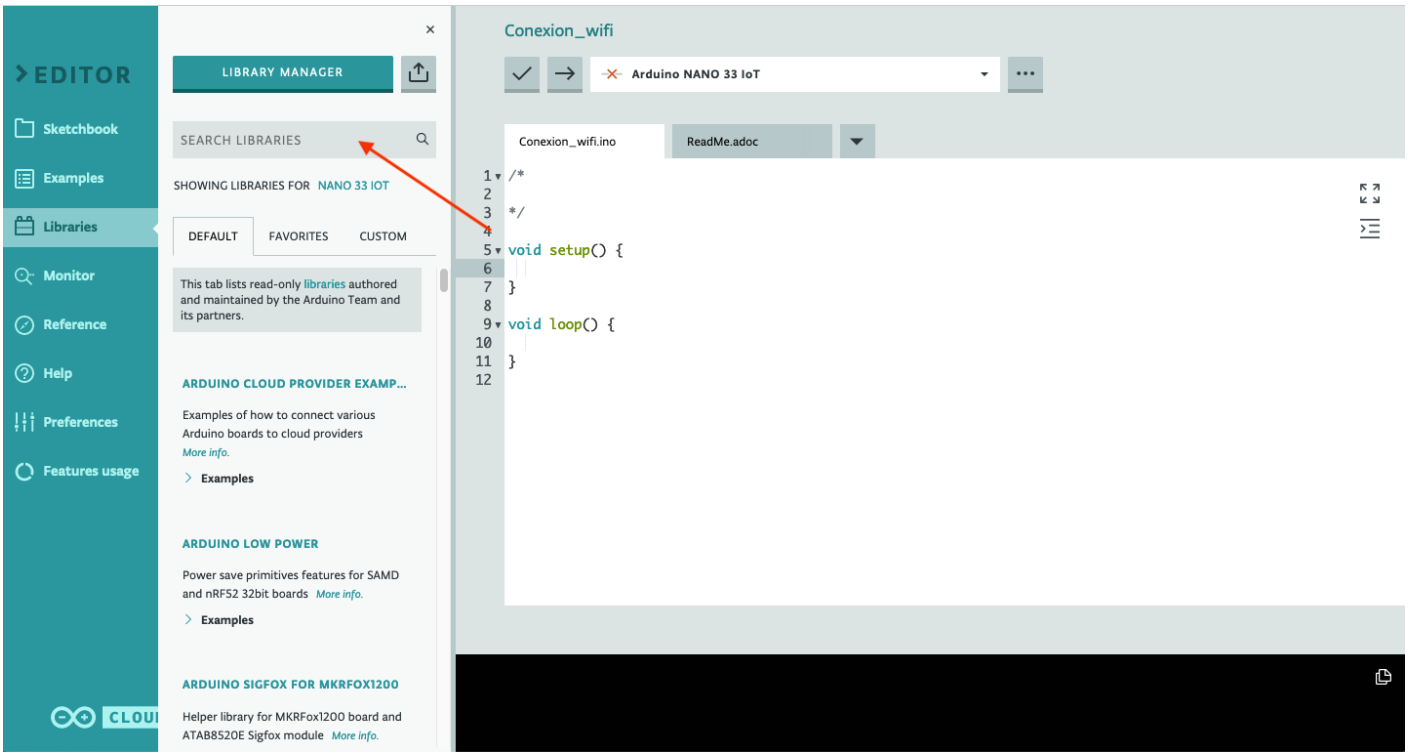
Importamos la librería

Una vez tengamos nuestro sketch creado y renombrado, vamos a importar la librería **WiFiNINA**. Para ello iremos a la columna azul de la izquierda y haremos click sobre **Libraries:**

Copyright 2026

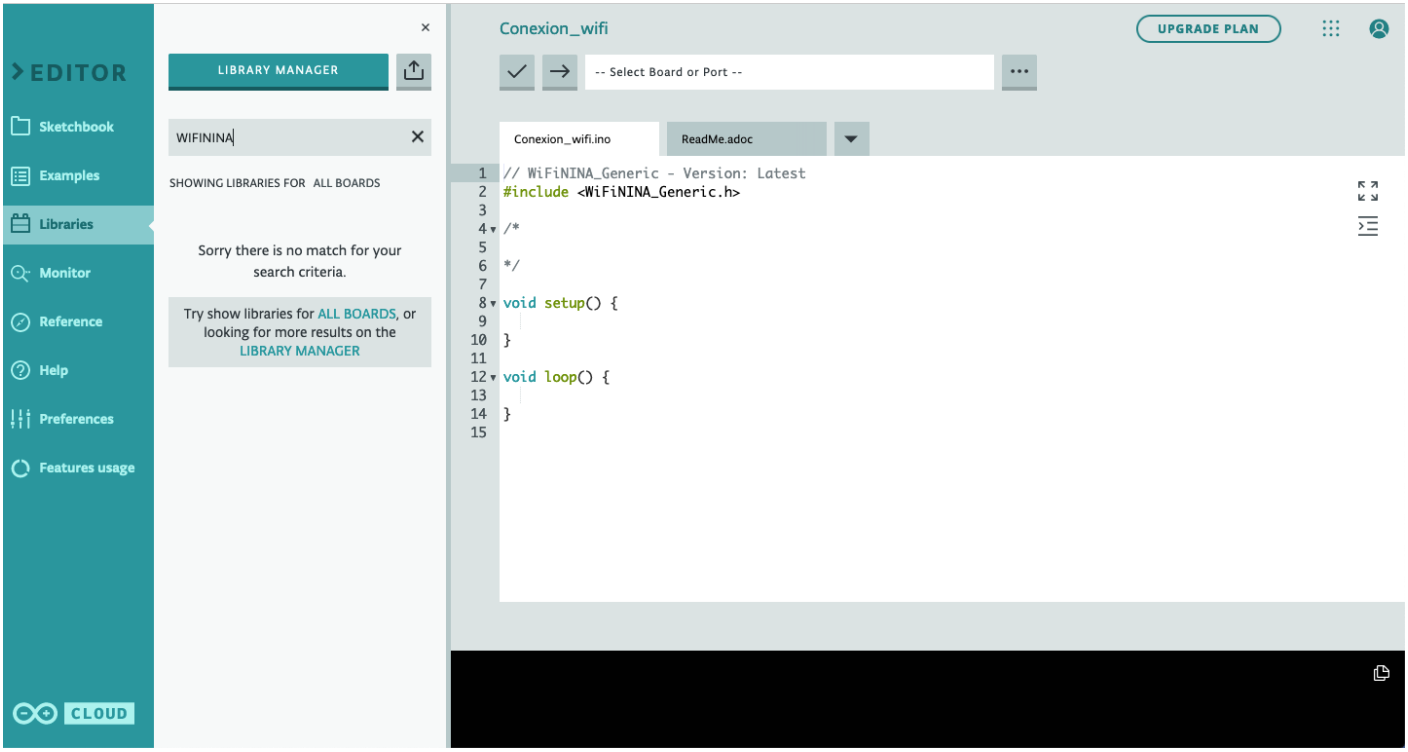


Veremos que en la columna blanca de la izquierda ahora nos aparecen diferentes opciones. En la parte superior pone **LIBRARY MANAGER** y justo debajo un campo en color gris, **SEARCH LIBRARIES** (te lo indico con una flecha)en el que nos permite buscar entre las diferentes librerías:



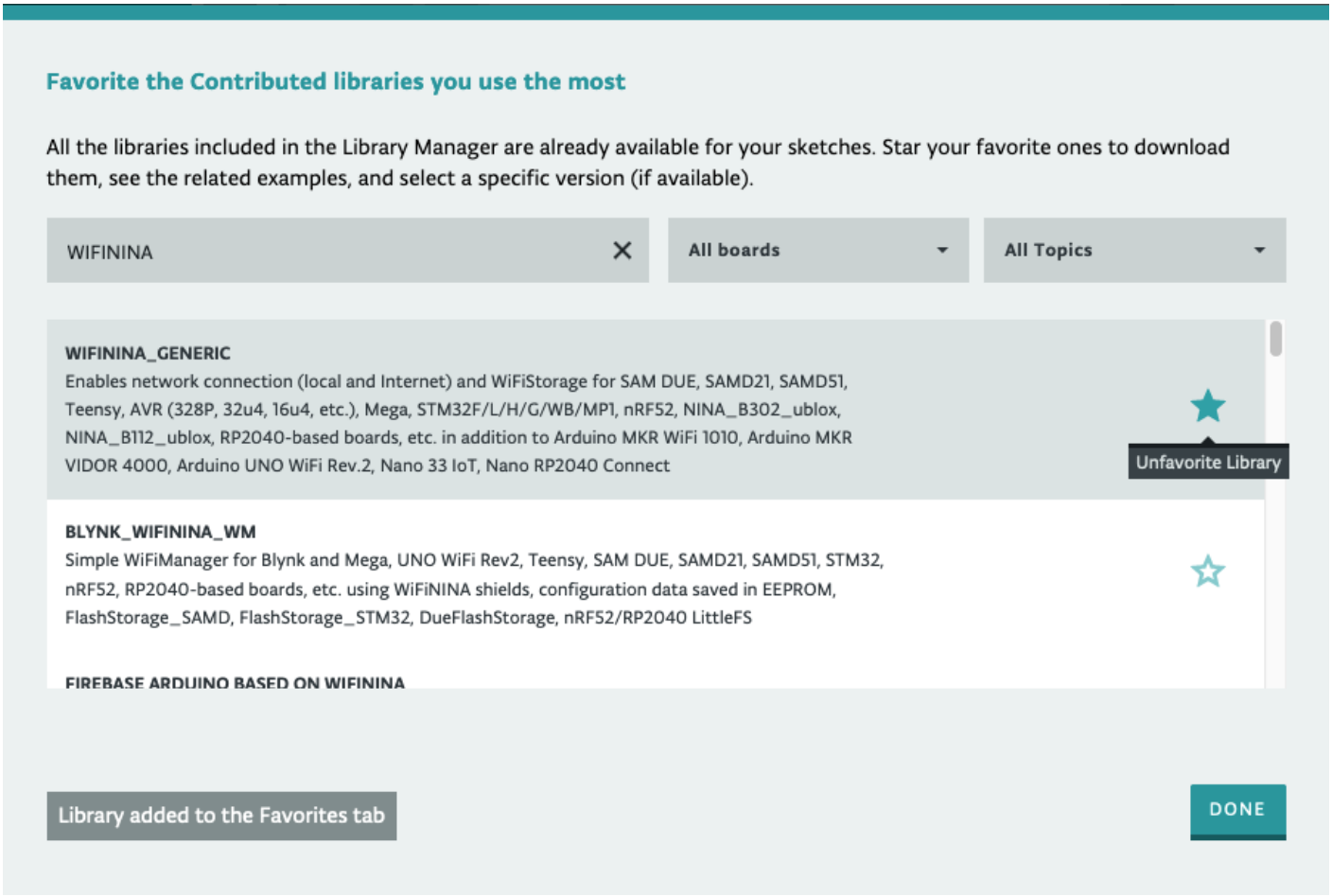
Copyright 2026

Cuando busquemos **WIFININA**, veremos que **no nos aparece**:



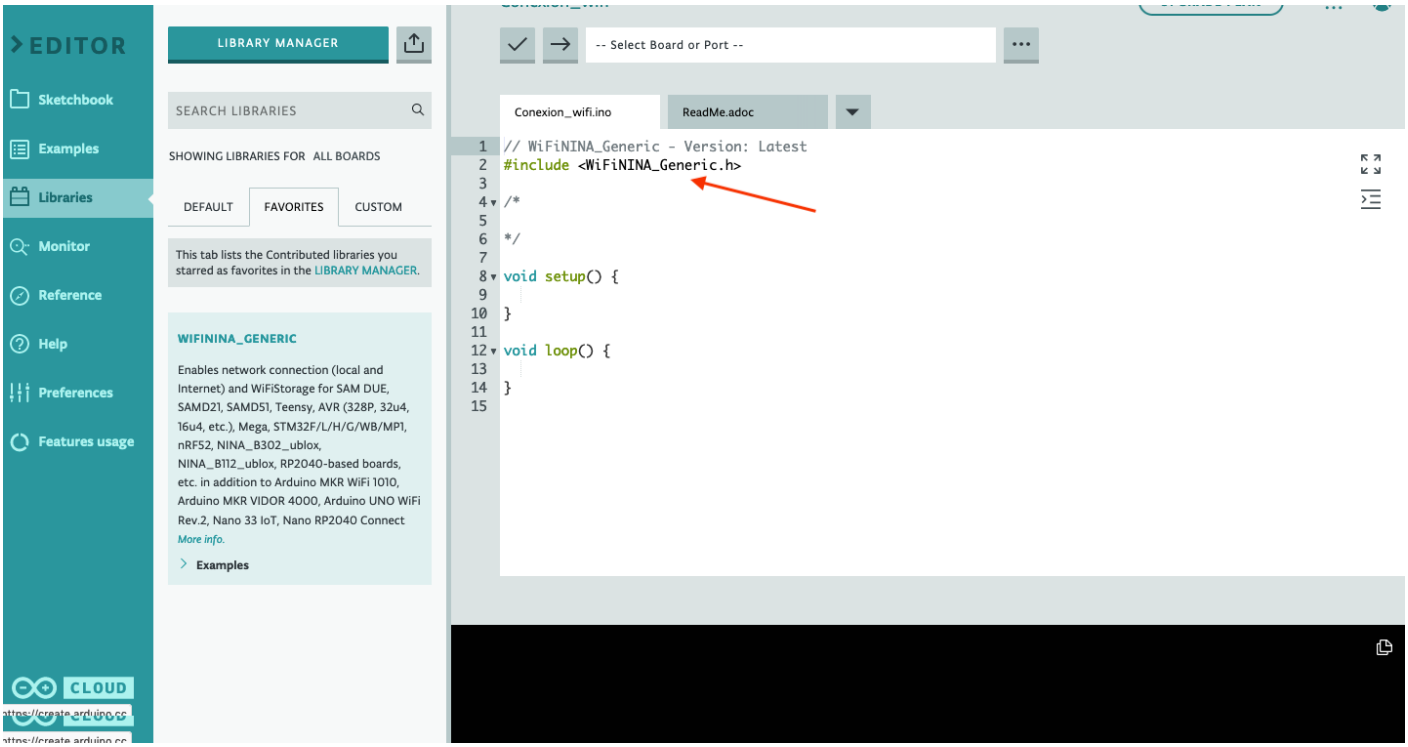
Así que, tendremos que hacer click en **LIBRARY MANAGER** y se nos abrirá una ventana emergente. Ahí encontraremos una librería que se llama **WIFININA_GENERIC.** Tendremos que añadirla a nuestras librerías favoritas haciendo click sobre la estrella:

Copyright 2026



Una vez la hemos añadido, haremos click en **DONE.** Nos aparecerá en la pestaña de nuestras librerías favoritas y desde ahí pondremos nuestro ratón encima de la librería y veremos que nos aparece el botón **INCLUDE.** Haremos click y veremos que una nueva línea se ha añadido a nuestro sketch:

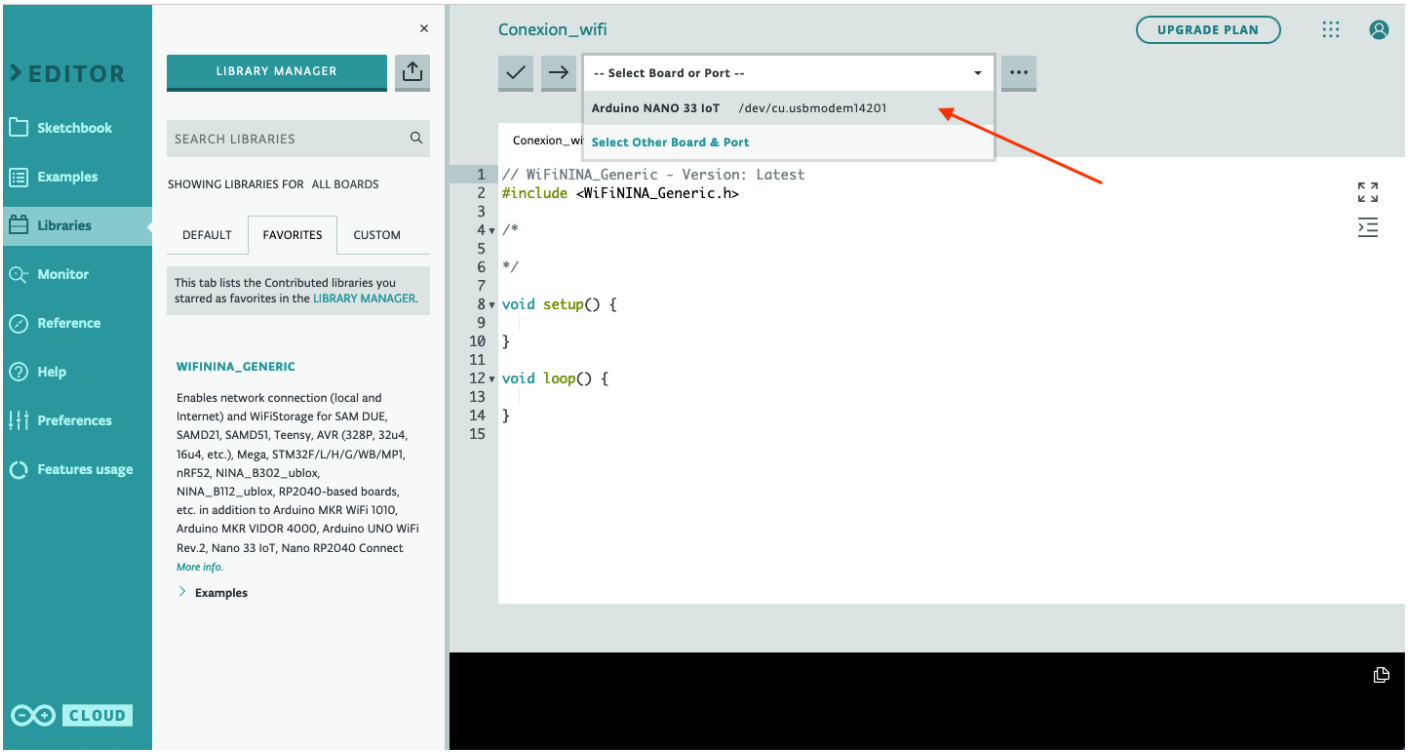
Copyright 2026

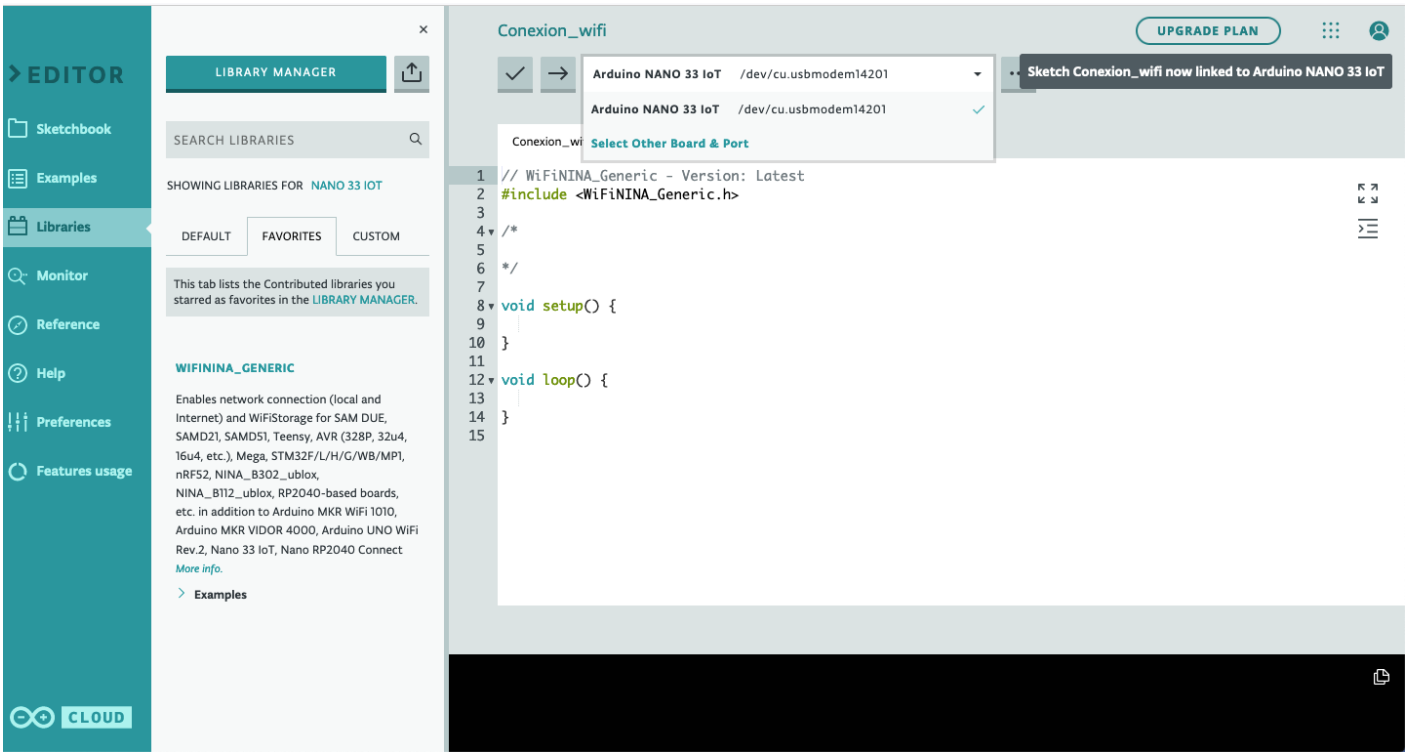


> Si recuerdas, este proceso de importar una librería ya lo hicimos en la Práctica 2.2. En ese caso desde la IDE de Arduino, en lugar desde el editor online.

Conectamos nuestro NANO 33 IoT

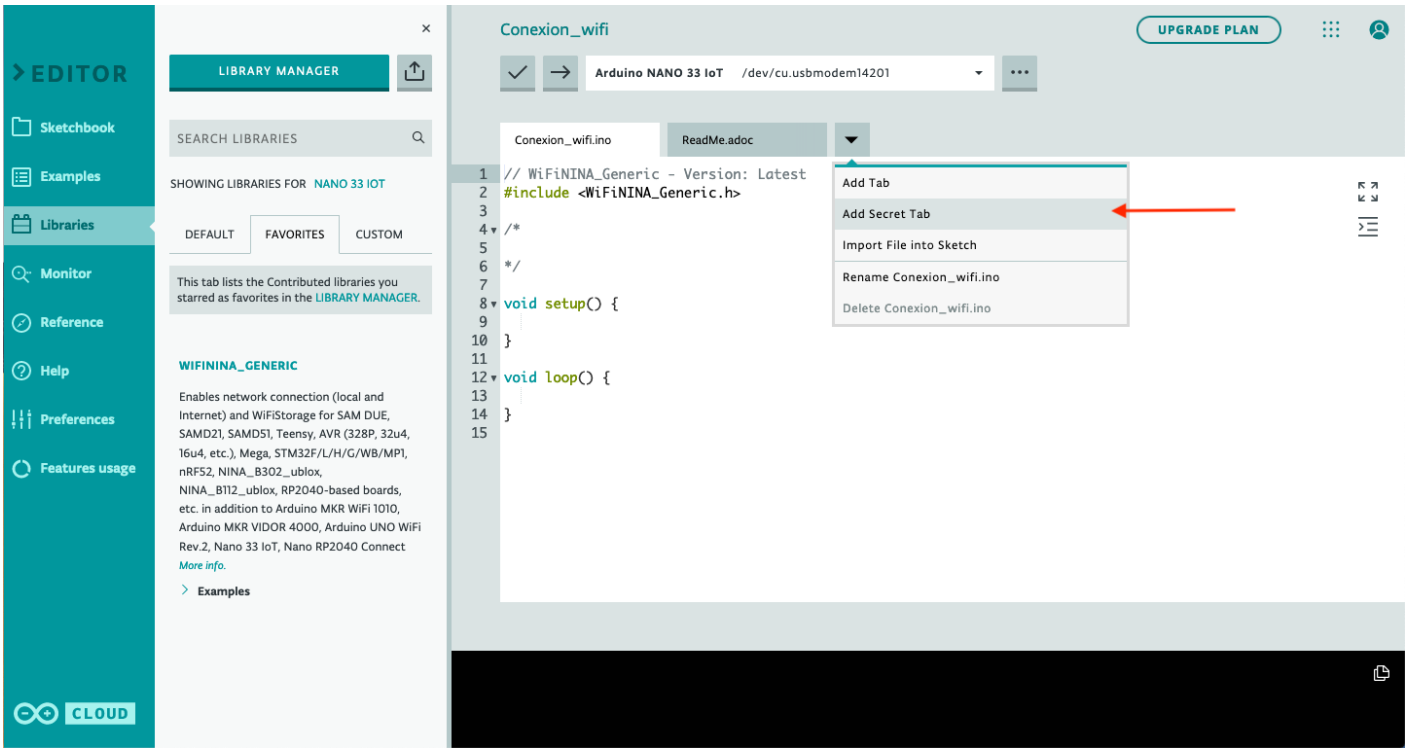
Conectaremos nuestro Arduino y nos aseguraremos de seleccionarlo en la barra horizontal en la que pone **--Select Board or Port--**:



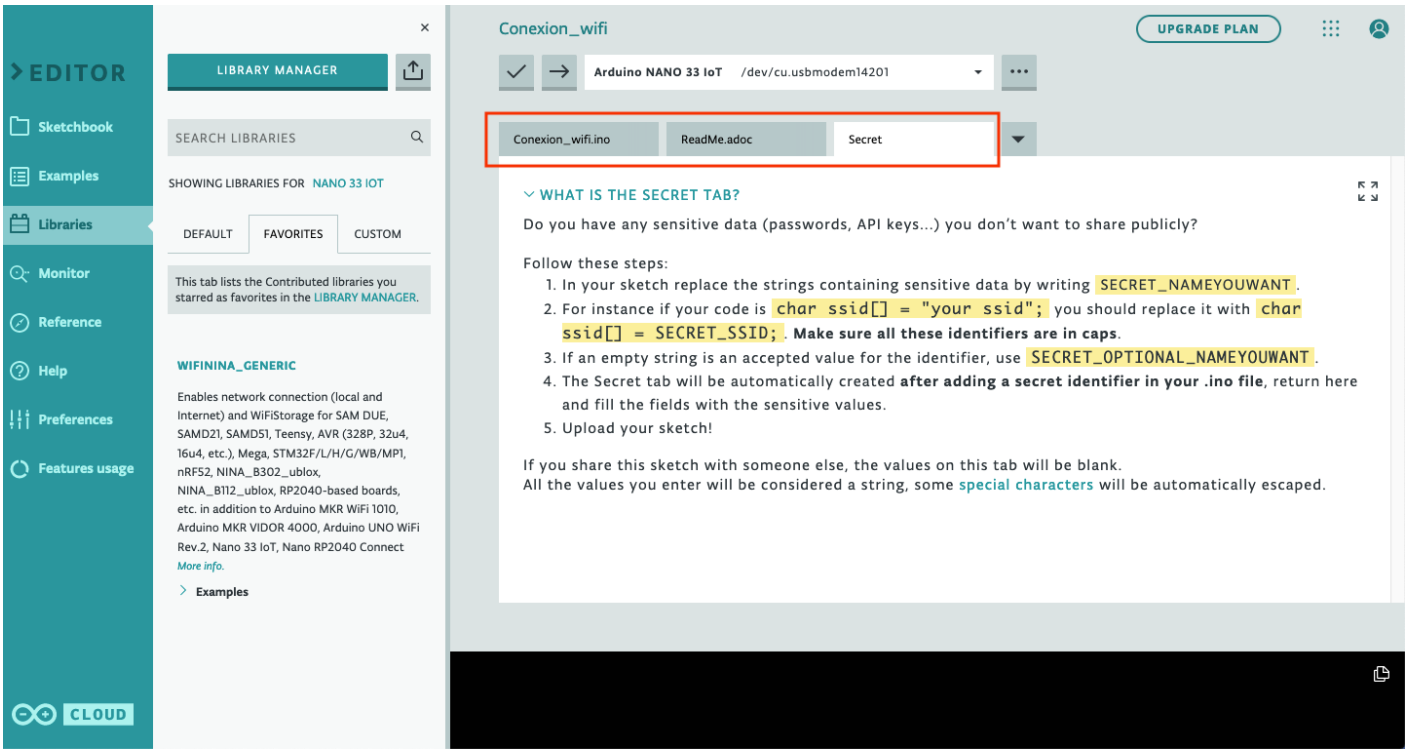


Creamos una Secret Tab

A continuación, crearemos una **Secret Tab** que se encargará de almacenar las credenciales de nuestra red Wi-Fi. Para ello, haremos click en el **triángulo negro hacia abaj**o que aparece en la parte superior de nuestro sketch, al lado de ReadMe.adoc:



Veremos que ahora tenemos 3 pestañas: Conexión_wifi.ino, ReadMe.adoc y Secret:



Añadimos las credenciales de nuestra red

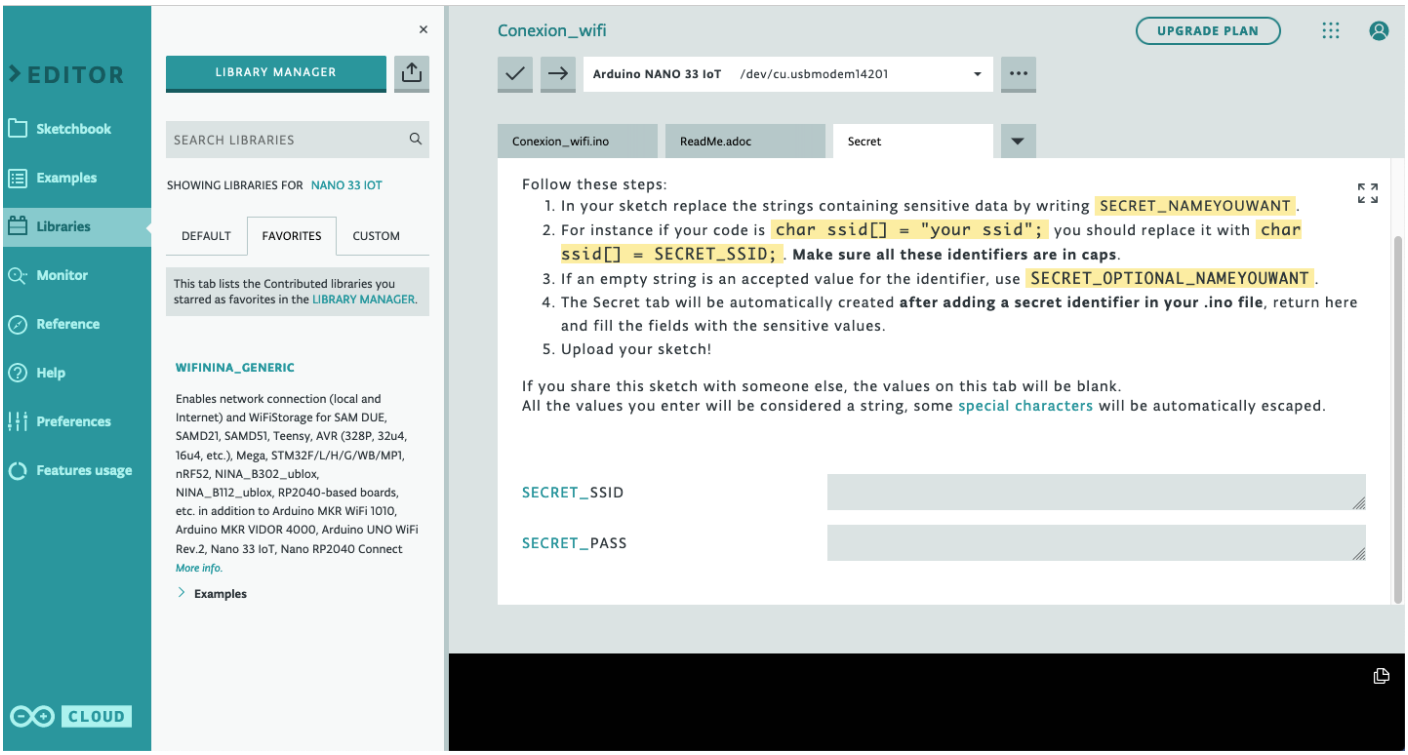
Volveremos a nuestra pestaña **Conexion_wifi.ino** y pegaremos lo siguiente:

```
//introducimos nuestras credenciales

char ssid[] = SECRET_SSID;                // SSID de la red(nombre)

char pass[] = SECRET_PASS;                // contraseña de la red
```

Si ahora volvemos a la pestaña Secret, veremos que aparecen dos campos nuevos en los cuales podemos escribir nuestro usuario y contraseña de la red WiFi. En **SECRET_SSID** escribiremos el nombre de la red y en **SECRET_PASS** la contraseña:



Veamos el código

Antes de la función setup()

Ahora que ya tenemos nuestra Secret Tab configurada, podemos volver a la pestaña **Conexion_wifi.ino** y ver el código necesario para que el LED de nuestro Arduino parpadee más rápido cuando la intensidad de la señal de nuestra red sea mayor.

Lo primero que necesitaremos será inicializar cinco variables para almacenar el estado de la radio Wi-Fi, el estado del LED y la hora de la última actualización.

```
int status = WL_IDLE_STATUS;              // estado de la radio Wi-Fi
int ledState = LOW;                        //Estado del LED (encendido o apagado)
unsigned long previousMillisInfo = 0;      //almacenará la última vez que se actualizó la
información del Wi-Fi
unsigned long previousMillisLED = 0;       // almacena la última vez que se actualizó el LED
const int intervalInfo = 5000;             // intervalo al que actualizaremos la información
```

En estas 5 variables encontramos distintos tipos, el **int** ya lo conocemos, es un n**úmero entero**, pero el **unsigned long** es nuevo. Lo que significa es que el valor que almacena es un entero que puede alcanzar un valor bastante alto, pero a diferencia de los long normales, los unsigned long solo pueden almacenar **valores positivos**.
En la línea 5, vemos que delante de int aparece la palabra **const**. Esto significa que este valor será **constante** y que, aunque queramos, no lo vamos a poder modificar.
Tu sketch debería tener el siguiente aspecto:

Conexion_wifi.ino | ReadMe.adoc | Secret

```
1  // WiFiNINA_Generic - Version: Latest
2  #include <WiFiNINA_Generic.h>
3
4  //please enter your sensitive data in the Secret tab
5
6  char ssid[] = SECRET_SSID;               // your network SSID (name)
7
8  char pass[] = SECRET_PASS;               // your network password (use for WPA, or use as key for WEP)
9
10 int status = WL_IDLE_STATUS;             // the Wi-Fi radio's status
11
12 int ledState = LOW;                      //ledState used to set the LED
13
14 unsigned long previousMillisInfo = 0;    //will store last time Wi-Fi information was updated
15
16 unsigned long previousMillisLED = 0;     // will store the last time LED was updated
17
18 const int intervalInfo = 5000;          // interval at which to update the board information
```

Dentro de la función setup()

En la función **setup()** iniciamos la comunicación en serie a **9600 baudios**, seguido de **while(!Serial)** que básicamente significa que a menos que abramos el **Monitor Serial** el programa no se ejecutará.

Luego usamos **pinMode()** para establecer el pin **LED_BUILTIN** como un **OUTPUT.**

Copyright 2026

```
void setup() {
  //Inicializa el puerto serial y lo abre
  Serial.begin(9600);
  while (!Serial);

  // Configuramos el LED como salida
  pinMode(LED_BUILTIN, OUTPUT);
```

A continuación seguiremos añadiendo líneas dentro de setup() Crearemos un bucle **while()** que comprobará si estamos conectados a la Wi-Fi. Si no lo estamos, intentará conectarnos a ella. Para ello utilizamos **status = WiFi.begin(ssid, pass)**;
para empezar a conectarnos a la Wi-Fi, y un retraso de 10 segundos para darle tiempo a conectarse:

```
// Intentamos conectar a red Wi-Fi
while (status != WL_CONNECTED) {
  Serial.print("Attempting to connect to network: ");
  Serial.println(ssid);

  // Conectamos a red WPA/WPA2:
  status = WiFi.begin(ssid, pass);

  // Esperamos 10 segundos para conectar
  delay(10000);
}
```

Al final de la función setup() utilizamos dos **Serial.println()** para imprimir en el monitor serie que, si todo va bien, ahora estamos conectados a la red y para separar el bloque de datos.

```
// Una vez estamos conectados, imprimimos la información
Serial.println("You're connected to the network");
Serial.println("---------------------------------------");
}
```

El aspecto final de la función setup() sería el siguiente:

Copyright 2026

Conexion_wifi.ino | ReadMe.adoc | Secret

```
21 void setup() {
22   //Initialize serial and wait for port to open:
23   Serial.begin(9600);
24   while (!Serial);
25
26   // set the LED as output
27   pinMode(LED_BUILTIN, OUTPUT);
28
29   // attempt to connect to Wi-Fi network:
30   while (status != WL_CONNECTED) {
31     Serial.print("Attempting to connect to network: ");
32     Serial.println(ssid);
33
34     // Connect to WPA/WPA2 network:
35     status = WiFi.begin(ssid, pass);
36
37     // wait 10 seconds for connection:
38     delay(10000);
39   }
40
41   // you're connected now, so print out the data:
42   Serial.println("You're connected to the network");
43   Serial.println("---------------------------------------");
44 }
```

Dentro de la función loop()

En la función **loop()** inicializaremos una nueva variable que almacenará el tiempo desde que el sketch comienza a ejecutarse. A continuación, utilizaremos un if() para comprobar si el tiempo después de la última actualización es mayor que el intervalo que hemos establecido antes, en las **5 variables del principio**.

```
void loop() {
  unsigned long currentMillisInfo = millis();


  // Comprueba si el tiempo que ha pasado desde la última actualización es mayor que el
intervalo fijado
if (currentMillisInfo - previousMillisInfo >= intervalInfo) {
```

Si entramos en el if, significa que ha llegado el momento de actualizar la información, así que lo primero que haremos dentro de la función **if()** es darle a la variable **previousMillisInfo** el valor del momento actual con **currentMillisInfo.** Luego imprimiremos tres tipos diferentes de información usando tres funciones que nos proporciona la librería WifiNINA:

La dirección IP de la placa usando la función **WiFi.localIP()**
Nombre de la red a la que está conectada usando la función **WiFi.SSID()**
Intensidad de la señal utilizando la función **WiFi.RSSI()**

```
previousMillisInfo = currentMillisInfo;

    Serial.println("Board Information:");
    // Imprime la dirección IP de nuestro Arduino:
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);



    // Imprime el SSID de nuestra Red:
    Serial.println();
    Serial.println("Network Information:");
    Serial.print("SSID: ");
    Serial.println(WiFi.SSID());



    // Imprime la fuerza de señal recibida:
    long rssi = WiFi.RSSI();
    Serial.print("signal strength (RSSI):");

    Serial.println(rssi);
    Serial.println("---------------------------------------");

  }
```

Después de esto, ya solo nos queda ocuparnos del LED. Necesitaremos inicializar otras dos variables, una para llevar la cuenta del tiempo del LED y otra para convertir la fuerza de la señal medida por la función WiFi.RSSI() a un intervalo de tiempo que será la velocidad a la que parpadeará nuestra lucecita.

```
unsigned long currentMillisLED = millis();


  // measure the signal strength and convert it into a time interval
  int intervalLED = WiFi.RSSI() * -10;
```

Por último, creamos otro if() para comprobar si el tiempo después del último parpadeo es mayor que el intervalo, de la misma manera que hicimos antes. Dentro de la sentencia if() añadimos una

nueva sentencia **else if()** para encender el LED si estaba apagado y viceversa, y una sentencia **digitalWrite()** para establecer el estado del LED.

```
// Comprueba si el tiempo desde el último parpadeo del LED es mayor que el intervalo fijado
  if (currentMillisLED - previousMillisLED >= intervalLED) {
    previousMillisLED = currentMillisLED;



    // si el LED está apagado, enciéndelo y viceversa
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(LED_BUILTIN, ledState);
  }
}
```

El aspecto de tu función loop() debería ser el siguiente:

```
void loop() {
  unsigned long currentMillisInfo = millis();

  // check if the time after the last update is bigger the interval
  if (currentMillisInfo - previousMillisInfo >= intervalInfo) {
    previousMillisInfo = currentMillisInfo;


    Serial.println("Board Information:");
    // print your board's IP address:
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);


    // print your network's SSID:
    Serial.println();
    Serial.println("Network Information:");
    Serial.print("SSID: ");

    Serial.println(WiFi.SSID());

    // print the received signal strength:
    long rssi = WiFi.RSSI();
    Serial.print("signal strength (RSSI):");
    Serial.println(rssi);
    Serial.println("----------------------------------------");
  }

  unsigned long currentMillisLED = millis();

  // measure the signal strength and convert it into a time interval
  int intervalLED = WiFi.RSSI() * -10;

  // check if the time after the last blink is bigger the interval
  if (currentMillisLED - previousMillisLED >= intervalLED) {
    previousMillisLED = currentMillisLED;

    // if the LED is off turn it on and vice-versa:

    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }


    // set the LED with the ledState of the variable:
    digitalWrite(LED_BUILTIN, ledState);
  }
}
```

De todas formas, aquí tiene todo el código en un bloque, por si lo prefieres copiar de una vez y evitar fallos:

```
// WiFiNINA_Generic - Version: Latest
#include <WiFiNINA_Generic.h>


//please enter your sensitive data in the Secret tab


char ssid[] = SECRET_SSID;                // your network SSID (name)
char pass[] = SECRET_PASS;                // your network password (use for WPA, or use as key
for WEP)
int status = WL_IDLE_STATUS;             // the Wi-Fi radio's status
int ledState = LOW;                       //ledState used to set the LED
unsigned long previousMillisInfo = 0;     //will store last time Wi-Fi information was updated
unsigned long previousMillisLED = 0;      // will store the last time LED was updated
```

```
const int intervalInfo = 5000;           // interval at which to update the board information


void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial);

  // set the LED as output
  pinMode(LED_BUILTIN, OUTPUT);

  // attempt to connect to Wi-Fi network:
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to network: ");
    Serial.println(ssid);

    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }

  // you're connected now, so print out the data:
  Serial.println("You're connected to the network");
  Serial.println("---------------------------------------");
}


void loop() {
  unsigned long currentMillisInfo = millis();

  // check if the time after the last update is bigger the interval
  if (currentMillisInfo - previousMillisInfo >= intervalInfo) {
    previousMillisInfo = currentMillisInfo;
```

```
  Serial.println("Board Information:");
  // print your board's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);


  // print your network's SSID:
  Serial.println();
  Serial.println("Network Information:");
  Serial.print("SSID: ");

  Serial.println(WiFi.SSID());

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.println(rssi);
  Serial.println("---------------------------------------");
}

unsigned long currentMillisLED = millis();

// measure the signal strength and convert it into a time interval
int intervalLED = WiFi.RSSI() * -10;

// check if the time after the last blink is bigger the interval
if (currentMillisLED - previousMillisLED >= intervalLED) {
  previousMillisLED = currentMillisLED;

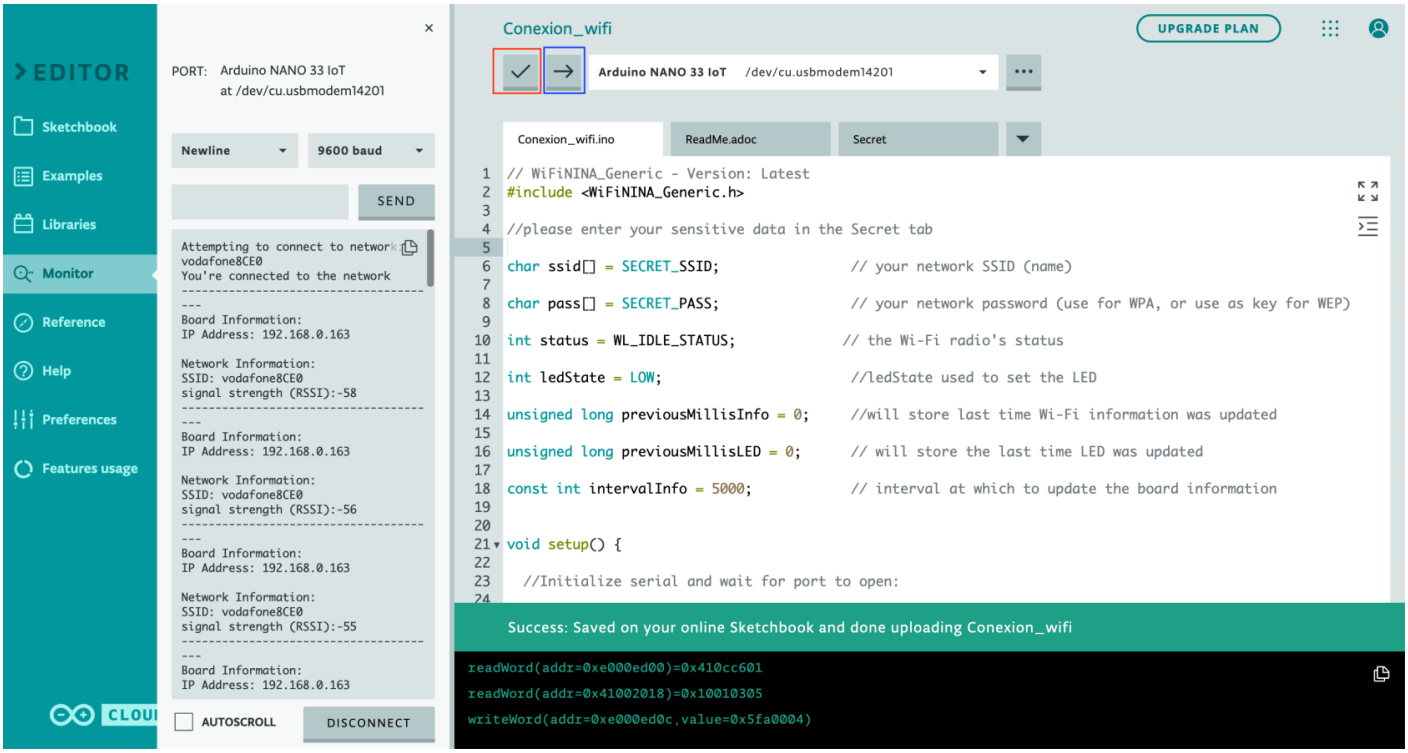
  // if the LED is off turn it on and vice-versa:

  if (ledState == LOW) {
    ledState = HIGH;
  } else {
```

```
      ledState = LOW;
    }

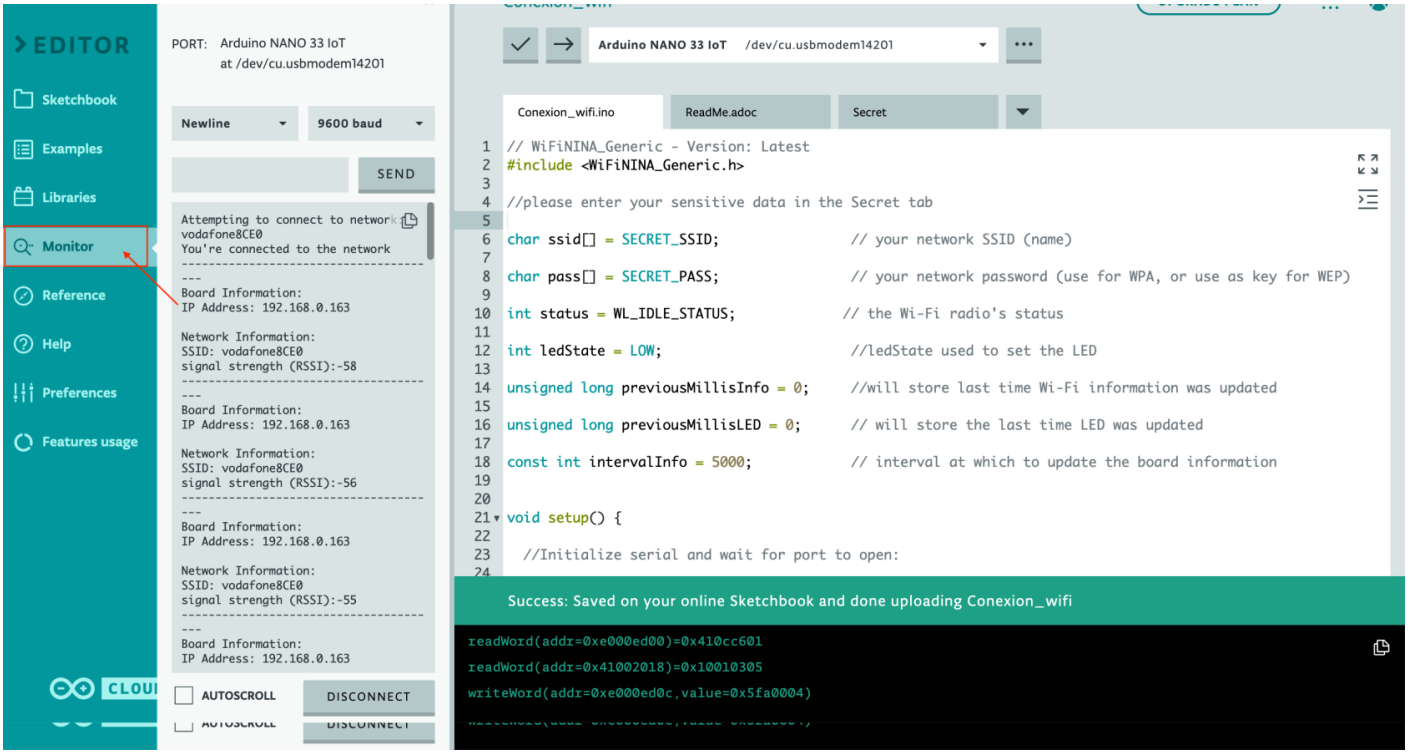

    // set the LED with the ledState of the variable:
    digitalWrite(LED_BUILTIN, ledState);
  }
}
```

Una vez hayamos copiado el código a nuestro sketch, habrá llegado el momento de verificar que todo está correcto. En este caso, **si hemos seguido todos los pasos**, no nos debería dar ningún problema, pero cuando escribamos nuestro propio código, podemos obtener errores. Para verificar el código, haremos click sobre el **símbolo tick**, que aparece en la siguiente imagen dentro de un **recuadro rojo**. Una vez lo hayamos verificado, nuestro código estará listo para ser subido a nuestro Arduino. Así que, haremos click sobre la **flecha hacia la derecha**, que en la imagen aparece dentro de un **recuadro azul**:



Nuestro Arduino puede tardar unos segundos (a veces algún minuto) en subir el código y cuando lo haga veremos el mensaje '**Sucess:...**' dentro del cuadrado verde, en la zona de la consola de nuestro sketch.

Para ver los diferentes valores que obtenemos necesitaremos abrir el puerto serie de nuestro Arduino. Para ello iremos a la columna de la izquierda y haremos click sobre **Monitor**. Veremos que en la columna justo a la derecha se comenzarán a actualizar valores y aparecerá información como el nombre de la red a la que estamos conectados y la fuerza de la señal que recibimos.



Si estamos trabajando con un ordenador portátil, puedes experimentar acercándote con él y tu Arduino hacia tu router. Verás como la luz de Arduino parpadea más rápido y también los valores que aparecen a la derecha de **signal strength** aumentan.

¿Qué tenemos que entregar?

Graba un video en el que primero se vea la pantalla con el monitor serie abierto y cómo se actualiza la información sobre la intensidad con la que se recibe la señal y luego muevas la cámara para enfocar tu NANO con el LED parpadeando.

Para enviarlo puedes mandar el video en formato gif o subirlo a Youtube o a Drive, configurarlo como oculto solo para aquellos que tengan el enlace, y compartir el enlace en la tarea correspondiente del moodle del curso.

FUENTES:

Conectar Arduino a WiFi: https://docs.arduino.cc/tutorials/nano-33-iot/wifi-connection

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



Revision #21
Created 2022-09-15 13:36:50 CEST by Marta P. Campos
Updated 2023-01-17 16:09:05 CET by Equipo CATEDU

Copyright 2026