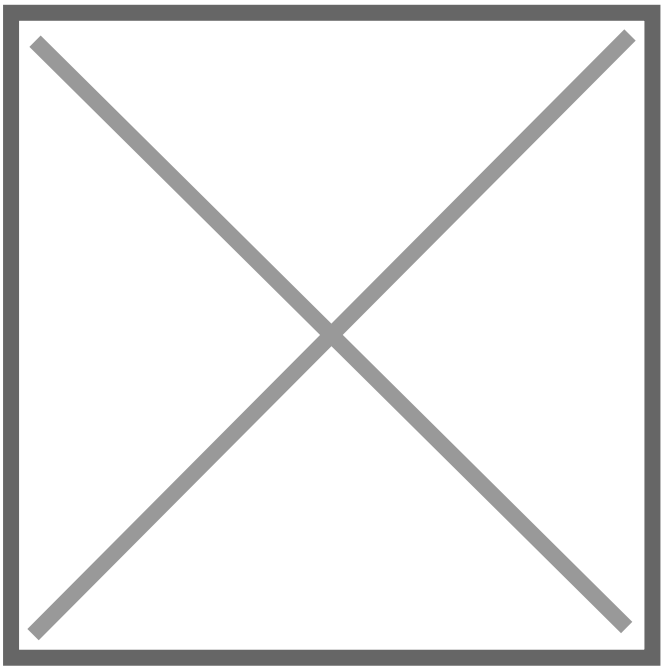
3. Nos conectamos a Internet

El internet de las cosas (IoT)

> El **Internet de las cosas** (Internet of Thing IoT) describe objetos físicos —o grupos de estos— con sensores, capacidad de procesamiento, software y otras tecnologías que se conectan e intercambian datos con otros dispositivos y sistemas a través de internet u otras redes de comunicación. El Internet de las cosas se ha considerado un término erróneo porque los dispositivos no necesitan estar conectados a la Internet pública. Sólo necesitan estar conectadas a una red y ser direccionables individualmente

Fuente Wikipedia IoT Internet de las cosas CC-BY-SA

En nuestro caso, lo que vamos a hacer es conectar nuestro Arduino a Internet y ver cómo mandamos la información desde nuestra placa a la plataforma Arduino IoT. Veremos cómo nuestro Arduino Nano 33 IoT nos permite conectarnos a una red wifi, que en nuestro caso sí que va a estar conectada a Internet, pero también podríamos utilizar nuestro Arduino para conectarlo a una red local que no necesariamente estaría conectada a Internet.



De Drawed by Wilgengebroed on FlickrTranslated by Prades97 CC BY-SA 3.0

Conectamos objetos entre sí

Aunque en este curso no vamos a conectar diferentes objetos entre sí, está bien conocer que el Internet de las cosas se creó originalmente con esa intención.

Estamos hablando de dispositivos que se conectan a internet de forma desatendida, por vía hardware (o mejor dicho firmware) a diferencia de un ordenador, tablet o móvil, donde tienes que configurar por software el dispositivo y hay un diálogo entre usuario y dispositivo sobre el uso de Internet (el software solicita tal página web, tales datos etc por voluntad del usuario o por diálogo con el usuario) Aquí los dispositivos están ya configurados de los datos que se comunican. Es decir "conectar y olvidar".

Piensa en la diferencia entre un enchufe inteligente y un ordenador, el primero es lo que se considera dentro de IoT

FUENTES:

IoT: https://libros.catedu.es/books/rover-marciano-con-arduinoblocks-e-internet-de-las-cosas-iot/page/internet-de-las-cosas-iot

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

logo.png

Arduino IoT Cloud

image-1664198884282.jpg

Si queremos conectar nuestro Arduino a Internet para enviar información, una de las maneras más sencillas es empleando nuestro Arduino Nano 33 IoT y la plataforma online Arduino IoT Cloud.

Los usos que se le pueden dar a esta plataforma son:

- **Monitorización** de datos: es posible monitorizar fácilmente los valores de los sensores de Arduino a través de un panel de control.
- **Sincronización** de variables: la sincronización de variables nos permite sincronizar variables entre dispositivos, permitiendo la comunicación entre dispositivos con una codificación mínima.
- **Programación**: podemos programar trabajos para que se activen/desactiven diferentes funciones durante una cantidad de tiempo específica (segundos, minutos, horas).
- Cargas Over-The-Air (OTA): permite cargar nuestro código a dispositivos no conectados al ordenador.
- Webhooks: integra tu proyecto con otro servicio, como IFTTT.
- Soporte de Amazon Alexa: permite que nuestro proyecto sea controlado por voz con la integración de Amazon.
- Dashboard Sharing: comparte tus datos con otras personas de todo el mundo.

> Si quieres echarle un vistazo a la plataforma IoT Cloud, puedes hacerlo en este enlace.

Una de las ventajas principales con las que cuenta esta plataforma es que al haber sido creada desde Arduino, existe mucha documentación y tutoriales sobre ella, lo que nos va a facilitar su uso.

En nuestro caso, vamos a enviar información a esta plataforma y gracias a ella produciremos ciertos cambios como hacer sonar una alarma. Aunque primero, tendremos que conectar nuestro Arduino a internet...

¿Qué modelos de placa Arduino pueden usar IoT Cloud?

En realidad, cualquier placa que esté basada en el microcontrolador ESP32/ESP8266 puede conectarse, ya que es necesario utilizar Wi-Fi. Nuestro Nano 33 IoT cuenta con una antena

integrada que nos permite la conexión, lo veremos un poco más adelante, cuando la conectemos.

En la próxima página pasaremos a ver algunos proyectos que emplean Arduino y el internet de las cosas y que nos puede resultar interesante conocer.

FUENTES:

Arduino IoT Cloud: https://docs.arduino.cc/arduino-cloud/getting-started/iot-cloud-getting-started

Foto de C Dustin en Unsplash.

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

logo.png

Proyectos que usan IoT y Arduino

A continuación, vamos a ver algunos proyectos a modo de ejemplo, que nos pueden dar ideas de lo que podríamos hacer en el aula con nuestro Arduino y la plataforma IoT Cloud. Se ha intentado crear una selección de proyectos que abarquen tanto a aquellos que tendrían cabida dentro del mundo del arte y el diseño, como otros más centrados en el ámbito educativo.

iNecklace

image-1664223878196.24.24.png

Este primer proyecto consiste en un collar que se ilumina según los colores que configuramos desde nuestro ordenador. En este proyecto hay varios aspectos interesantes que me gustaría comentarte.

El primero de ellos es la manera en la que está creado el vestible. Para conectar los LED a Arduino se han utilizado cables de cobre esmaltado enrollados en lana, como podemos ver en esta imagen:

image-1664225928034.45.12.png

En cuanto al software, hace uso de un repositorio para poder conectar Arduino a la misma red a la que tenemos conectado nuestro ordenador y también para controlar a los LED que en este caso, al cambiar de color, requieren de una librería especial. Aparte de ello, existe una aplicación, programada por sus creadores, pero que ponen a disposición de cualquiera, que es la que directamente controla el color de cada uno de los LED.

En la página de Instructables, una web en la que podrás encontrar gran cantidad de proyectos basados en Arduino, los creadores de iNecklace han creado un tutorial sobre este proyecto, pero requiere de unos conocimientos de Arduino más avanzados que lo que se va a ver en este curso. Este punto es importante, ya que veremos que muchos de los proyectos creados con Arduino cuentan con tutoriales para **facilitar que cualquiera los pueda replicar** o reutilizar realizando modificaciones.

Brazo robótico con materiales reciclados

image-1664229062756.gif

Este segundo proyecto propone el uso de materiales que mayormente podrían considerarse como de desecho, como son una botella de plástico y trozos de madera, para construir un brazo robótico. En este proyecto, los creadores no han utilizado el internet de las cosas, sino que el brazo lo controlan empleando un joystick. No obstante, utilizando otro Arduino (en este caso usan un Arduino Uno) como por ejemplo nuestro Arduino Nano 33 IoT podríamos conectar nuestro brazo robótico a una red Wi-Fi y controlarlo remotamente desde un ordenador.

Lo más interesante de este proyecto es cómo con muy pocos materiales consiguen un buen resultado. Como ya he comentado, para controlar el movimiento emplean un joystick y el movimiento lo realizan unos motores servo, sobre los cuales ya hemos hablado en el apartado sobre actuadores.

Puedes ver el tutorial del proyecto en el siguiente enlace.

Comunicador humano-planta

image-1664266743899.17.50.png

Con ese llamativo título parece que vayamos a ser capaces de mantener una conversación con nuestras plantas, pero en realidad, de lo que trata este proyecto es de que seamos capaces de monitorizar ciertos aspectos, como si la planta nos dijese que está bien o que le falta algo. Para ello, se utilizan **sensores que controlan la cantidad de humedad** y **luz**, así como la **temperatura**.

Por ejemplo, la cantidad de luz es medida con un sensor como el que hemos utilizado en una de las prácticas del bloque anterior. Este proyecto está realizado con una placa de Arduino diferente a la que nosotros empleamos, pero también con la capacidad de conectarse a una red Wi-Fi para que, al igual que el nuestro, pueda comunicarse fácilmente con otros dispositivos.

Una de las librerías empleadas en este proyecto es la que controla que cada día nuestra planta nos envíe un email compartiéndonos 'cómo está' a una determinada hora. Esto lo veremos en un apartado posterior, pero de momento te puedo decir que es posible hacerlo gracias a que nuestro Arduino tiene un reloj a tiempo real.

Puedes echarle un vistazo al tutorial de este proyecto aquí.

Almohada ILoveYou

image-1664350784700.39.18.png

No es difícil adivinar cuál es el propósito de este proyecto: enviar abrazos a distancia. Si recordáis la sección en la que vimos una selección de proyectos dentro del arte y el diseño, uno de ellos era

un proyecto que nos permitía sentir el sonido, la *Sound Shirt*. Pues bien, sus creadores también cuentan entre sus diseños con la *Hug Shirt*, la cual realiza una función parecida a la de nuestra almohada. La diferencia de base entre ambos proyectos es que el primero es de código abierto, mientras que el segundo es una prenda de vestir comercializable.

Otra diferencia principal es que nuestra almohada cuenta con un circuito más sencillo que el de la camiseta. Como podemos ver en este tutorial. Aparte de un Arduino, utilizan un **zumbador**, **papel de aluminio**, **cables**, una **resistencia** y una **pila**.
En el área del software emplea la aplicación de mensajería instantánea Telegram para crear un bot, que se encargará de enviar nuestro amor, con la forma del emoji de un corazón □□, a quien escojamos.

> El papel de aluminio lo utilizan para crear un sensor capacitivo el cual genera un campo eléctrico que es modificado al aproximarse a él un objeto que conduzca la electricidad, en nuestro caso nuestro cuerpo al abrazar el cojín. Este principio es el que se usa en las pantallas táctiles, por ejemplo. Aquí tienes un ejemplo.

Por otro lado, el zumbador lo emplean para crear la sensación de un corazón latiendo. El tutorial de este proyecto puedes verlo al completo aquí.

FUENTES:

iNecklace: https://github.com/Lilooo/iNecklace

Brazo robot con materiales reciclados: https://create.arduino.cc/projecthub/circuito-io-team/robotic-arm-from-recycled-materials-7e318a/

Comunicador humano-planta: https://create.arduino.cc/projecthub/arduino/plant-communicator-7ea06f?ref=tag&ref_id=iot&offset=6

Almohada ILoveYou: https://create.arduino.cc/projecthub/arduino/love-you-pillow-f08931

Sensor capacitivo y Arduino: https://www.aranacorp.com/es/creando-un-sensor-capacitivo-con-arduino/

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

logo.png

¡Vamos a conectarnos! (Parte 1)

Llegados a este punto, veamos cómo conectar nuestro Nano 33 IoT al IoT Cloud.

Creamos una cuenta

image-1664353472710.23.45.png

El primer paso, y además obligatorio, es crearnos una cuenta dentro de la web de Arduino. Esto es lógico, ya que tendremos que guardar la información que recopilemos con los sensores de nuestro Arduino y enviarla a esta plataforma. El proceso es similar a cuando tenemos que crearnos una cuenta en cualquier plataforma. Nos irán pidiendo diferentes datos como nuestra fecha de nacimiento, nombre de usuario, email, etc; así que, vamos a omitir este paso en el tutorial.

También te da la opción de conectarte usando tus credenciales de Google, GitHub, Facebook o Apple.

Si ya tienes una cuenta con Arduino, puedes saltarte este paso y acceder directamente usando tu nombre de usuario y contraseña.

FUENTES:

https://docs.arduino.cc/arduino-cloud/getting-started/iot-cloud-getting-started

Instalar Create Agent: https://support.arduino.cc/hc/en-us/articles/360014869820-Install-the-Arduino-Create-Agent

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

logo.png

¡Vamos a conectarnos! (Parte 2)

Accedemos a IoT Cloud

Una vez hemos creado nuestra cuenta, ya podemos entrar y veremos algo parecido a esto. Para acceder a IoT Cloud, clicaremos arriba a la derecha

image-1667980487757.png

Una vez dentro de IoT Cloud accederemos a la siguiente pantalla:

image-1664353733531.27.57.png

En la barra superior puedes ver 5 apartados diferentes: **Things, Dashboards, Devices, Integrations y Templates.** De momento, vamos a comenzar con el primero: **Things**. Dentro de este apartado incluiremos todas las *cosas* que vamos a conectar a IoT Cloud. Así que para poder conectar nuestro Arduino, lo primero que tenemos que crear es una thing, una *cosa*.

Obviamente, si no hemos utilizado antes IoT Cloud no tendremos nada creado y veremos algo parecido a esto:

image-1664354224346.33.41.png

Dentro de los cuadrados rojos tenemos las partes más importantes de nuestra *thing*:

- **Variables:** En este apartado incluiremos todas las variables que queramos monitorizar.
- **Device:** aquí configuraremos nuestro dispositivo para que se comunique con IoT Cloud.
- **Network:** para conectar nuestro dispositivo a Wi-Fi necesitaremos introducir el nombre y contraseña de nuestra red.

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

logo.png

¡Vamos a conectarnos! (Parte 3)

Instalamos Arduino Create Agent

Para poder conectar nuestro dispositivo a IoT Cloud, vamos a tener que instalar un programita adicional, el Arduino Create Agent. Lo que nos va a permitir es que nuestro Arduino sea reconocido por la plataforma, podremos subir los programas que escribamos y podremos ver la información transmitida por el puerto serie directamente.

image-1664355406473.56.37.png

MAC OS

Haremos click en START y nos aparecerá la siguiente pantalla en la cual ya podremos descargar la aplicación:

image-1664355745999.02.16.png

Una vez se haya descargado, iremos a la carpeta de Descargas y haremos doble click en el instalador. Una vez ahí, seguiremos las instrucciones.

Si vas a usar un navegador que no sea Google Chrome o Mozilla Firefox, asegúrate de responder **Sí** en el paso de **compatibilidad con el navegador (Browser Support)**.

Una vez lo hayamos instalado, haremos click en siguiente (NEXT) y veremos el icono de Arduino en la parte superior de nuestro ordenador:

image-1664356076435.07.48.png

LINUX

El proceso es muy similar, también tendremos que descargar la aplicación.

image-1664873780692.png

Nos descargará un ejecutable comprimido en tar.gz (formato de Linux) y al hacer doble clic sobre él, el asistente nos guiará para su instalación.

image-1664873869063.png

Si todo ha ido bien, veremos al final una pantalla como esta:

image-1664873929385.png

WINDOWS

El proceso es muy similar, también tendremos que descargar la aplicación, solo que en este caso nos dará la opción de descargarlo para Win32 o Win64, dependiendo de la arquitectura de nuestro ordenador. Si no lo sabemos, nos aparecerá marcada en verde oscuro la opción correcta:

image-1664356686136.12.42.png

Una vez se haya descargado, iremos a la carpeta de Descargas y haremos doble click en el instalador.
Una vez ahí, seguiremos las instrucciones. Si vas a usar un navegador que no sea Google Chrome o Mozilla Firefox, asegúrate de responder **Sí** en el paso de **compatibilidad con el navegador (Browser Support)**.

En Windows, puede que recibas una advertencia de seguridad y se te pedirá que apruebes la instalación de un certificado raíz (root) de Arduino. Elige **Sí** para instalar el certificado y continúa con la instalación.

Si la instalación ha ido bien, verás esta pantalla:

image-1664356906559.13.48.png

¿Cómo comprobamos que Create Agent funciona?

Para asegurarnos de que Create Agent no está en pausa, buscaremos el **icono del Create Agent** image-1664359383634.png en la parte superior derecha de la barra de menús (macOS) o en la parte inferior derecha de la barra de tareas (Windows, Linux), normalmente dentro de la bandeja del sistema.

image-1667295740215.png

Si hace click en él y la opción **Reanudar Agente** (Resume Agent) está disponible, el agente está actualmente en pausa. Haga clic en **Reanudar Agente** para reanudar.

Cuando esté funcionando, nos dará la opción de pausarlo haciendo click en **Pause Agent**:

image-1664359553180.05.31.png

¿Qué problemas podemos encontrarnos?

- Tenemos que asegurarnos de tener permiso de administrador; si el ordenador que estás utilizando no es tu propio ordenador personal, pide a tu administrador de TI un perfil

actualizado.
- A veces la instalación del agente puede ser bloqueada por el cortafuegos (firewall) o el software antivirus; en este último caso, puedes intentar añadir el agente a tu lista blanca de antivirus.

FUENTES:

Instalar Create Agent: https://support.arduino.cc/hc/en-us/articles/360014869820-Install-the-Arduino-Create-Agent

Cómo comprobar si Create Agent está funcionando: https://support.arduino.cc/hc/en-us/articles/4980687506844-Check-if-the-Arduino-Create-Agent-is-installed-and-running

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

logo.png

¡Vamos a conectarnos! (Parte 4)

Configuramos nuestro dispositivo

El proceso de configuración es rápido y sencillo (bueno, eso nos dicen desde Arduino, pero ya veremos...), y se puede realizar pulsando el botón "Seleccionar dispositivo" (**Select Device**) en la vista general de Thing. Cuando ya tengamos placas vinculadas, podremos elegir entre cualquier placa que haya sido configurada, o seleccionar la opción "Configurar nuevo dispositivo".

image-1664357528875.33.41 copia.png

Una vez ahí nos dará dos opciones:

image-1664357687399.34.11.png

Escogeremos la primera y nos pedirá que conectemos nuestro dispositivo, en este caso el Arduino Nano 33 IoT, al ordenador. Es aconsejable que si tenemos otros dispositivos conectados a nuestro ordenador a través de algún puerto USB, los desconectemos: image-1664359704088.34.16.png
Una vez lo hayamos conectado a uno de nuestros puertos USB, tendremos que tener bastante paciencia. ¿Por qué? Bueno, en este punto nos pueden pasar varias cosas:

1. Esta pantalla nos puede dejar varios minutos esperando hasta decirnos si nuestro dispositivo ha sido reconocido o no, y después de haber esperado un rato, puede que nos diga que el dispositivo no ha sido reconocido. No pasa nada, lo que haremos será **desconectar nuestro Arduino**, **cerrar** la ventana emergente de **Setup device** y repetir el proceso anterior.

2. Si a la segunda no nos funciona, lo intentaremos una tercera vez.

3. Si así tampoco, lo conectaremos a otro puerto USB y repetiremos el proceso.

Finalmente obtendremos esto:

image-1664360794324.19.25.png

Haremos click en **CONFIGURE** y la próxima ventana emergente nos mostrará nuestro Arduino, al que le habrán asignado un nuevo nombre. Nosotros podremos cambiarlo por otro si queremos, aunque no es necesario:image-1664360945164.28.56.png

Una vez tengamos el nombre, haremos click en NEXT y comenzará un proceso que, al igual que en el paso Setup Device, nos puede llevar varios minutos y acabar generándonos errores, por lo que

es posible que tengamos que repetir el proceso varias veces. Así que, paciencia.

Veremos esta ventana emergente:

image-1664361272781.30.56.png

Nos dicen que en el primer paso van a subir un programita a nuestro Arduino que hará que sea configurado para comunicarse de manera segura con IoT Cloud. Nos avisan de que puede costar hasta 5 minutos, pero puedo asegurarte que, en ocasiones, puede alargarse unos cuantos minutos más.

Una vez el sketch haya sido subido a nuestro Arduino, pasará al paso 2:

image-1664361385178.31.06.pngEn ese paso se comprobará la conectividad, y es el paso que nos puede dar errores... haciendo que tengamos que desconectar nuestro Arduino y volver a conectar. Sobre todo, NO DESCONECTES EL ARDUINO SI NO TE HA DICHO QUE HA HABIDO UN ERROR, porque es probable que el proceso esté funcionando correctamente, solo que por problemas de conexión esté tardando más tiempo del que se consideraría normal. Por eso, te pido, PACIENCIA.

Y una vez lo hayamos conseguido:

image-1664361513779.31.27.png

Y ya, por fin, nos aparecerá nuestro Arduino en la parte derecha como dispositivo asociado (**Associated Device**) a nuestra cuenta de IoT Cloud:

image-1664361590471.38.57.png

Si ahora vamos al apartado **Devices (Dispositivos),** veremos que ya nos aparece nuestro Nano 33 IoT. Aunque esté conectado, pondrá que está Offline, no te preocupes, eso ya lo veremos más adelante:

image-1664361819359.42.51.png

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

logo.png

¡Vamos a conectarnos! (Parte 5)

Ahora, necesitaremos proporcionar nuestro nombre y clave de red Wi-Fi para poder conectar nuestro proyecto a Internet. Estas credenciales se añadirán automáticamente al programa que cargaremos en nuestro Arduino, de eso no tendremos que preocuparnos nosotros, se encarga IoT Cloud.

Una vez hagamos click en **Configure** del apartado **Network**:

image-1664362138639.33.41 copia 2.png

Nos aparecerá esta ventana emergente, en la que introduciremos los datos:

image-1664362040960.44.54.pngEste proceso no debería generarnos ningún error, a no ser que no hayas escrito correctamente alguno de los dos campos. Asegúrate y revísalos si te da algún problema.

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

logo.png

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?

image-1664438083182.jpg

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.

> 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:

image-1664440758397.36.41 copia.png

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...** :

image-1664440931565.40.48.png

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:**

image-1664449474236.23.02 copia.png

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:

image-1664449809946.23.02 copia 2.png

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

image-1664449912028.34.15.png

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:

image-1664450110827.34.30.png

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:

image-1664450307208.34.39 copia.png

> 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--**:

image-1664611413284.37.24.png

image-1664612357504.37.30.png

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:

image-1664612435327.38.23.png

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

image-1664612678054.38.35.png

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:

image-1664612721563.41.30.png

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:

image-1664613843173.43.37.png

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.**

```
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:

image-1664615802087.14.56.png

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:
image-1664617745544.48.34.png

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**:

image-1665041823626.49.19 copia.png

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.

image-1665042301441.49.19 copia 2.png

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

logo.png

Tic tac, tic tac

Ahora que ya hemos conectado nuestro Arduino a internet, iremos un paso más allá y conectaremos nuestro Arduino a IoT Cloud para crear una alarma que funcione con el reloj interno de nuestro Arduino. Esta alarma lo que hará es encender un LED virtual cuando sea la hora que nosotras decidamos. No vamos a necesitar material adicional únicamente nuestro NANO 33 IoT y el cable USB para conectarlo a nuestro ordenador.

RTC: el reloj de nuestro Arduino

Un reloj en tiempo real es un reloj que lleva la cuenta de la hora actual y puede utilizarse para programar acciones específicas en un momento determinado. El término **reloj en tiempo real (RTC)** se utiliza para evitar la confusión con los **relojes de hardware ordinarios**, que son sólo señales que gobiernan la electrónica digital, y no son usados para contar en horas, minutos y segundos, sino que se utilizan para el funcionamiento interno del propio componente electrónico.

La mayoría de los RTC (Real Time Clock) utilizan un oscilador de cristal cuya frecuencia es de 32,768 KHz (la misma frecuencia utilizada en los relojes de cuarzo). Además, el RTC sigue funcionando en modo de reposo, por lo que se puede utilizar para despertar el dispositivo de los modos de reposo de forma programada. Cada vez que se alimenta la placa, el RTC se resetea y comienza desde la fecha estándar. Para mantener la hora y el RTC en funcionamiento es necesario mantener el dispositivo alimentado.

Para utilizar este reloj, usaremos **la librería RTCZero**. Esta librería nos permite controlar y utilizar el RTC interno en todas las placas Arduino con la arquitectura SAMD (como la que tiene nuestro NANO 33 IoT). La librería se encarga de la inicialización del RTC interno e incluye varias funciones que utilizaremos en esta práctica.

> Para saber más sobre esta librería, haz click aquí.

Primer paso: Creamos nuestra primera Thing

Para poder conectar a nuestro NANO 33 IoT, tendremos que crear un componente virtual llamado 'Thing'. Estas 'cosas' son las que permiten que nuestros proyectos se comuniquen con IoT Cloud. Como dicen en la imagen siguiente, estas Thing pueden también comunicarse entre ellas, aunque por el momento, nosotros solo crearemos una de ellas. Para crearla, iremos al menú horizontal superior (Things, Dashboards, Devices, Integrations and Templates) y haremos click donde pone **Things**.

image-1665055346267.18.27.png

Renombramos nuestra thing como **Nano Alarm**, aunque puedes escoger otro nombre si lo prefieres, te aconsejo que sea descriptivo, haciendo click sobre el rectángulo en el que pone **Untitled**:

image-1665055416726.18.41 copia.png

Una vez hemos cambiado el nombre, el siguiente paso será vincular un dispositivo a nuestra Thing. En este punto, conectaremos nuestro Arduino al ordenador por **USB.** Ahora, haremos click en el apartado **Associated Device** y ahí en **Select Device.**

image-1665055483548.18.41 copia 2.png

Como ya lo habíamos asociado antes, nos debería aparecer esta pantalla, en la que IoT Cloud ya lo reconoce, si no, seguiremos los pasos que nos indique y que serán parecidos a los que ya seguimos:

image-1665055511230.19.08.png

Segundo paso: creamos nuestras variables

Cuando hayamos configurado nuestro NANO 33 IoT, nos aparecerá en la columna de la derecha, como en la imagen inferior.

> Aunque en esta imagen aparezca online, por el momento **no te preocupes** si el tuyo aparece **offline**.

Para añadir nuestras variables, haremos click en **ADD VARIABLE.** Esto abrirá una nueva ventana, donde podemos elegir el nombre, el tipo de datos, la configuración de permisos y la configuración de actualización.

image-1665055900397.19.17 copia.png

Nos aparecerá la siguiente ventana:

image-1665055988275.32.35.png

En ella introduciremos la información de nuestra primera variable:
Name: **alarm_state**
Select variable type: **boolean**
Variable Permission: **Read & Write**

Variable Update Policy: **On change**

image-1665056132088.34.40.png

Le daremos a **ADD VARIABLE** y nos aparecerá así:

image-1665056261340.37.03.png

Haremos lo mismo con las siguientes variables:

Nombre	Tipo de datos	Permiso de la variable	Uso
alarm_state	boolean	Read & Write	Activa o desactiva la alarma
hours_alarm	int	Read & Write	Establece la hora de la alarma
minutes_alarm	int	Read & Write	Establece los minutos de la alarma
seconds_alarm	int	Read & Write	Establece los segundos de la alarma
led	boolean	Read Only	Indica si la alarma está activada o no

> El tipo de datos **int** nos aparecerá como **Integer Numbe**r:

image-1665056843240.39.18 copia.png

Una vez hayamos creado todas las variables, nos aparecerán como en la siguiente imagen:

image-1665057437084.56.57.png

> El orden en que nos aparezcan puede variar, pero eso no es problema. Lo importante es que alarm_state y led tengan valores boolean y que tanto hours_alarm, minutes_alarm y seconds_alarm tengan valores integer.

Tercer paso: introducimos las credenciales de nuestro Wi-Fi

No debemos olvidar introducir las credenciales de nuestra red. Ese proceso ya te lo he explicado en este apartado.

Una vez hayamos introducido el nombre y contraseña de nuestro Wi-Fi, haremos click sobre el apartado **Dashboards,** justo a la derecha de **Thing**.

Cuarto paso: creamos nuestro dashboard

Una vez hagamos click sobre **Dashboards**, veremos la siguiente pantalla, en la que pulsaremos sobre **BUILD DASHBOARD**:

image-1665057185882.51.34.png

Es el momento de cambiar el nombre de nuestro dashboard "**Untitled**" por **Nano Alarm** y empezar a añadir los diferentes widgets que necesitaremos:

image-1665095405976.54.06.png

Para ello hacemos clic en el botón **ADD,** en la esquina superior izquierda y nos aparecerá esta ventana emergente:

image-1665138945590.54.51.png

El dashboard necesita cinco widgets en total, cada uno de ellos aparece en la siguiente tabla.

Widget	Variable a la que se conecta
value	hours_alarm
value	minutes_alarm
value	seconds_alarm
switch	alarm_state
led	led

Una vez que seleccionemos cada uno de los widgets, necesitaremos **vincular cada widget a una de las variables** que ya creamos en el paso anterior. Revisa la tabla de arriba para conocer las correspondencias entre widgets y variables.

image-1665139294703.38.56.png

Una vez hayamos creado todos los widgets, podemos organizarlos como queramos pulsando el botón de las flechas de la parte superior izquierda, pero realmente, si los has creado en orden, deberían tener este aspecto, que está bastante organizado:

image-1665139716202.29.21.png

Quinto paso: creamos el código

Todas las configuraciones que hemos hecho hasta ahora, han sido básicamente compiladas en un sketch especial. Este sketch contiene información sobre el dispositivo que estamos utilizando, un ID único de nuestra Thing, las variables que hemos creado y la información de red que hemos introducido. Todo lo que tenemos que hacer ahora, es añadir algo de código al sketch para crear la alarma.

Le daremos a **Things** en el menu superior horizontal y nos aparecerá Thing llamada "Nano Alarm". Ahí, hacemos clic en la pestaña **Sketch**. Aquí, deberíamos ver un boceto generado automáticamente. Este boceto es prácticamente un "caparazón" para el código que añadiremos a continuación, lo que significa que aún no tiene ninguna funcionalidad.

Lo primero que haremos será añadir las librerías que necesitaremos y crear un objeto llamado alarma, del tipo RTCZero:

```
#include "thingProperties.h"
#include "RTCZero.h"


RTCZero alarm;
```

Después, necesitaremos iniciar la comunicación a través del puerto serial en 9600 baudios, como ya hicimos en la práctica del acelerómetro:

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

Para asegurarnos de que nuestro Arduino se conecta a IoT Cloud, le pediremos que nos imprima la información sobre esa conexión con printDebugInfo(). Seguidamente inicializaremos nuestro objeto alarm, que hemos creado un poco antes y apagaremos nuestro LED dándole el valor false:

```
ArduinoCloud.printDebugInfo();
  alarm.begin();
  led = false;
}
```

Ahora, pasaremos al la función loop(). Después de actualizar la información enviada a IoT Cloud con la función update(), imprimiremos por el puerto serie la hora, minutos y segundos reales:

```
void loop() {
  ArduinoCloud.update();
```

```
  Serial.print(alarm.getHours());
  Serial.print(":");
  Serial.print(alarm.getMinutes());
  Serial.print(":");
  Serial.println(alarm.getSeconds());
}
```

Ahora, veremos que después del loop() hay algunas funciones vacías. Estas funciones han sido creadas automáticamente desde la nube y no deben ser eliminadas o el sketch no compilará. Estas funciones se crean cuando añadimos una nueva variable con permiso de lectura y escritura, y son llamadas cada vez que cambiamos el valor de la variable desde IoT Cloud. Así, para establecer una nueva hora de alarma cada vez que cambiamos el valor desde la nube, tenemos que añadir una función .setAlarmTime() dentro de las funciones que estaban vacías:

```
void onHoursAlarmChange() {
  alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
}

void onMinutesAlarmChange() {
  alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
}

void onSecondsAlarmChange() {
  alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
}
```

La función onAlarmStateChange() es llamada cada vez que encendamos o apaguemos nuestra alarma desde el interruptor (Switch) que hemos creado antes en nuestra Dashboard. Activaremos la alarma si el interruptor está en ON (true) y lo desactivaremos si lo colocamos en OFF (false). Aquí, necesitaremos un condicional para ver si el interruptor está encendido o apagado.

Si la alarma está encendida, usaremos las funciones **.setAlarmTime()**, **.enableAlarm()** y . **attachInterrupt()**, que ya vienen programadas y vinculadas a nuestro objeto alarm, para activarla.

Si el interruptor está apagado, usaremos la función .disableAlarm() para desactivar nuestra alarma y apagaremos nuestro LED:

```
void onAlarmStateChange() {
  if (alarm_state == true) {
    alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
    alarm.enableAlarm(alarm.MATCH_HHMMSS);
    alarm.attachInterrupt(alarmMatch);
  }
  else {
    alarm.disableAlarm();
    led = false;
  }
}
```

El último paso es crear una nueva función llamada **.alarmMatch()** que controlará lo que ocurre cuando la alarma coincida con la hora real enviada por nuestro NANO 33 IoT. En este caso, usaremos esta función para encender el LED de la Nube poniendo el valor del LED a true:

```
void alarmMatch() {
  led = true;
}
```

Por si te has perdido en algún paso copiando y pegando el código, lo encuentras todo a continuación:

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Nano_Alarm"
  Arduino IoT Cloud Variables description
  The following variables are automatically generated and updated when changes are made to the
Thing
  int hours_alarm;
  int seconds_alarm;
  int minutes_alarm;
  bool alarm_state;
  bool led;
  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/
```

```
#include "thingProperties.h"
#include "RTCZero.h"

RTCZero alarm;

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

  // This delay gives the chance to wait for a Serial Monitor without blocking if none is
found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
  */

  setDebugMessageLevel(2);

  ArduinoCloud.printDebugInfo();

  alarm.begin();

  led = false;
}
```

```
void loop() {
  ArduinoCloud.update();

  Serial.print(alarm.getHours());
  Serial.print(":");
  Serial.print(alarm.getMinutes());
  Serial.print(":");
  Serial.println(alarm.getSeconds());
}

void onHoursAlarmChange() {
  alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
}

void onMinutesAlarmChange() {
  alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
}

void onSecondsAlarmChange() {
  alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
}

void onAlarmStateChange() {
  if (alarm_state == true) {
    alarm.setAlarmTime(hours_alarm, minutes_alarm, seconds_alarm);
    alarm.enableAlarm(alarm.MATCH_HHMMSS);
    alarm.attachInterrupt(alarmMatch);
  }
  else {
    alarm.disableAlarm();
    led = false;
  }
}

void alarmMatch() {
```

```
  led = true;
}
```

Una vez lo tengamos en nuestro sketch, llegará el momento de verificarlo, haciendo click sobre el icono del tick y posteriormente subirlo a nuestro Arduino haciendo click sobre la flecha hacia la derecha:

image-1665163975375.04.50 copia.png

Una vez lo hayamos subido, veremos algo parecido a esto:

image-1665164989181.gif

Si te has fijado, la hora en mi ordenador son las 14:50 y no las 12:50 cuando la alarma se enciende. Eso es porque mi Arduino daba la hora con 2h de retraso. ¿Cómo podemos solucionarlo? Muy fácil, en nuestro código he modificado una línea, la que aparece en el siguiente bloque en la línea 3, y le he restado 2h a la hora a la que he fijado mi alarma, para que así coincida con la hora de mi Arduino:

```
void onAlarmStateChange() {
  if (alarm_state == true) {
    alarm.setAlarmTime(hours_alarm-2, minutes_alarm, seconds_alarm);
    alarm.enableAlarm(alarm.MATCH_HHMMSS);
    alarm.attachInterrupt(alarmMatch);
  }
  else {
    alarm.disableAlarm();
    led = false;
  }
}
```

> Para saber la hora de nuestro Arduino, abviamente con nuestro Arduino conectado, tendremos que abrir el puerto serie. Para ello, haremos clic en la lupa que se ve en la siguiente imagen:

Captura de pantalla 2023-01-27 a las 21.53.07.png

Una vez sepamos la hora de nuestro Arduino, podremos escribirla correctamente en nuestra Alarma.

Ahora ya funciona perfectamente:

image-1665165906946.gif

> Tenemos que considerar que al ser un servicio conectado a Internet, hay cierta latencia y los valores no se actualizan inmediatamente. Si queremos comprobar que nuestra alarma funciona, es mejor que la pongas con un margen de unos 2 o 3 minutos.

Problema frecuente

Puede que nuestro Arduino nos aparezca offline aunque nos permita subir el código.
1. Elimina tu Arduino completamente de **Devices** y vuélvelo a instalar.
2. Como IoT Cloud es un servicio que vamos a utilizar en su versión gratuita, puede presentar latencias y problemas de conexión si hay mucha gente utilizándolo. En mi experiencia personal, he tenido más problemas tratando de conectarme por las mañanas que por las tardes.
3. Para realizar pruebas, te aconsejo que te crees dos cuentas en Arduino IoT Cloud, utilizando dos cuentas de email diferentes, para poder alternar tus proyectos entre ellas. Puede ocurrir que tu Arduino te aparezca offline en una de tus cuentas y online en la otra.

¿Qué tenemos que entregar?

Graba un video similar al .gif que aparece en el apartado anterior, que se vea la alarma apagada y que cuando marca la hora se encienda el LED.

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 esta tarea.

FUENTES:
Alarma con Arduino 33 IoT: https://docs.arduino.cc/tutorials/nano-33-iot/iot-cloud

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

logo.png

Proyecto Final

En esta última parte del curso, me gustaría que **idearas un proyecto** en el que combines parte de los sensores que hemos utilizado con la plataforma IoT Cloud, para poder implementar lo que hemos visto en este curso dentro de tus clases.

Para ello, lo primero que vas a tener que hacer es pensar en una aplicación de los **sensores** y **actuadores** que hemos utilizado, y con **pensar en una aplicación** me refiero a que vas a tener que pensar en una situación en la que estos sensores y actuadores podrían ayudar a resolver un problema o a llamar la atención sobre una determinada cuestión.

Una pregunta con la que comenzar podría ser, **¿podría un vestible solucionar o hacernos reflexionar sobre qué problema/situación?**

Algunos consejos

No te compliques la vida intentando integrar varios sensores y actuadores. Es mejor utilizar **un solo sensor** y **un único actuador**. Una vez funcione, podremos plantearnos añadir más elementos.

El **actuador** que escojas para tu proyecto si es físico tiene que ser el LED pero también puede ser virtual, como el LED que nos ofrece Arduino IoT Cloud y que ya utilizamos en una de las prácticas anteriores.

¿Qué tengo que presentar?

MEMORIA:
Extensión: no más de 5 páginas (sin contar la portada)
Página 0: portada con nombre completo, título del proyecto
Página 1: Motivación: por qué este proyecto; Elección de sensores y actuadores
Páginas 2-3: Funcionamiento: qué harán los sensores, adaptación del código que hemos visto a nuestro proyecto. Será necesaria una transcripción del código que utilices.
Páginas 4-5: Conclusiones, dificultades encontradas y referencias o fuentes de inspiración

DOCUMENTACIÓN: Será necesario un video o gif de menos de 1 minuto en el que se vea el funcionamiento.

Un pequeño ejemplo

En el .gif que puedes ver a continuación, he usado el código de uno de nuestros sketchs, el que utilizábamos con el **acelerómetro,** para enviar el cambio en el **eje x** a IoT Cloud y poder almacenarlo en la nube. Ha sido necesario realizar algunas adaptaciones, como crear la variable x en IoT Cloud (ya hemos visto en las prácticas anteriores cómo se hacía) y eliminarla del sketch, para que no aparezca como duplicada.

A partir de este pequeño ejemplo, si quisiera, podría encender un **LED virtual** si ese eje alcanzará una posición determinada.

image-1665161365476.gif

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

logo.png