



UNIVERSIDAD
DE LIMA

FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

ASIGNATURA:

INTERNET DE LAS COSAS IoT

GUÍA DE LABORATORIO: TEORÍA

PRÁCTICA DE LABORATORIO N.º 3 - 1

CONECTIVIDAD A LA NUBE - AWS

*Este material de apoyo académico
se hace para uso exclusivo de los alumnos
de la Universidad de Lima y en concordancia
con lo dispuesto por la legislación sobre
los derechos de autor: Decreto Legislativo 822*

PRÁCTICA DE LABORATORIO N.º 3 - 1 CONECTIVIDAD A LA NUBE - AWS

1. OBJETIVOS

- Comprensión de los principales comandos de la librería WIFI
- Enviar información al servidor AWS

2. FUNDAMENTO TEÓRICO

2.1 AWS IoT Core

Amazon Web Service es el servicio de nube de la empresa Amazon que ofrece más de 200 servicios en la nube en las que destacan análisis de datos, machine learning, almacenamiento, base de datos y servicios IoT. Entre los diversos servicios IoT vamos a destacar AWS IoT Core, este servicio nos permite conectar los dispositivos en la nube de una manera fácil y segura.

Se usará el servicio bajo el protocolo MQTT al ser uno de los protocolos más usados en el IoT en el cual vamos a publicar y suscribir al Broker.

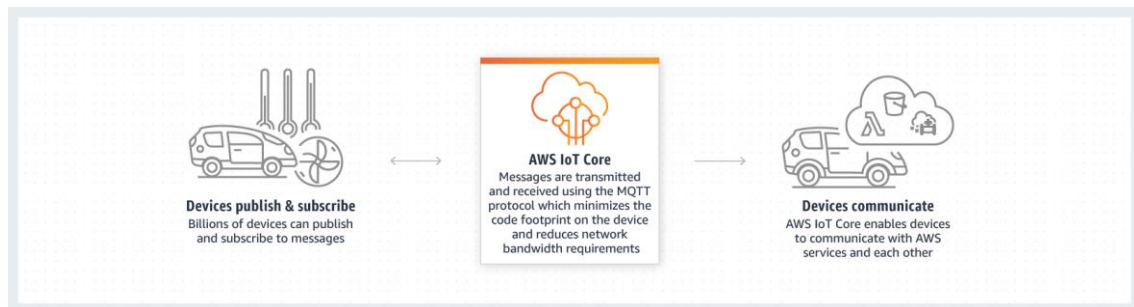


Imagen 1: Arquitectura de AWS IoT Core

Podremos crear soluciones que escalen de manera eficiente con soporte nativo para un agente MQTT administrado que admite funciones persistentes, conexiones permanentes, políticas avanzadas de retención de mensajes e identificadores. Con millones de dispositivos y temas simultáneamente, AWS IoT Core permite escalar automáticamente una solución conectada para procesar billones de mensajes y, al mismo tiempo, evitar los costos de infraestructura, las tarifas de licencia y otros gastos operativos.

AWS IoT Core incluye capacidades para varios métodos de autenticación y políticas de acceso para proteger su solución contra las vulnerabilidades. Además AWS IoT Core Device Advisor, puede acceder a conjuntos de pruebas prediseñados para validar la funcionalidad MQTT de un dispositivo durante la fase de desarrollo, incluso antes de incorporarlos a la nube.

2.2 SPIFFS

SPIFFS (SPI Flash File System) es un sistema de archivos diseñado para funcionar en memorias flash conectadas por SPI en dispositivos embebidos y con escasa cantidad de RAM, como el ESP8266 y el ESP32. Por este motivo tiene ciertas “simplificaciones” respecto a un sistema de ficheros como el que estamos habituado. Por ejemplo, SPIFFS no soporta directorios, aunque en la práctica consigue que a menudo de veces no notemos la diferencia. Así crear un fichero miDir/miFile.txt creará un fichero llamado así, en lugar de un fichero miFile.txt dentro del directorio miDir. Al listar los ficheros de un directorio, básicamente, el sistema SPIFFS “filtra” los ficheros cuya ruta empieza por el directorio deseado.

Otra limitación importante es que la longitud máxima de una ruta es de 31 caracteres, incluyendo directorio, subdirectorios, caracteres '/', nombre de archivo, y extensión. Esto es una ruta muy corta, por lo que tenedlo en cuenta a la hora de nombrar vuestros ficheros. Para nuestro caso utilizaremos el SPIFFS para poder cargar los certificados necesarios para conseguir una correcta conectividad

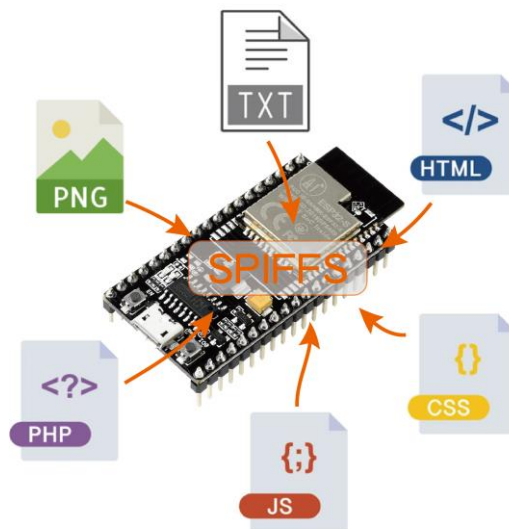


Imagen 2: Funciones SPIFFS

3. CONFIGURACIÓN DEL AWS IoT Core

Vamos a explicar todos los pasos detallados para conseguir enviar datos de forma segura al servicio IoT Core de Amazon Web Service.

3.1 Añadiendo SPIFFS al IDE de Arduino

El primer paso es instalar la herramienta SPIFFS a nuestro IDE, esto nos permitirá subir los archivos .txt con los certificados del AWS IoT Core. Necesitamos descargar el archivo comprimido desde el siguiente [link](#).

Una vez descargado el archivo buscamos en el disco duro la carpeta de Arduino, dentro de la carpeta debemos buscar otra carpeta llamada tools y descomprimir la carpeta dentro de tools, tal como se muestra en la imagen:

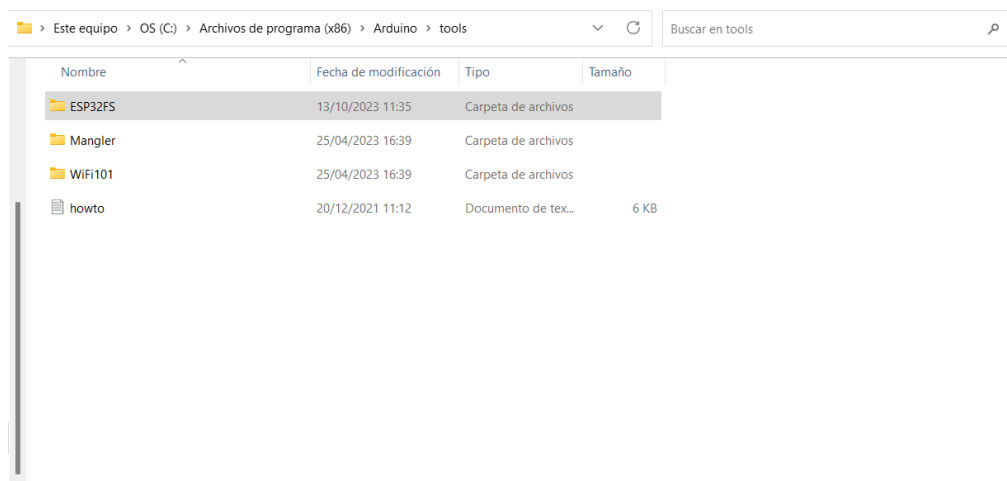


Imagen 3: Carpeta tools del IDE Arduino

El resultado final que debemos tener en el IDE Arduino es una funcionalidad nueva en el menú de herramienta del IDE que se llama ESP32 Sketch Data Upload como se muestra a continuación.

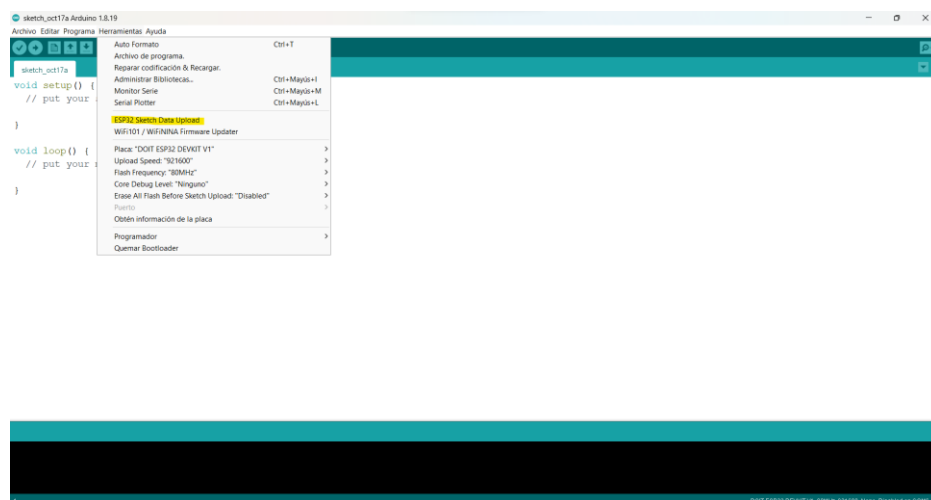


Imagen 4: IDE Arduino con SPIFFS

A partir de ahora debemos de utilizar esta forma de cargar archivos externos a nuestro IDE aprovechando la memoria del ESP32 asegurándonos el puerto COM que esta asignado el microcontrolador.

3.2 Configuración del AWS IoT Core

Antes que nada debemos de crear nuestra cuenta AWS, son varios pasos que debemos de seguir para poder habilitar el servidor de manera correcta. Una vez registrado y con la sesión iniciada, buscamos en el navegador AWS IoT Core, tal como se muestra la imagen:

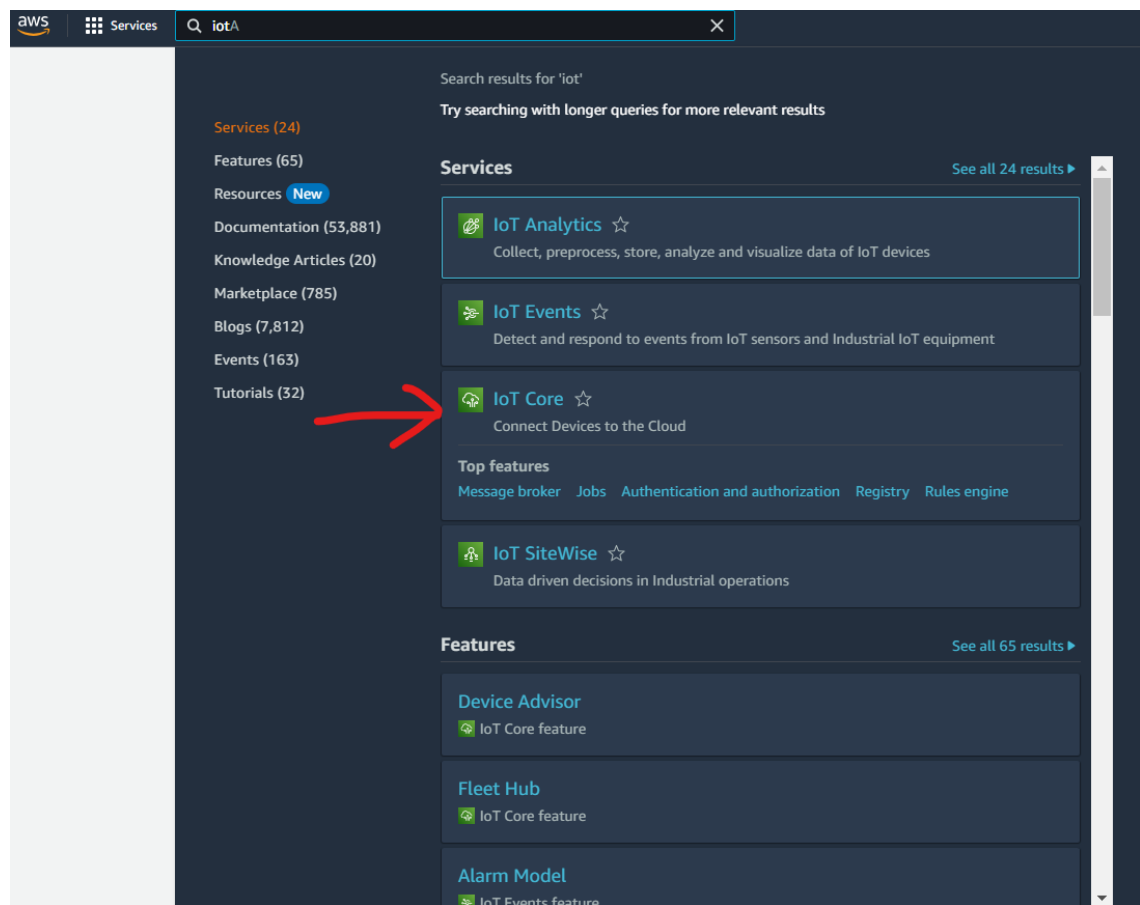


Imagen 5: AWS IoT Core

Al ingresar al AWS IoT Core debemos buscar en el menú desplegable en la izquierda, vamos a crear un *Things*, es decir un nuevo proyecto. En la imagen se verá la ruta que se debe de seguir, ya existe un proyecto llamado potenciómetro que se usó para las primeras conexiones, vamos a empezar con un nuevo proyecto dando un click en *create things*

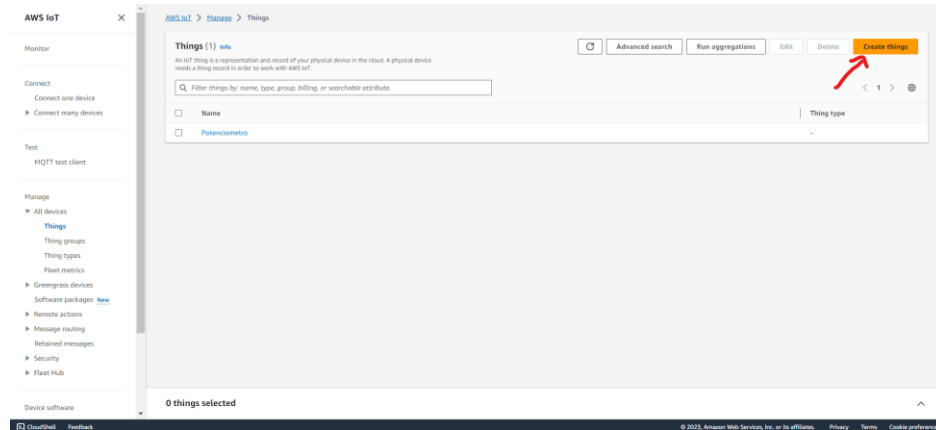


Imagen 6: Creación de un nuevo proyecto

Creamos un proyecto nuevo simple (Create simple thing) y le asignamos un nombre, en este caso lo llamaremos: Laboratorio_prueba, el resto de parámetros no se modifican.

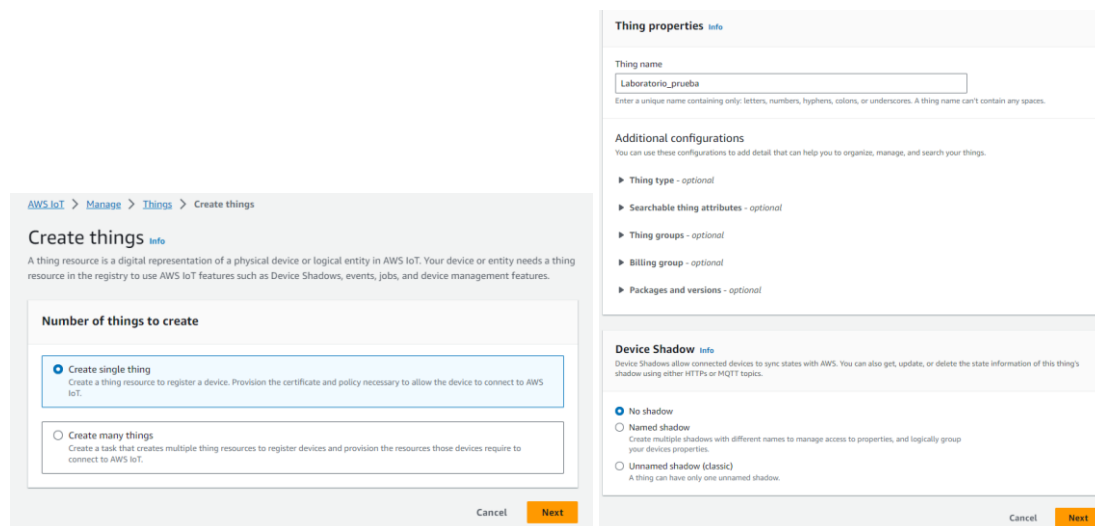


Imagen 7: Parámetros y nombre del nuevo proyecto

Por defecto el servidor te solicita la creación inmediata de los certificados para poder asignar la conexión MQTT, algo similar a la asignación de una API. En este caso no los crearemos ahora ya que solo tenemos una sola oportunidad para descargarlos, así que lo haremos en un paso posterior

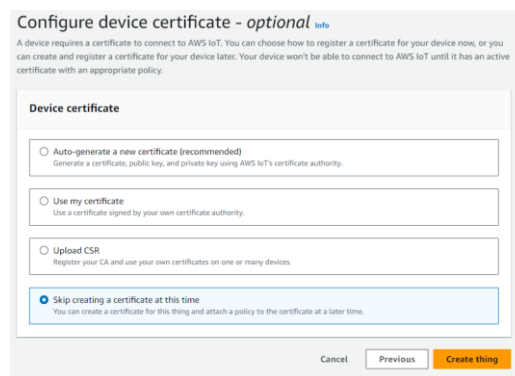


Imagen 8: Paso final para la creación del proyecto

Las políticas de AWS IoT le permiten controlar el acceso a las operaciones de nuestros datos en el AWS IoT Core. Para ello vamos a asignarle una política de seguridad, en el menú izquierda buscamos el apartado seguridad y luego políticas.

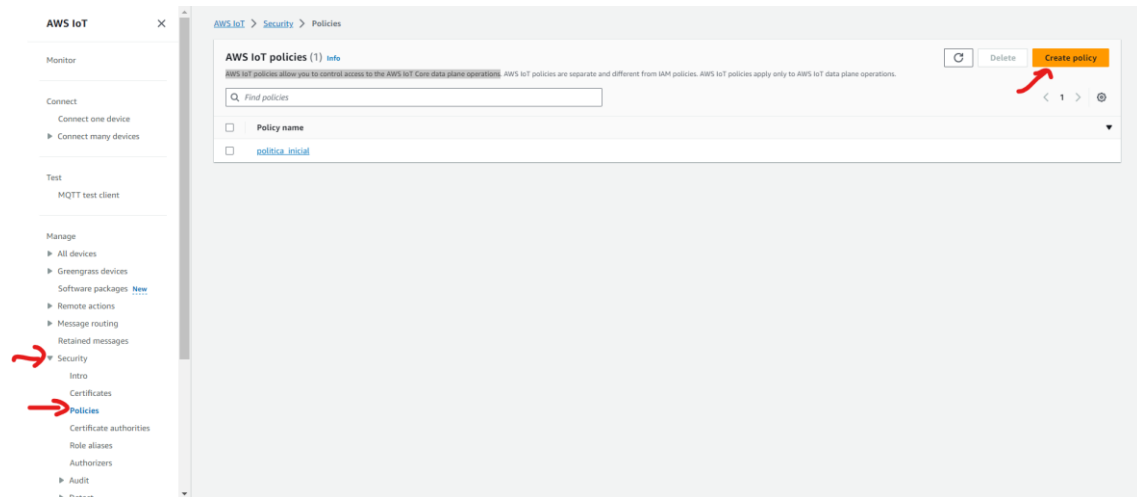


Imagen 9: Creación de una política de seguridad

Le asignamos el nombre de la política política_lab_prueba, en el apartado de builder eligo las opciones de *policy effect*, *policy action* y *policy resource* las opciones Allow y *. Este asterisco indica que eliger todas las acciones y recursos disponibles a la política creada.

El siguiente paso es crear los certificados que se subirán al ESP32 mediante el SPIFFS. En el menú de la izquierda busco en el apartado de seguridad y luego certificados. En la imagen ya existe un certificado pero debemos de crear uno nuevo, en la imagen se aprecia como crear uno en lugar de añadir un certificado:

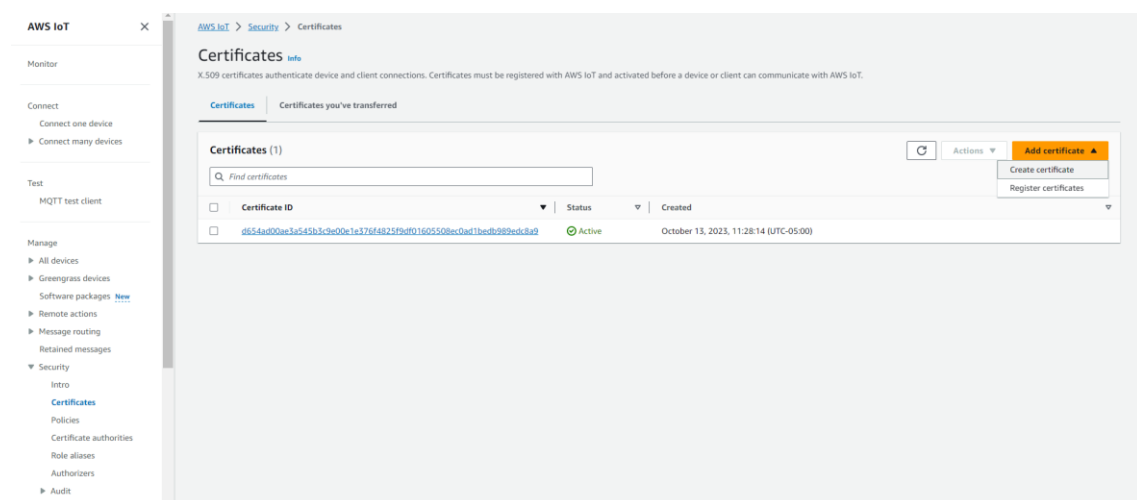
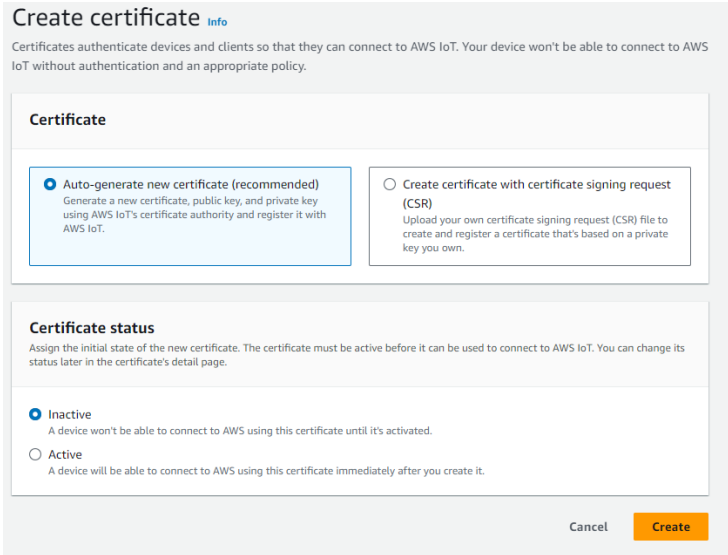


Imagen 10: Creación de un certificado

Es el momento de crear el certificado, hacemos que el servidor genere un certificado automático y lo dejamos inactivo por el momento, esto sirve para poder descargarlos sin problemas, recordar que una vez pasado este paso no podemos volver a descargarlo.



Create certificate [Info](#)

Certificates authenticate devices and clients so that they can connect to AWS IoT. Your device won't be able to connect to AWS IoT without authentication and an appropriate policy.

Certificate

☒ **Auto-generate new certificate (recommended)**
Generate a new certificate, public key, and private key using AWS IoT's certificate authority and register it with AWS IoT.

☐ **Create certificate with certificate signing request (CSR)**
Upload your own certificate signing request (CSR) file to create and register a certificate that's based on a private key you own.

Certificate status

Assign the initial state of the new certificate. The certificate must be active before it can be used to connect to AWS IoT. You can change its status later in the certificate's detail page.

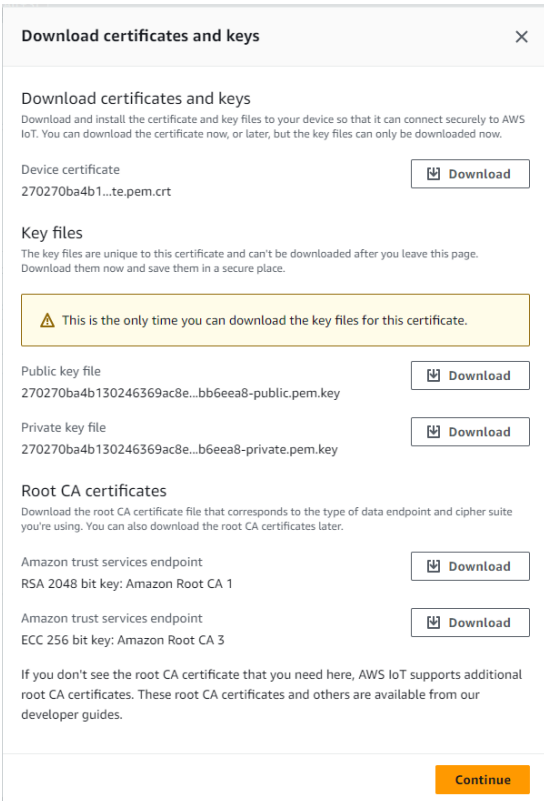
☒ **Inactive**
A device won't be able to connect to AWS using this certificate until it's activated.

☐ **Active**
A device will be able to connect to AWS using this certificate immediately after you create it.

Cancel **Create**

Imagen 11: Opciones del certificado

Una vez creados los certificados se verá la siguiente imagen con todos los archivos que debemos de descargar en una carpeta definida:



Download certificates and keys ✕

Download certificates and keys

Download and install the certificate and key files to your device so that it can connect securely to AWS IoT. You can download the certificate now, or later, but the key files can only be downloaded now.

Device certificate
270270ba4b1...te.pem.crt **Download**

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

Public key file
270270ba4b130246369ac8e...bb6eea8-public.pem.key **Download**

Private key file
270270ba4b130246369ac8e...b6eea8-private.pem.key **Download**

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1 **Download**

Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3 **Download**

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available from our developer guides.

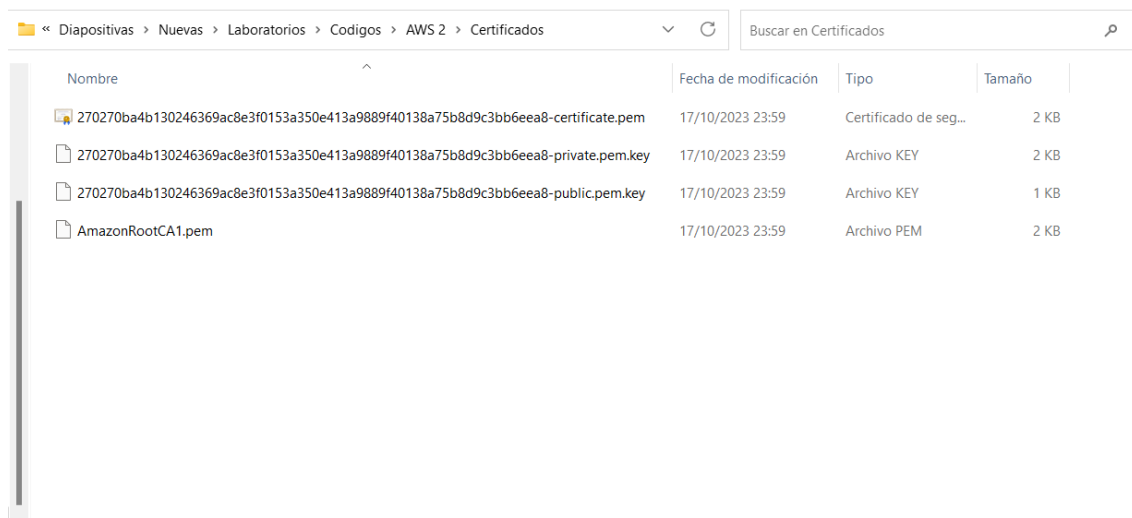
Continue

Imagen 12: Archivos para decargar

Los archivos que vamos a descargar son los siguientes:

- Device certificate
- Public Key file
- Private key file
- Amazon Root CA1

Estos archivos los vamos a utilizar más adelante, por el momento lo dejamos en una carpeta todos los archivos.







« Diapositivas > Nuevas > Laboratorios > Codigos > AWS 2 > Certificados					Buscar en Certificados	
Nombre	Fecha de modificación	Tipo	Tamaño			
 270270ba4b130246369ac8e3f0153a350e413a9889f40138a75b8d9c3bb6eea8-certificate.pem	17/10/2023 23:59	Certificado de seg...	2 KB			
 270270ba4b130246369ac8e3f0153a350e413a9889f40138a75b8d9c3bb6eea8-private.pem.key	17/10/2023 23:59	Archivo KEY	2 KB			
 270270ba4b130246369ac8e3f0153a350e413a9889f40138a75b8d9c3bb6eea8-public.pem.key	17/10/2023 23:59	Archivo KEY	1 KB			
 AmazonRootCA1.pem	17/10/2023 23:59	Archivo PEM	2 KB			

Imagen 13: Certificados descargados

Todos los certificados en estos momentos están inactivos y no se asignaron a una política de seguridad, para ello vamos a activar el certificado y luego asignarle una política, tal como se muestra en la imagen

The screenshot shows the AWS IoT console 'Certificates' page. It displays a table of certificates with columns for Certificate ID, Status, and Created. Two certificates are listed: one active and one inactive. The 'Actions' menu is open for the active certificate, showing options like Activate, Deactivate, Revoke, etc. Below the table, a modal window titled 'Attach policies to the certificate' is shown, displaying the certificate ID and a list of policies to attach. The policy 'politica_lab_prueba' is selected. At the bottom, there are 'Cancel' and 'Attach policies' buttons.

Certificates info
X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with AWS IoT.

Certificates Certificates you've transferred

Certificates (1/2)

	Certificate ID	Status	Created
<input type="checkbox"/>	d654ad00ae3a545b3c9e0e1e376f4825f9d01605508ec0ad1bedb98bedc8a9	Active	October 13, 2023, 11:28:14 (UTC-05:00)
<input checked="" type="checkbox"/>	270270ba4b130246369ac8e3f0153a350e413a9889f40138a75b8d9c3bb6eea8	Inactive	October 17, 2023, 23:55:52 (UTC-05:00)

Attach policies to the certificate
270270ba4b130246369ac8e3f0153a350e413a9889f40138a75b8d9c3bb6eea8.

Policies
Choose policies to attach to this certificate. The certificate can have up to 10 policies attached to it.

Choose AWS IoT policy

politica_lab_prueba

Cancel Attach policies

Imagen 14: Asignación de una política al certificado

Finalizado toda la configuración del servidor, es momento de analizar el código de Arduino para enlazarlo con nuestra cuenta de AWS IoT Core

3.3 Código de arduino

El primer paso que debemos de asegurarnos es tener el SPIIFS instalado y que funcione la librería, además de instalar la librería PubSubClient que realizará la conexión MQTT con el servidor, lo podremos descargar del siguiente [enlace](#).

Debemos de enlazar nuestro ESP32 a nuestro servidor, para ello debemos de copiarlo desde nuestra cuenta, para ello buscamos en el menú izquierdo settings y procedemos a copiar el Endpoint, tal como se muestra en la imagen:

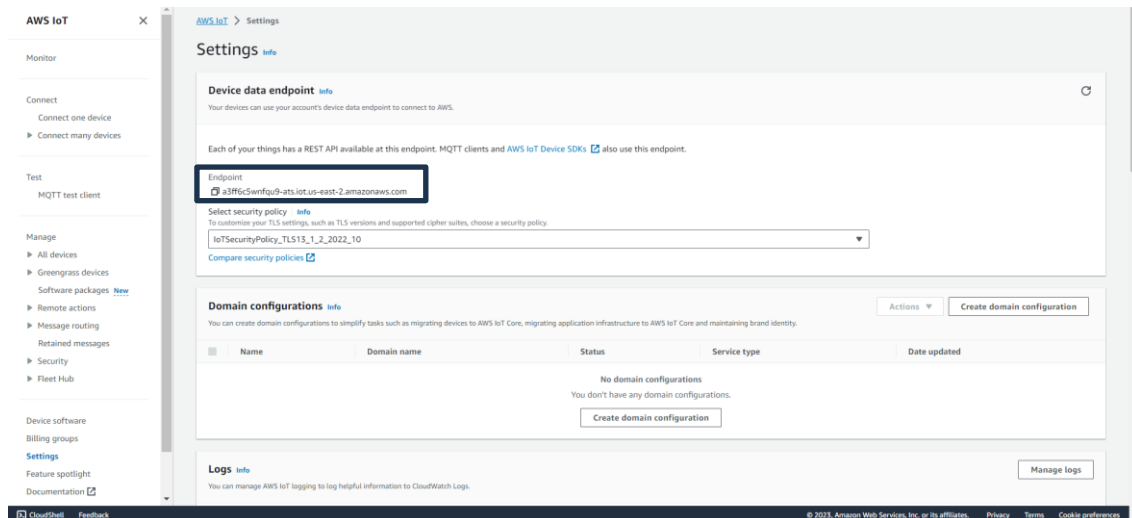


Imagen 15: API key de nuestro servidor

Procedemos a crear un cliente por medio de la librería Wifi y a su vez lo asigno a un cliente de la librería de suscripción, todo esto para poder crear mi cliente MQTT. Los códigos son los siguiente:

```
WiFiClientSecure espClient;
PubSubClient client(espClient);
```

Las siguientes líneas corresponden a la conexión de la red WiFi y la subrutina callback, encargada de enviar la información al servidor usando las propiedades de comunicación del servidor MQTT. La subrutina *reconnect()* es la encargada de conectarse al servidor MQTT y mantenerse conectada.

Dentro del setup procedemos a inicializar el SPIFFS para subir los certificados. Es muy importante crear una carpeta llamada data dentro de la carpeta del código, tal como se muestra en la imagen:

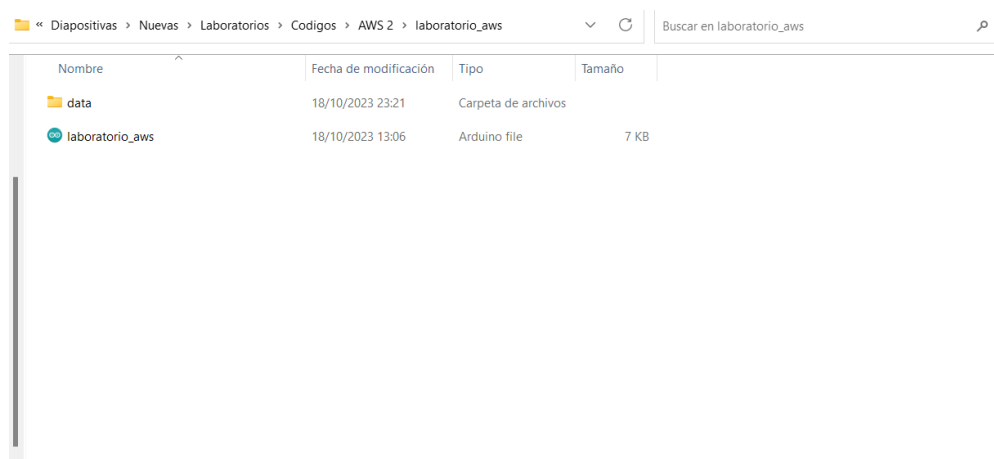
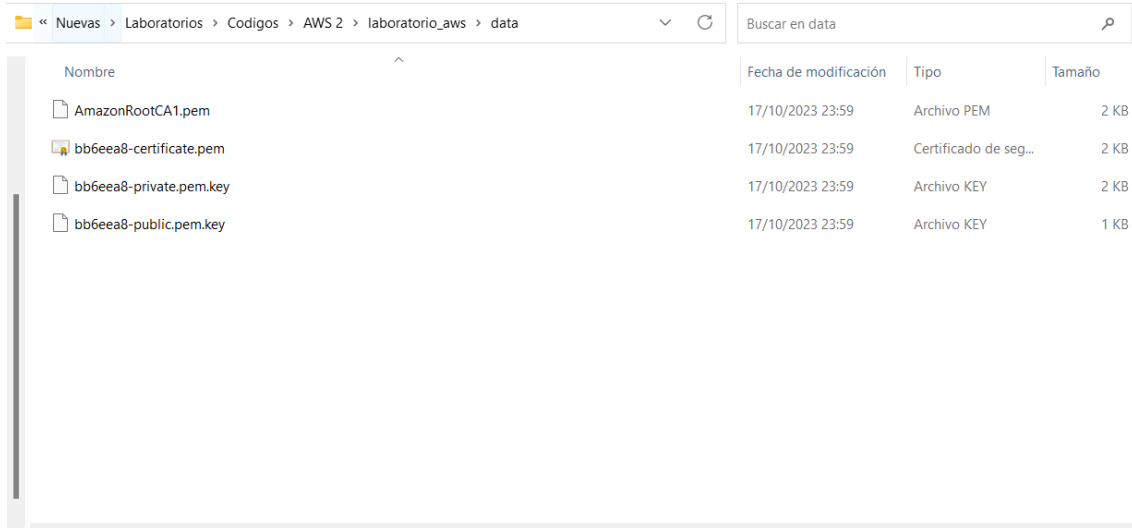


Imagen 16: Carpeta del código a usar

Dentro de esta carpeta vamos a guardar los certificados descargados con anterioridad, una vez pegados los archivos en este punto vamos a modificar sus nombres, debemos recordar que el SPIFFS tiene un máximo de 31 caracteres, por lo que dejamos los archivos similares a la imagen. Solo el archivo raíz se mantiene.



Nombre	Fecha de modificación	Tipo	Tamaño
AmazonRootCA1.pem	17/10/2023 23:59	Archivo PEM	2 KB
bb6eea8-certificate.pem	17/10/2023 23:59	Certificado de seg...	2 KB
bb6eea8-private.pem.key	17/10/2023 23:59	Archivo KEY	2 KB
bb6eea8-public.pem.key	17/10/2023 23:59	Archivo KEY	1 KB

Imagen 17: Archivos reducidos

Dentro del código los pegamos en sus respectivas secciones, tal como se muestra en el siguiente código

```
//*****
// Cert leer archivo
File file4 = SPIFFS.open("/bb6eea8-certificate.pem.crt", "r");
if (!file4) {
    Serial.println("No se pudo abrir el archivo para leerlo");
    return;
}
Serial.println("Cert File Content:");
while (file4.available()) {
    Read_cert = file4.readString();
    Serial.println(Read_cert);
}
//*****
//Privatekey leer archivo
File file6 = SPIFFS.open("/bb6eea8-private.pem.key", "r");
if (!file6) {
    Serial.println("No se pudo abrir el archivo para leerlo");
    return;
}
Serial.println("privateKey contenido:");
while (file6.available()) {
    Read_privatekey = file6.readString();
    Serial.println(Read_privatekey);
}
//=====
```

Imagen 18: Modificación de los archivos

El resto de setup se encarga de subir los certificados, imprimirlos en el monitor serial y conectarnos al servidor MQTT del AWS.

En el loop creamos 2 variables aleatorias, estos valores pueden ser tranquilamente lecturas de sensores que deseamos enviar a nuestro servidor. Estos valores tienen que ser valores tipo string para poder crear una cadena con la información necesaria para realizar la carga de datos en el servidor. Una vez concatenado toda la información, la enviamos a un topic que lo llamaremos *laboratorio*.

Las últimas líneas nos darán como referencia al momento que enviamos un dato, se encenderá y apagará un led.

El siguiente paso es importante, no debemos de cargar el código como solemos cargar un código normal, debemos de realizarlo a través del SPIFFS, tal como se muestra en la imagen:

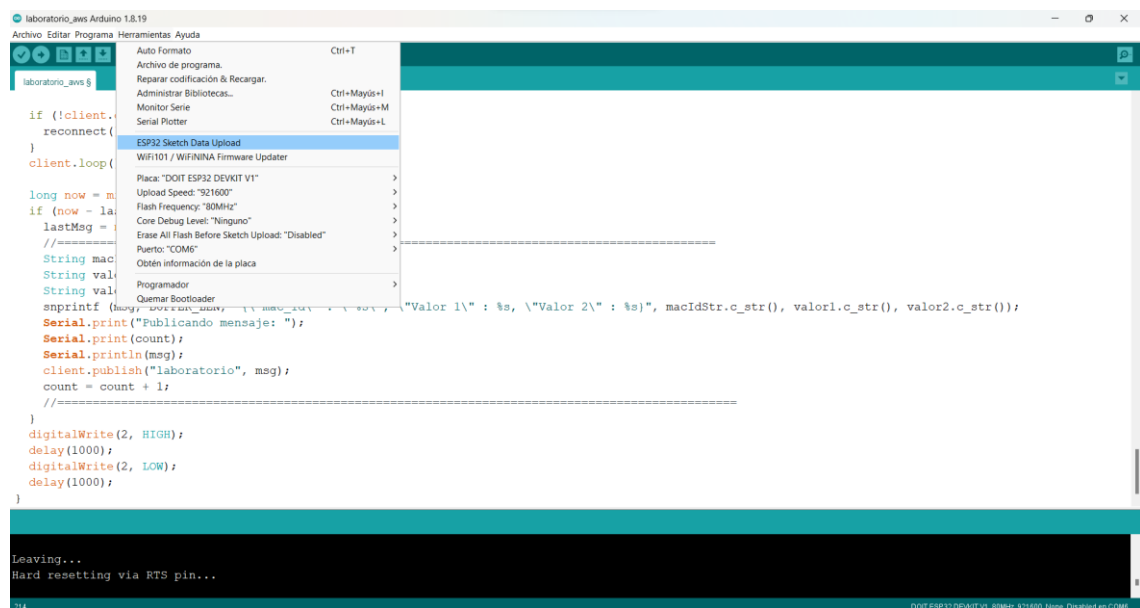
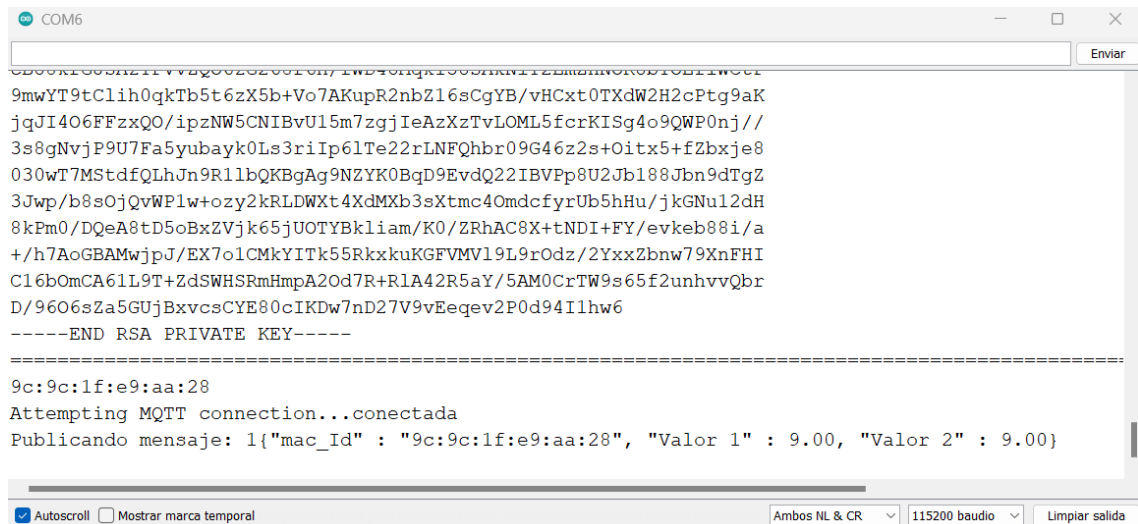


Imagen 19: Cargando el código

Una vez cargado el código y si se siguió la guía, nos conectaremos al servidor y se mostrará la cadena concatenada con la información que se envió, algo muy similar a la imagen:



```

COM6
-----END RSA PRIVATE KEY-----
9c:9c:1f:e9:aa:28
Attempting MQTT connection...conectada
Publicando mensaje: 1{"mac_Id" : "9c:9c:1f:e9:aa:28", "Valor 1" : 9.00, "Valor 2" : 9.00}
Autoscroll [x] Mostrar marca temporal [x] Ambos NL & CR [v] 115200 baudio [v] Limpiar salida [v]

```

Imagen 20: Mensaje enviado

Para poder visualizar los datos enviados tenemos que buscar en el menú de la izquierda la opción MQTT test client y escribir el topic que decidimos llamarlo laboratorio. Una vez suscritos al topic empezarán a aparecer los datos:

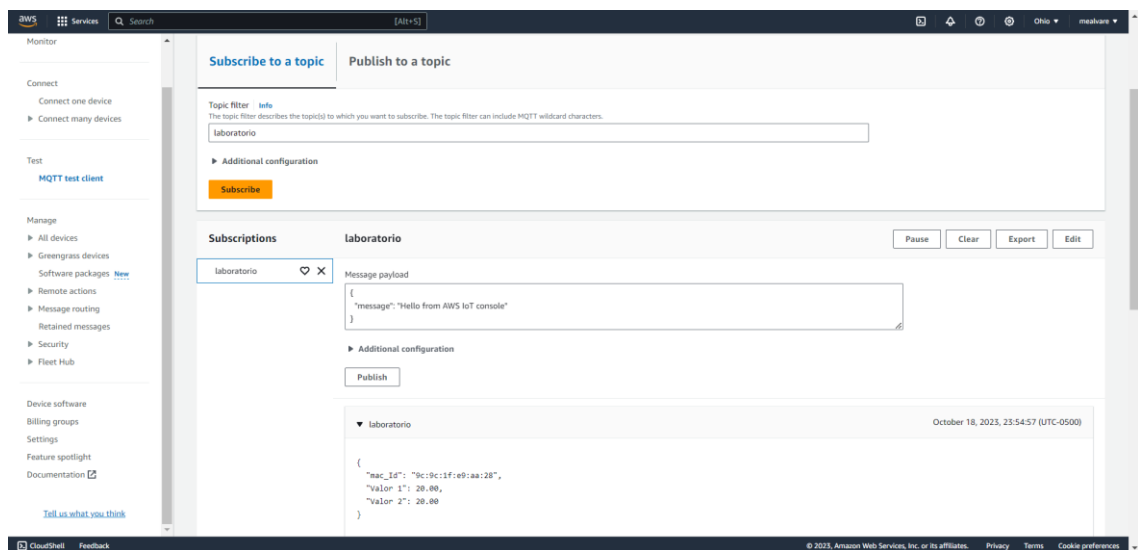


Imagen 21: Mensaje recibido en el servidor

Si comparamos los datos enviados con los que se muestra en el monitor serial se aprecia que se pueden perder algunos datos o existe un delay al subir la información con respecto a la mostrada en el monitor serial, ya queda en cada uno saber cuando y que datos enviar a la nube.