République de Côte d'Ivoire

Union – discipline – Travail

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Institut National Polytechnique

Felix Houphouët Boigny

GENIE ELECTRIQUE ET ELECTRONIQUE

Cloud Computing

Détecteur de présence à partir du cloud AWS ou AZURE

ENCADREUR PEDAGOGIQUE

M. BOBET GOUALO

CLASSE

Ing. STIC 2 INFO

EQUIPE PROJET (thème 7)

- TONDE NDEBIA Junior Gael
- COULIBALY GNINLIBAKO
- TIEKOURA
- KOUASSI Moayé Grace

ANNEE ACADEMIQUE 2023-2024

Table des matières

INTRODUCTION

Avec l'avènement de l'Internet des Objets (IoT), les solutions pour monitorer et sécuriser les espaces sont devenues plus intelligentes et plus connectées. Les plateformes de cloud comme AWS et Azure offrent des outils puissants pour créer des systèmes de détection de présence qui non seulement améliorent la sécurité mais optimisent aussi la gestion des ressources. Ce projet vise à développer un détecteur de présence sophistiqué utilisant les technologies IoT sur l'une de ces plateformes cloud. Par suite, nous allons vous montrer les différentes étapes nécessaires à la réalisation de ce projet de la conception à la réalisation.

PARTIE 1 : CADRE ET CONTEXTE DU PROJET

I. Contexte du Projet

Dans un monde où la sécurité et l'efficacité opérationnelle sont primordiales, les systèmes de détection de présence jouent un rôle crucial. Ces systèmes sont essentiels dans divers domaines tels que la sécurité domestique, la gestion des bâtiments et les systèmes de contrôle industriel. Le choix de la plateforme cloud (AWS ou Azure) pour déployer ce système est stratégique et doit être basé sur des critères de performance, de sécurité, de coût, et de facilité d'intégration.

II. Objectifs du projet

- **Développer un système fiable de détection de présence** qui utilise des capteurs IoT pour surveiller divers environnements.
- **Assurer une analyse en temps réel** des données pour une réponse rapide en cas de détection.
- **Offrir une interface utilisateur intuitive** pour le monitoring et la gestion des alertes.
- **Garantir la sécurité et la confidentialité** des données collectées.
- **Créer un système évolutif** qui peut s'adapter aux changements de taille et de configuration de l'espace surveillé.

III. Cahier de charges

- Fonctionnalités

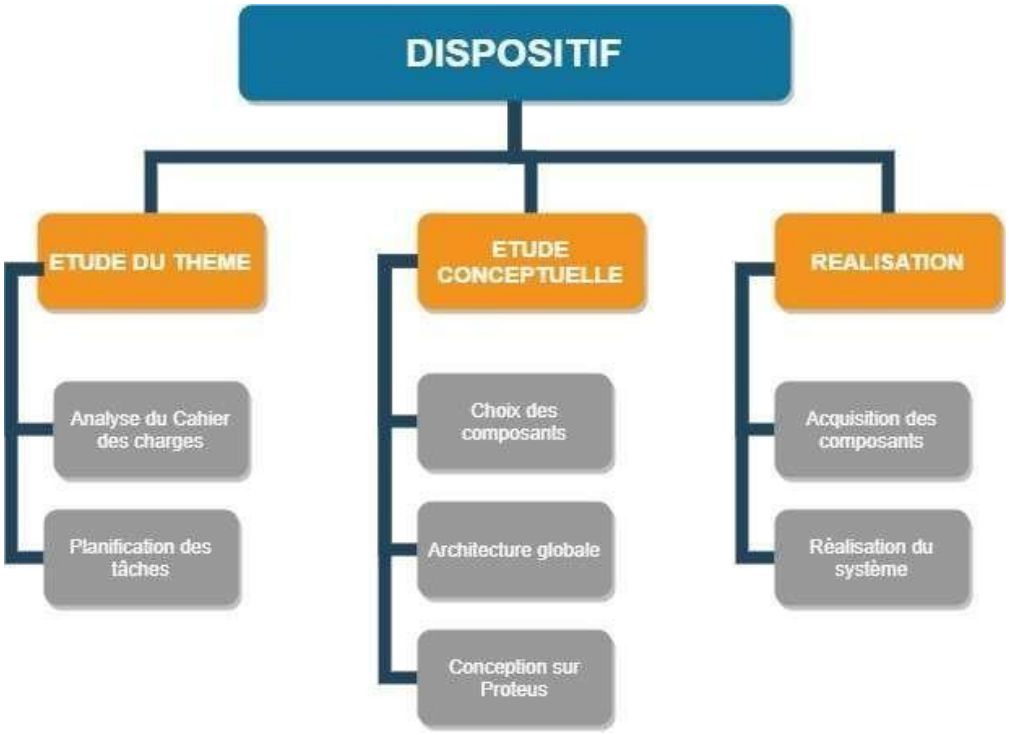
    - Surveillance en temps réel : Transmission continue des données des capteurs vers le cloud pour analyse.
    - Alertes automatiques : Envoi d'alertes en cas de détection de mouvement non autorisé.
    - Tableau de bord de gestion : Interface web et mobile pour visualiser en temps réel les activités et gérer les paramètres.
    - Intégration avec d'autres systèmes : Possibilité de connecter le système à d'autres applications de sécurité ou de gestion de bâtiment.

- Contraintes

- Fiabilité : Le système doit fonctionner de manière continue sans interruption.
- Sécurité des données : Utilisation de protocoles de sécurité avancés pour protéger les informations.
- Scalabilité : Capacité à augmenter le nombre de capteurs et à étendre la couverture sans dégradation de performance.
- Coût : Optimisation des coûts d'opération et de maintenance.

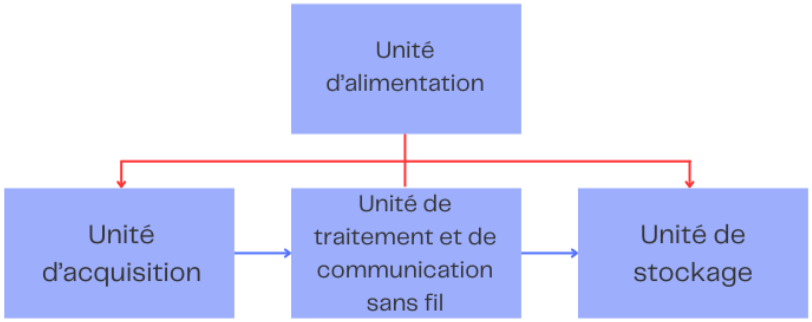
IV. Organigramme des taches

Pour mieux appréhender le travail à effectuer, nous structurons notre projet en construisant un organigramme de tâches permettant de décomposer de manière arborescente l'ensemble du projet, sans chercher à prendre en compte prématurément les notions de planning et de ressources :
La figure ci-dessous présente notre organigramme des tâches :



PARTIE 2 : CHAPITRE 2 : ETUDE CONCEPTUELLEET TECHNIQUE

I. Schéma synoptique



II. Etude détaillée du système

Notre système est composé de 4 unités à savoir :

- **L'unité d'alimentation :** l'alimentation a pour rôle de fournir la tension et le courant nécessaire au bon fonctionnement de toutes les autres unités fonctionnelles de notre système. Elle sera reliée à toutes ses différentes unités afin d'assurer leur fonctionnement. Pour alimenter notre dispositif, nous pouvons utiliser le courant fourni par un power Bank sachant que notre système sera immobile et d'autres éléments pour adapter ce courant à nos composants.
- **L'unité de traitement et de communication sans fil** : cette partie sera consacrée au traitement, c'est-à-dire traiter toutes les informations qui arriveront à son niveau. Elle sera assurée par un microcontrôleur qui traite les informations relatives aux places dans le parking. Elle envoie aussi ces informations sur un serveur en ligne, ainsi ces informations seront disponibles sur le cloud.
- **L'unité d'acquisition** : il s'agira de détecter la présence d'individu grâces aux capteurs de présence.
- **Unité de stockage :** il s'agira de stocker les informations sur le cloud grâce à AWS IoT core

III. Choix des composants électroniques et des outils logiciels

1. Outils logiciels

- **IDE ARDUINO**

L'IDE Arduino est un environnement de développement intégré (IDE) open-source conçu pour faciliter la programmation des microcontrôleurs Arduino. Il offre une interface utilisateur simple, avec un éditeur de code, des fonctions de compilation, et un programmeur pour transférer le code sur le microcontrôleur. L'IDE supporte les langages C et C++ à travers des structures simplifiées et des bibliothèques spécifiques qui rendent le codage plus accessible, notamment pour les débutants en programmation. Cet IDE est disponible pour Windows, macOS, et Linux, et il est largement utilisé pour des projets de robotique, de domotique et d'électronique DIY (Do It Yourself).

- **Choix de la technologie cloud : AWS IOT Core**

AWS IoT Core est un service cloud géré par Amazon Web Services qui permet de connecter facilement des dispositifs IoT (Internet of Things) au cloud. Ce service supporte des milliards de dispositifs et peut traiter et acheminer des trillions de messages vers et depuis ces dispositifs. AWS IoT Core permet aux applications connectées de garder une interaction sécurisée avec tous leurs dispositifs, même lorsque ceux-ci ne sont pas connectés à Internet. Ce service intègre également des fonctionnalités de sécurité robustes pour assurer l'authentification et le cryptage des communications entre les dispositifs et le cloud.

Pour ce projet, nous avions le choix entre AWS et Azure pour la partie Cloud Computing. Choisir la technologie cloud Amazon AWS IoT Core par rapport à la technologie Microsoft

qu’est Azure IoT Edge peut être justifié par plusieurs raisons :

- Intégration avec AWS : AWS IoT Core s'intègre naturellement avec une large gamme de services AWS, facilitant ainsi le développement et le déploiement de solutions IoT complètes.
- Gestion centralisée : AWS IoT Core permet une gestion centralisée des dispositifs IoT qui peut être cruciale pour la surveillance et la maintenance à grande échelle.
- Sécurité : AWS offre des solutions robustes de sécurité qui sont essentielles pour protéger les données sensibles traitées par les systèmes de détection de présence.

2. Composants électroniques

❖ **La carte ESP :**

La carte ESP est un circuit intégré à microcontrôleur avec connexion Wi-Fi. Elle analyse des signaux électriques de manière à effectuer de taches très diverses. Pour programmer cette carte, on peut utiliser l’IDE Arduino. Il y a de nombreuse cartes ESP possédant des plateformes basées sur des microcontrôleurs disponibles pour l’électronique programmée. Voir la figure suivante :

❖ **Capteur de présence :**

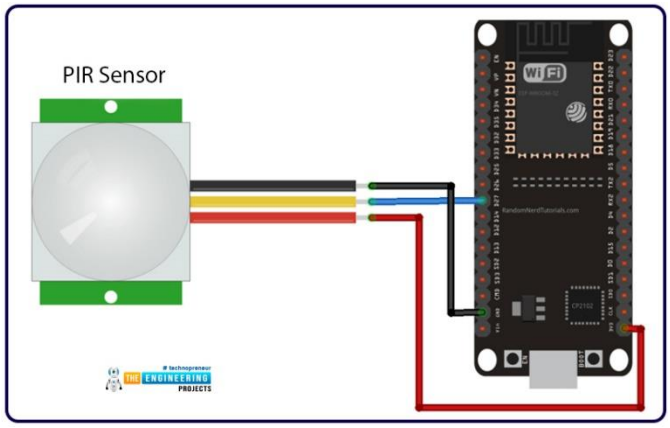
Le capteur PIR (Passive Infrared Sensor) est un dispositif électronique qui détecte les variations de chaleur émises par les objets en mouvement dans son champ de vision. Contrairement à d'autres capteurs de mouvement, le capteur PIR ne produit pas de signal actif pour mesurer le mouvement, mais plutôt il réagit aux changements de température passifs. Il est couramment utilisé dans les systèmes de sécurité domestique, les éclairages automatiques et d'autres applications où la détection de mouvement est nécessaire.

Composants
ESP 32
Capteur PIR

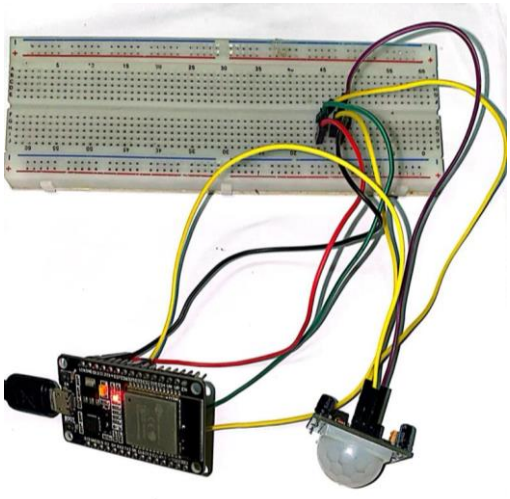
PARTIE 3 : CHAPITRE 3 : REALISATION

I. Configuration matérielle

Pour ce projet nous avons choisit d'utiliser un capteur PIR et une carte ESP 32 pour les raisons su-citées.



- Connectez le capteur PIR à la carte ESP32 comme indiqué dans le schéma de circuit ici.



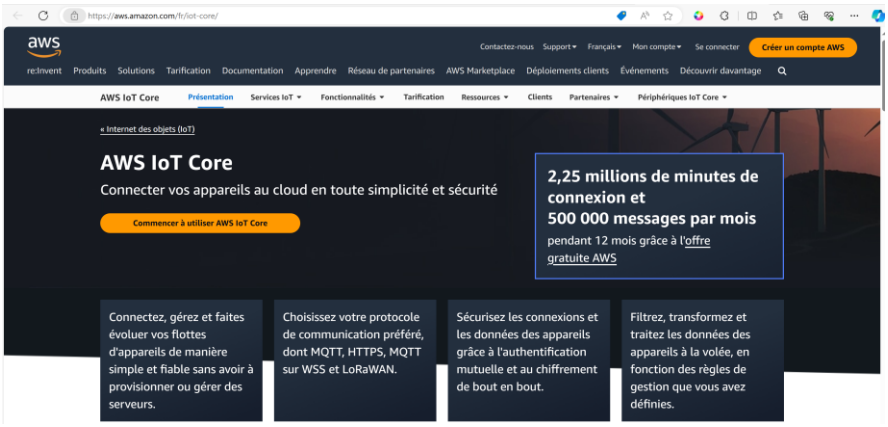
- Nous avons choisi d'utiliser une planche à pain pour la connexion nous aurions aussi simplement pu utiliser un fil de connexion mâle-femelle

II. Mise en route d'Amazon AWS IoT Core avec ESP32

La mise en route d'AWS IoT Core se fait en quelques étapes. Par suite nous allons configurer avec la carte ESP32 et lancer notre projet de détecteur de présence.

1. Connexion

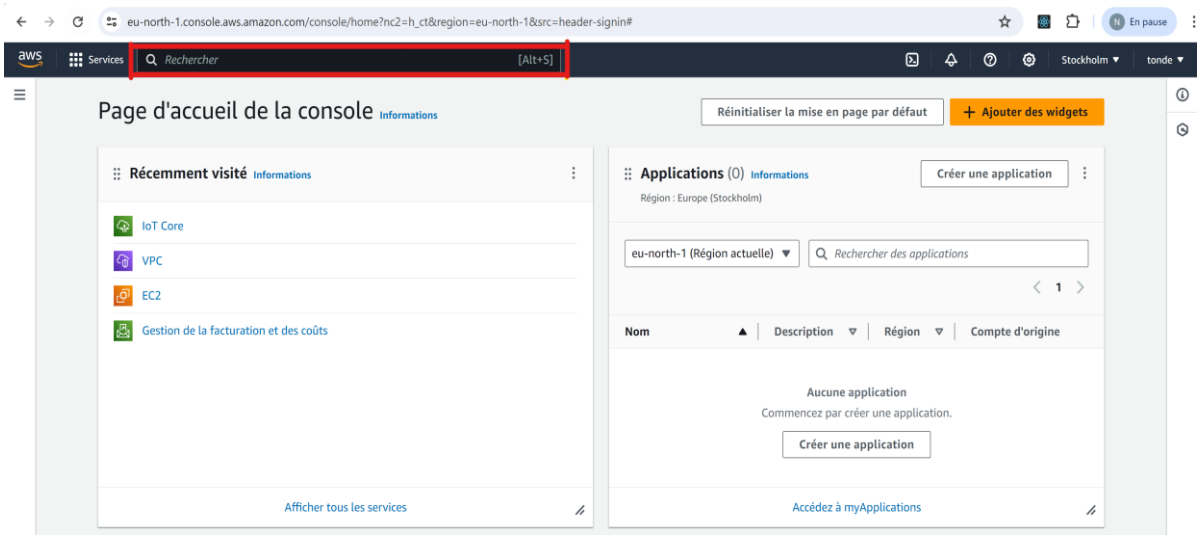
- On Accède à notre navigateur Web et recherchez le lien suivant : **aws.amazon.com/iot-core/**.



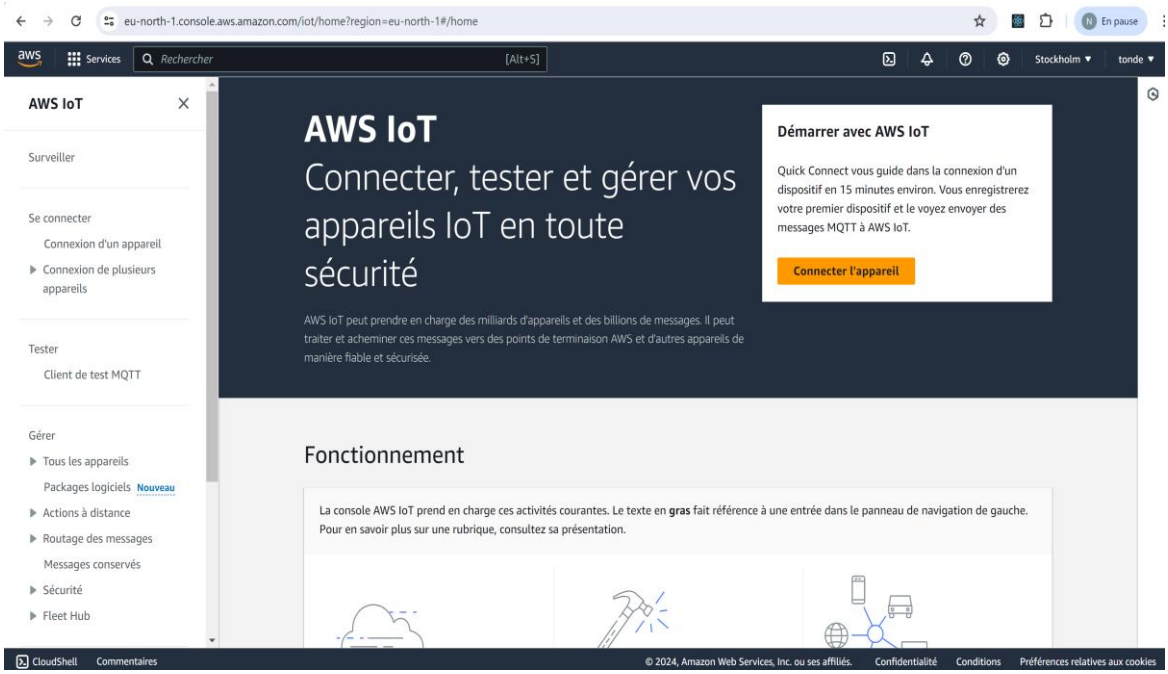
- Maintenant, nous devons configurer le compte AWS. Par conséquent, créez un compte à l'aide de l'identifiant de messagerie et du mot de passe. Le compte nécessite également des informations de carte de crédit bancaire. Il n'y aura pas de frais, mais AWS a juste besoin d'une vérification à l'aide d'un compte bancaire. Il demandera également une vérification du numéro de téléphone. Par conséquent, le compte sera créé avec succès.

2. Tableau de bord AWS IoT Core

Une fois la connexion établie, la fenêtre AWS Management Console s'ouvre. Dans l'onglet de recherche de services en haut, nous avons écrit « IoT core » et nous avons appuyé sur Entrée.



- Puis nous cliquons sur IoT Core, de sorte qu'un tableau de bord AWS IoT s'affiche.

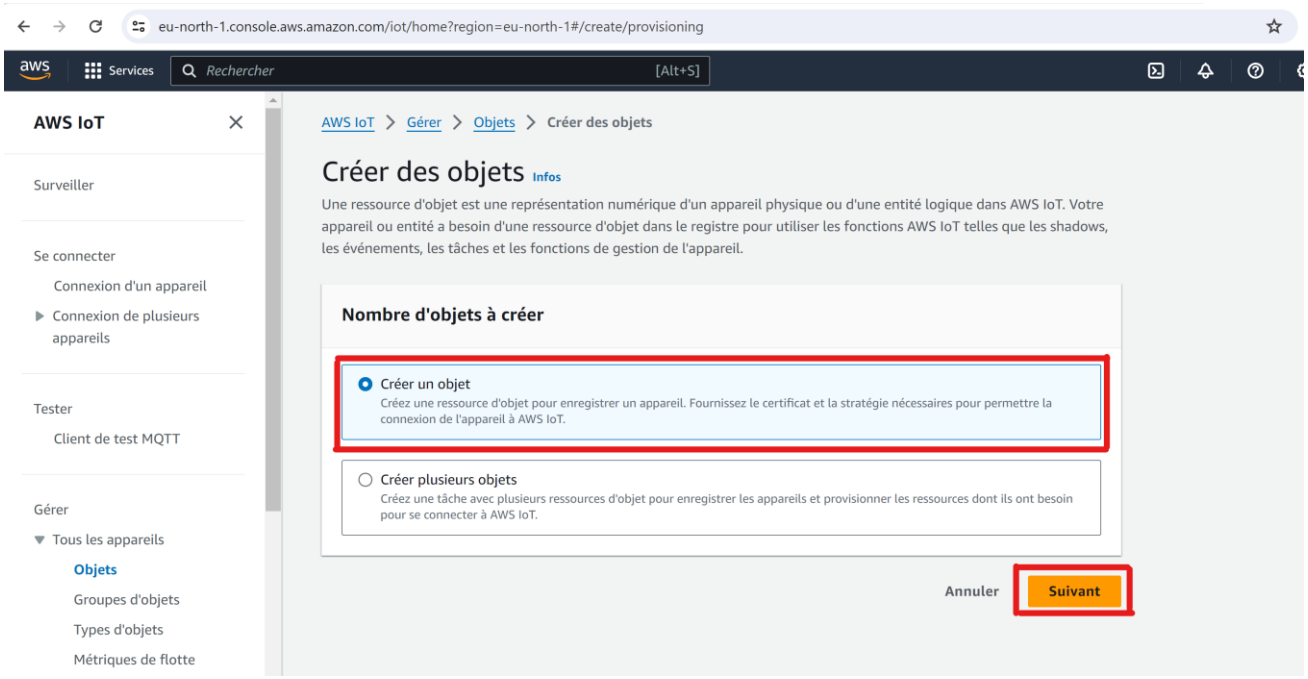


3. Créer une chose

❖ Ici, nous devons créer un « objet » associée à notre projet. Pour cela, nous avons suivis les étapes suivantes :

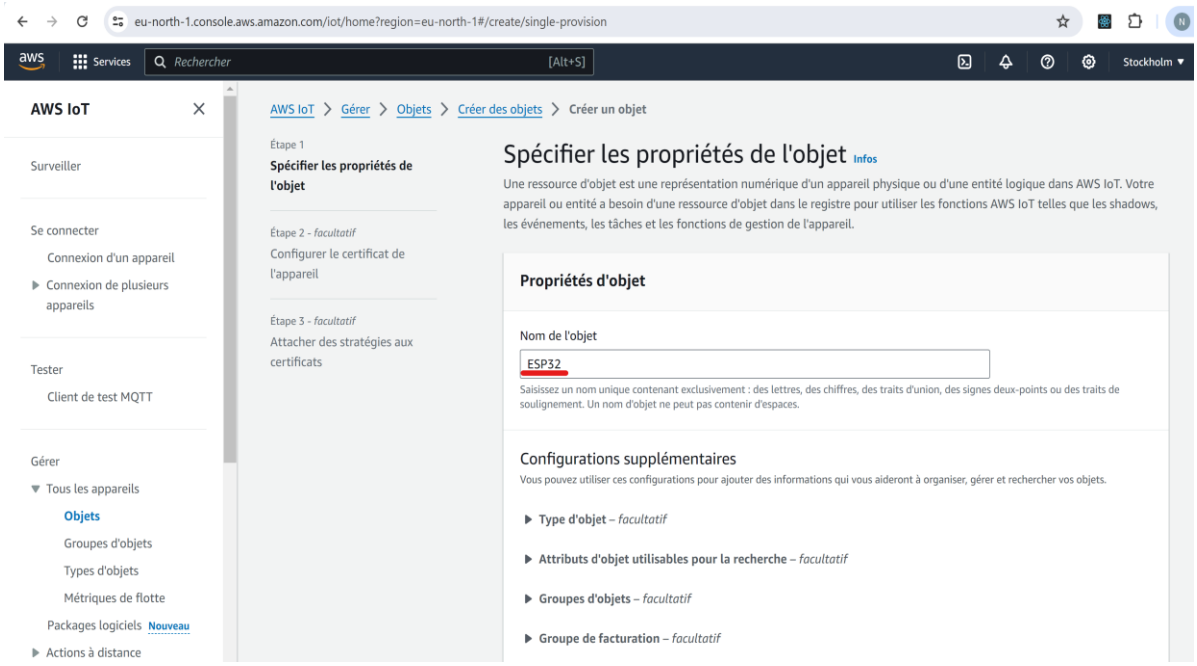
- Spécification des propriétés d'objet
- Configuration du certificat de périphérique
- Associer des stratégies à un certificat

❖ Sous l'option de « Gérer », l'on clique sur « Tous les appareils » et puis sur « Objet ». Ensuite, nous devons créer un objet. Alors, on clique sur « Créer un objet ».



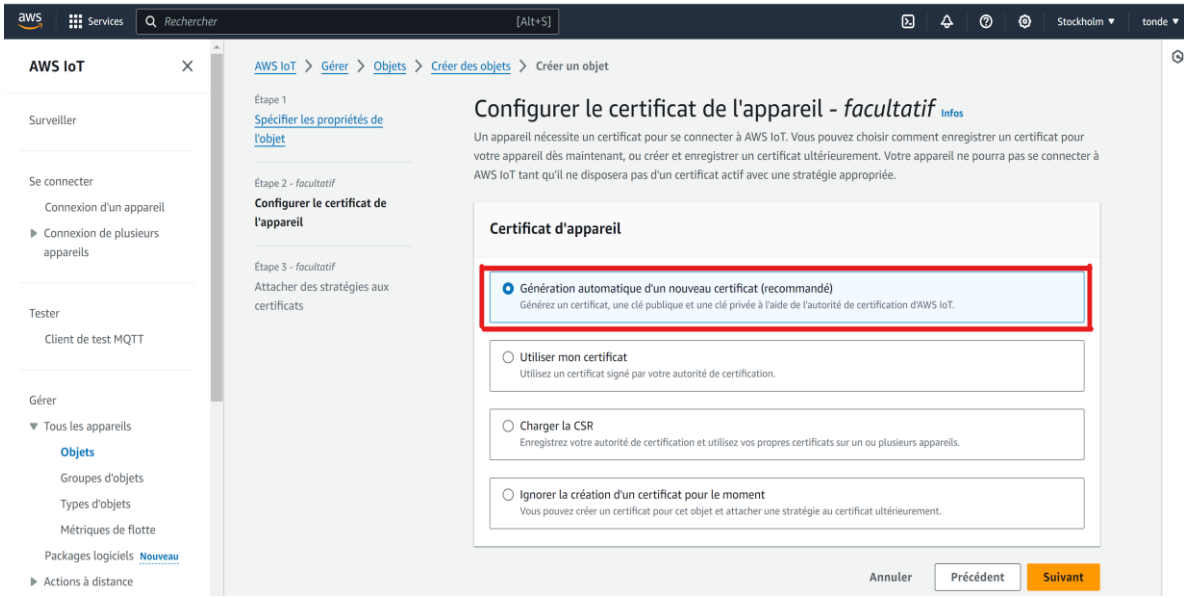
4. Spécifier les propriétés de l'objet

❖ Dans cette partie, nous devons spécifier les propriétés de l'objet. Tout d'abord, donnez un nom à l'objet. Nous lui avons donné le nom ESP32 et nous avons laissé les paramètres par défaut.



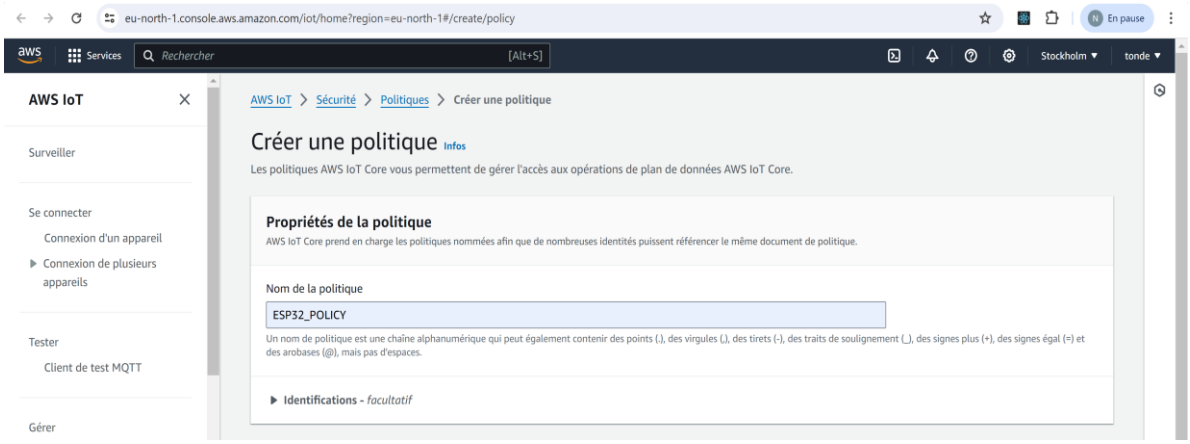
5. Générer un certificat d'appareil

- Ensuite, l'on a configuré le certificat de l'appareil : nous avons simplement généré automatiquement un nouveau certificat.

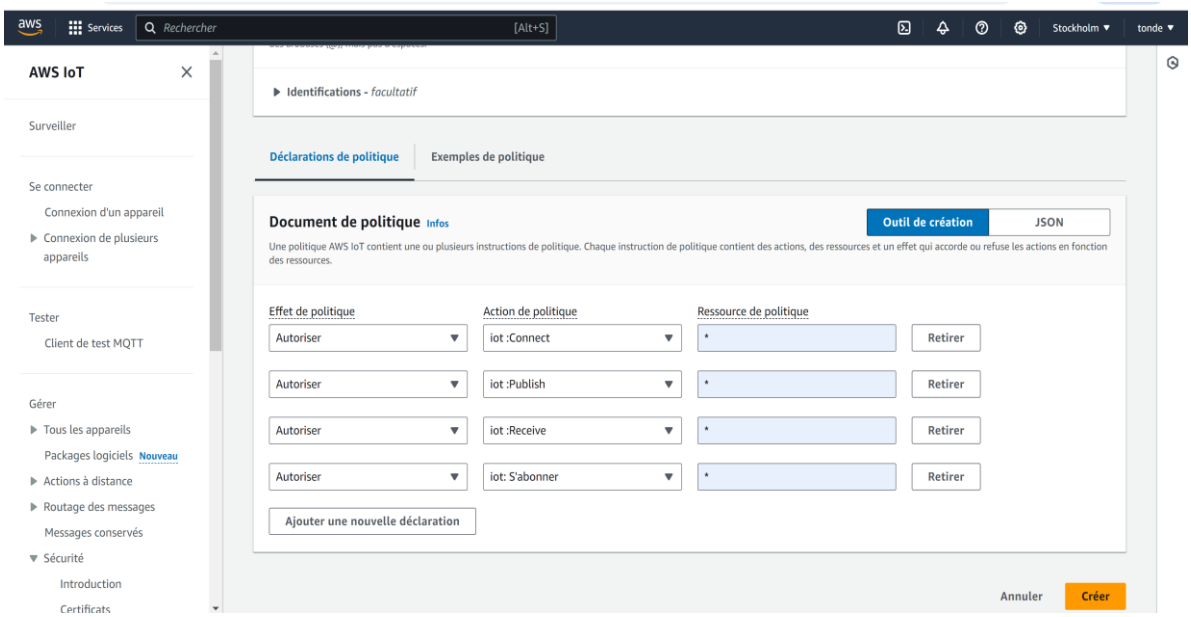


6. Créer et attacher une stratégie

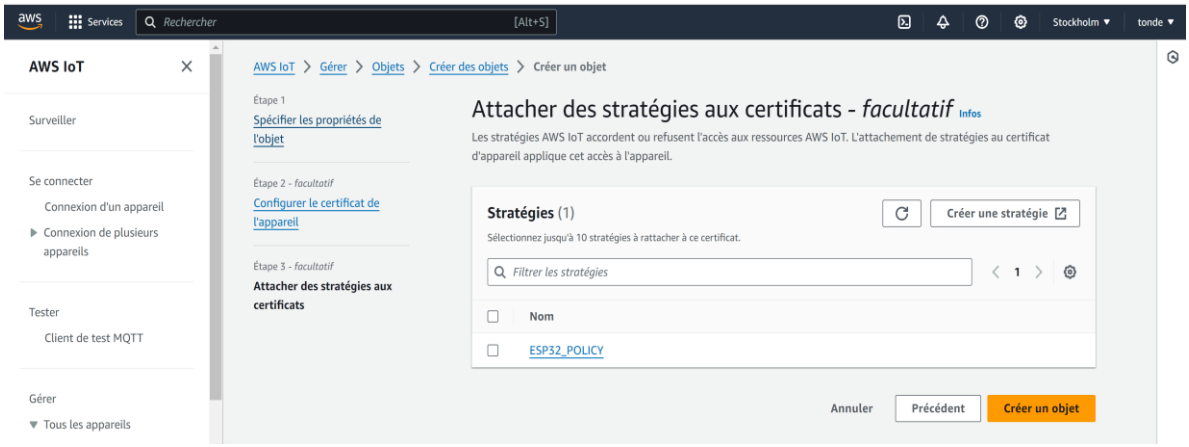
- Nous devons attacher une politique à l'objet que nous avons créé. Mais aucune politique n'est là pour le moment. Nous devons donc d'abord créer une politique que nous avons appelé « ESP32_POLICY ».



- Puis, nous remplissons la partie « Demande de Politique ». Dans cette partie, l'on définit les politiques suivant lesquels nos appareils IOT pourrons se lier à notre « objet » en fonction des ressources (l'option « * » que nous avons sélectionné indique que la politique décrite s'adresse à tout type de ressources). À partir de là, nous n'aurons plus qu'à sélectionner les actions de politiques : publier, nous abonner, nous connecter et recevoir (des explications plus détaillées de ces politiques sont dans la section « Exemples de politique »).

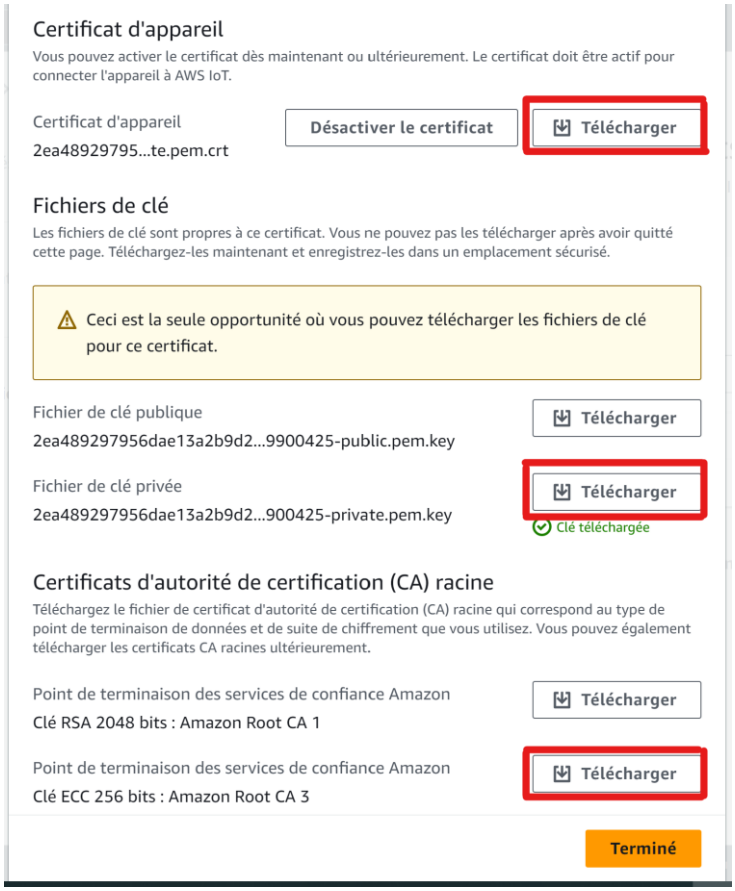


❖ Revenons maintenant à l'option « Créer un objet ». Une option de stratégie apparaîtra donc. Nous devons joindre les stratégies au certificat. L'on sélectionne donc la politique apparue et on clique sur « Créer un objet ».



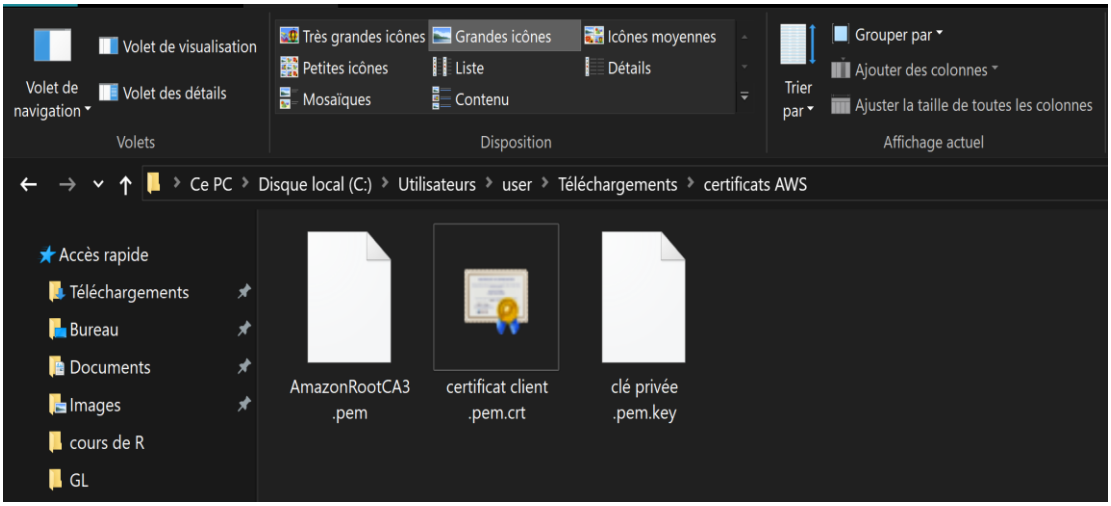
7. Téléchargement de certificats et de clés : sécurité

❖ Nous devons maintenant télécharger les certificats requis à partir de cette liste.



- **Certificat Du client**, puis nous l'avons renommé en tant que « certificat client » à des fins d'identification.
- **Clé privée** et renommez-la en « clé privée ».
- Certificat racine, il y a deux certificats ici. Mais nous avons choisit **la Clé ECC 256 bits : Amazon Root CA 3**. Les raisons de ce choix sont les suivantes :

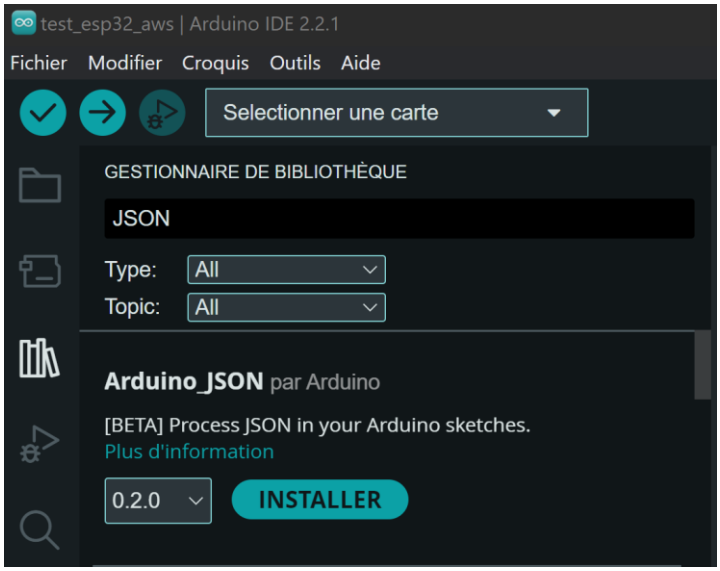
    - **Taille de clé réduite** : ECC offre un niveau de sécurité comparable à RSA avec une taille de clé beaucoup plus petite. Par exemple, une clé ECC de 256 bits est généralement considérée comme offrant un niveau de sécurité équivalent à une clé RSA de 3072 bits. Cette réduction significative de la taille de la clé permet une économie en termes de bande passante et de stockage.
    - **Rapidité des opérations** : Les clés plus petites et l'efficacité algorithmique d'ECC permettent des opérations de cryptage et de décryptage plus rapides, ce qui est particulièrement avantageux pour les systèmes où les ressources sont limitées ou pour les applications nécessitant une réponse rapide, comme les communications en temps réel.
    - **Consommation énergétique réduite** : L'utilisation de clés plus petites et d'opérations moins complexes signifie que ECC peut être plus économe en énergie, ce qui est crucial pour les dispositifs IoT et mobiles où la conservation de la batterie est essentielle.
    - **Conformité aux normes actuelles** : ECC est de plus en plus adopté et recommandé par de nombreux standards et organisations de sécurité. Son adoption dans des standards modernes et des protocoles tels que TLS 1.3 encourage son utilisation pour garantir la compatibilité avec les meilleures pratiques de sécurité.



III. Installation des bibliothèques Arduino nécessaires

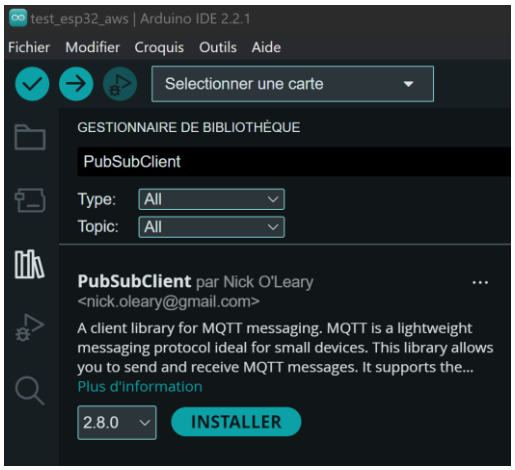
Apres l'installation des bibliothèques « WiFi.h » et « WiFiClientSecure.h » qui sont nécessaire pour connecter en toute sécurité l'ESP 32 a un Wifi et a d'autre ressources, nous installons d'autres bibliothèques :

1. Bibliothèque ArduinoJSON



- C'est une bibliothèque utile pour la gestion de données au format JSON (JavaScript Object Notation) sur des plateformes Arduino. Les données transiteront entre notre détecteur de présence et AWS IOT Core sous le format JSON.

2. Bibliothèque PubSubClient



- Nous utilisons la bibliothèque **PubSubClient** car nous développons une application IoT qui impliquent des communications MQTT (Message Queuing Telemetry Transport) ; qui est le protocole utilisé par AWS IOT Core. MQTT est un protocole de messagerie léger, idéal pour les dispositifs à faible bande passante et peu de ressources, souvent utilisé dans les systèmes de communication entre machines (M2M) et Internet des objets (IoT).

IV. Code source pour connecter AWS IoT Core à ESP32

Le programme qui **interface ESP32 avec le capteur PIR** et se connecte à Amazon **AWS IoT Core** est écrit dans l'IDE Arduino. Le code est divisé en deux sections. L'un est fait en un seul fichier ino principal.

1. On copie et on colle les contenus des différents certificats

```
//Contenu du certificat racine de AWS
const char aws_root_ca[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWljbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
)EOF";

 //Contenu du certificat client
 const char client_cert[] PROGMEM = R"KEY(
 -----BEGIN CERTIFICATE-----
 MIIDWTCCAkGgAwIBAgIUL6pyP3doHOCgDnSI3YkNtbuMJR0wDQYJKoZIhvcNAQEL
 BQAwTTFLMEkGA1UECwxCQW1hem9uIFdlYiBTZXJ2aWNlcyBPPUFtYXpvbi5jb20g
 SW5jLiBMPVNlYXR0bGUgU1Q9V2FzaGluZ3RvbiBDPVVTMB4XDTI0MDUwNTIxNTI1
 NloXDTQ5MTIzMTIzNTk1OVowHjEcMBoGA1UEAwwTQVdTIElvVCBDZXJ0aWZpY2F0
 ZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMiBECMLukW5P7rqkzOT
 G9rm6S9kbcP85deNhvGMkzO5U9tBW59cgIOHnAUxlZ1Sz2f90wbsariJKHwHJi4q
 TZvtwovN+eIgHwz64s88pSC0UT2PRYyxo56sVQqrD4FI5B5ouWKdMfnH8M388gjq
 kZLEV1sWnkx7cz4X/Ht6d3VuDLgE0LCdfhoNZAZ9JGZqYqf9OQ6PCWffMwmzwuq0
 khFlxh4pUvHLV0aSZ+DRoJ2bkx03fBKi1KEsatf8C76QDdIqlyTC2EDKUM6PrgyK
 msvo5FXcBocMTCqhVUKQVLsV1IJyR8uyGah5iOw7UfjHiRVdLM5qqN2ZyIjnSHtQ
 9g0CAwEAAaNgMF4wHwYDVR0jBBgwFoAUQ3xjgGj49zEC4R4JBSdQRgpWZYswHQYD
 VR0OBBYEFEf4coKn8fFvWer63lenLOKs8VIaMAwGA1UdEwEB/wQCMAAwDgYDVR0P
 AQH/BAQDAgeAMA0GCSqGSIb3DQEBCwUAA4IBAQCSYPK0AzsG9RD0GiyPLIVtZlOW
 Vr4y7J4HJxDZNdlDy84A3KpO5ekqAsX21SiZSJiQFGv+VGnu18tUnz805pKFB7BF
 wPDGhwCruOjfDv7fbeU6S97jNQz6ynAZYEdYPYk4S8aMor1KJBBKoKdBaVRJDg5y
 bN2oRBLpR7iYYwqh2M5Nw34R4fwgueSsiaXhoRqx2MWVjYq1NjnZovrDDUI6pEAI
 6N1mgxgmoqlIj+jAMFFUEF3FmG6R9U/4YwcN4BibQliTYug+4v0NgpcEopup1WRp
 wMtkM/bBSKcsyxNQ04eLpSVMtTRmR3ZEEf+ZV6Gt2nHagmXwbQd6dqR5nHey
 -----END CERTIFICATE-----
 )KEY";
```

```
48
49  //Contenu de la clé privée client
50  const char client_key[] PROGMEM = R"KEY(
51  -----BEGIN RSA PRIVATE KEY-----
52  MIIEpQIBAAKCAQEAyIEQIwu6Rbk/uuqTM5Mb2ubpL2Rtw/zl142G8YyTM7lT20Fb
53  n1yAg4ecBTGVnVLPZ/3TBuxquIkofAcmLipNm+3Ci8354iAfDPrizzylILRRPY9F
54  jLGjnqxVCqsPgUjkHmi5Yp0x+cfwzfzyCOqRksRXWxaeTHtzPhf8e3p3dW4MuATQ
55  sJ1+Gg1kBn0kZmpip/05Do8JZ98zCbPC6rSSEWXGHilS8ctXRpJn4NGgnZuTHTd8
56  EqLUoSxq1/wLvpAN0iqXJMLYQMpQzo+uDIqay+jkVdwGhwxMKqFVQpBUuxXUgnJH
57  y7IZqHmI7DtR+MeJFV0szmqo3ZnIiOdIe1D2DQIDAQABAoIBAAwN1k6FEfnHHVG7
58  cIYVPgN3BmXW/9JcjeTbmVVEQkVFywjPcRavlPpTWMnnJTuEEvwJgnpUFEZr2Ic8
59  VstHt8q8DIojI/a4P6WkH6ad2r+Ql6G/z0qaE2AitCokBpabLzu0yBBpqut9XbjE
60  kDiBAad1LltHjNTERzVgc8lon23ryHZSTgSNru4lYf/JsCs45Jnp9Mr56kQWW5tS
61  91JNydMPyNZdhUmyfUPFE4e0Yiovbjwv2ga77tPIyd95RXe/E870BXJjtH2wUR8i
62  x3WYkCXcDCkgu+gXOTNDpW3GxmrsqvGiqtYDWXB6AXUantQGAPbOQhfqEJw+7hRp
63  bi0ZcgECgYEA972C/FFOCe+a4IpwWDBUjcLFyZyzKEU3p0nzIW4ldRQFqDcDnpMs
64  eVARUlCgxOZShGxTFK5yJthx0MD7J4S+lHJ/GDdhfMREZWmUwxut308601L2uU/f
65  s9F+vQy/MHGXoLWA/PYlqY0rzJr09bna4s9Uvltw19vJZQxOKnAvlS0CgYEAzzBi
66  4S54CA8jPDJCVvAahXGny6XsLB2tBg1LgS03x22MC9SKkVWJ4fl4jZXqC17Q9Pxj
67  Fyy0WI5G1uOZZAefV7cBwZCFjlYBctZNZZatDsHM3PSLDLUFhsX6Q+kiN6tGM+kq
68  swJeeLHSKH47cvHwvmQ7UCjk/Y86iTb6Jrc1MGECgYEAmsGEkG+Nq0FE4103fq96
69  Mi/oc9/yPoa2lEbqnQh8V52RRxpOpYFUN9ffaic5tqYWB+jPIU0zu7ZbqsbSXUZb
70  8slvQUZfHJxSCXOh3RtuSt/oZFggDXSqzl5gqGHNEMr0qAyZretpbXflU2O8/78A
71  z8wqKNtgK0d9kCTWs0XvQ2UCgYEAmd81kEcSnemgIXSyEB8Bk1l1ZKk1XQ4tEfGN
72  wcHzSwjwmVigEI+Wd1zvSzerOgaQ6WEHto5c4efIdI9TvBZtIJepljfe+HanXMIL
73  eQcOP4duvipfmnBWSmF+HDFoWC+7U5I3Q/reboShr7EQNa25Pazy3/V/G1DkCEPs
74  RBSMTAECgYEA5GtKXib8Z5r3M5bwMbaVmZAnptJ4Ppmtq9uyVHnRucM1OK2tLZXQ
75  eAI74R7SHUtWWAPLz1/+pZfMmfYbh2FkhCQ7WYpZRL8TTNLSZ3zf29FKQqu9VGR5
76  UU4SvHB4c0XLVDnnoE7PPMQ8HdnJVCsVvBDChlkJiWBR0RUNCKVpL4U=
77  -----END RSA PRIVATE KEY-----
78  )KEY";
79
```

2. Importation des bibliothèques

```
80  //Importation des bibliothèques
81  //WiFi.h : connectivité wifi pour l'accès à internet
82  //WiFiClientSecure.h : Pour l'établissement d'une connexion sécurisée à AWS, avec les certs/clés
83  //PubSubClient.h : Pour la connexion effective à AWS IoT et l'échange de données
84  //ArduinoJson.h : Pour l'encodage des données au format json
85  #include <WiFi.h>
86  #include <WiFiClientSecure.h>
87  #include <PubSubClient.h>
88  #include <ArduinoJson.h>
89
90  //Paramètres de connexion au point d'accès WiFi
91  const char* ssid = "Princesse Lili Moayé";
92  const char* password = "19cca8a27495";
93
94  //Endpoint aws
95  const char* aws_endpoint = "a15v1mdbyec2qy-ats.iot.eu-north-1.amazonaws.com";
96
97  WiFiClientSecure wifi_client;
98  PubSubClient client(wifi_client);
99
100 //Port de connexion du capteur PIR à l'ESP32
101 const int capteur = 26;
```

- Le point de terminaison AWS IoT « aws_endpoint » que nous avons inséré est obtenu dans la partie des paramètres du tableau de bord d'AWS.

3. Connexion à AWS IOT

```
103  //Procédure permettant la connexion à AWS IoT
104  void connectAWS() {
105    //Chargement des certificats/clés
106    wifi_client.setCACert(aws_root_ca);
107    wifi_client.setCertificate(client_cert);
108    wifi_client.setPrivateKey(client_key);
109
110    //Chargement de l'endpoint et du port de connexion
111    client.setServer(aws_endpoint, 8883);
112
113    //Boucle de tentatives de connexion
114    while (!client.connected()) {
115      Serial.println("Connecting to AWS...");
116      if (client.connect("ESP32Client")) {
117        Serial.println("Connected!");
118      } else {
119        Serial.print("Connection failed, rc=");
120        Serial.print(client.state());
121        Serial.println(" trying again in 5 seconds");
122        delay(5000);
123      }
124    }
125  }
126
```

4. Se connecter au Wifi et envoyer une alerte à AWS IOT Core

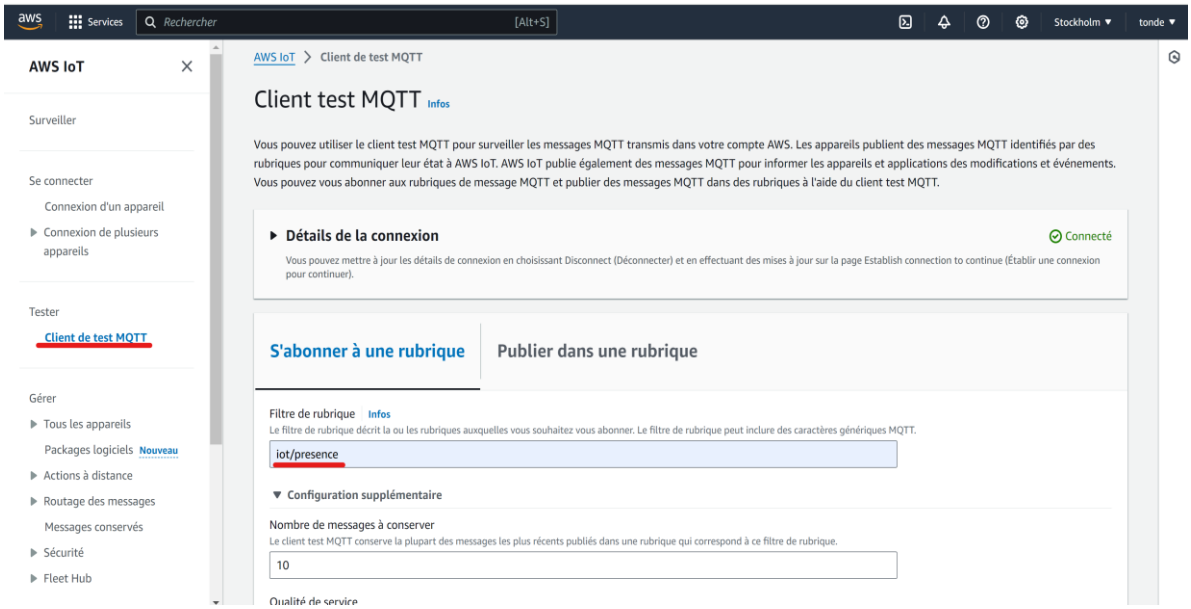
```
127  void setup() {
128    Serial.begin(115200);
129
130    //Boucle de connexion au wifi
131    WiFi.begin(ssid, password);
132    while (WiFi.status() != WL_CONNECTED) {
133      delay(500);
134      Serial.print(".");
135    }
136    Serial.println("Connected to WiFi");
137
138    connectAWS();
139    pinMode(capteur, INPUT);
140  }
141
142  //Procédure permettant d'envoyer un message sur AWS IoT
143  void publishMessage(){
144    StaticJsonDocument<200> doc;
145    doc["message"] = "Présence détectée";
146    char jsonBuffer[512];
147    serializeJson(doc, jsonBuffer);
148    client.publish("iot/presence", jsonBuffer);
149  }
150
151  void loop() {
152    //Permet de se reconnecter à AWS si jamais la connexion est perdue
153    client.loop();
154    if (!client.connected()) {
155      connectAWS();
156    }
157
158    //Envoie un message sur AWS IoT à chaque détection de présence
159    if(digitalRead(capteur)){
160      Serial.println("Presence");
161      publishMessage();
162    }
163  }
164
```

Apres avoir écrit le code dans l'IDE Arduino, nous l'avons compilé puis téléverser dans notre carte ESP 32.

V. Test : on envoie des donnees à AWS IOT Core

- Apres avoir connecter notre carte ESP 32 à une source d'alimentation et l'avoir connecté au point d'accès Wifi dont nous avons spécifié les paramètres dans le code, nous pouvons

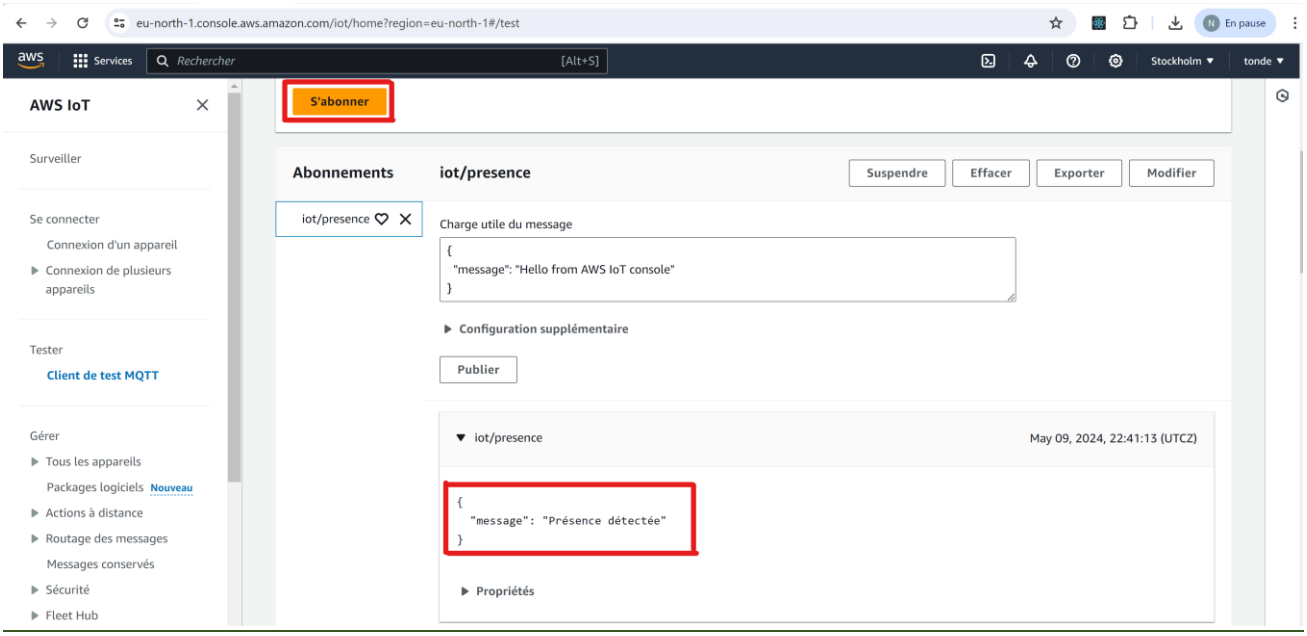
maintenant nous rendre sur le tableau de bord de AWS IOT Core pour effectuer les tests de notre détecteur de présence. Pour se faire on clique sur « Client de test MQTT » puis sur « S'abonner à une rubrique ».



- La mention « iot/presence » correspond au premier paramètre de la méthode « publish »

```
148     client.publish("iot/presence", jsonBuffer);
```

- On clique sur « S'abonner » et on obtient le message « Presence détectée » qui est envoyé par notre carte ESP 32 lorsque le capteur PIR détecte une presence.



VI. Problèmes et suggestions

Nous n’avons rencontré aucun problème dans la réalisation de ce projet. Par conséquent nous n’avons aucune suggestion particulière.

CONCLUSION

En conclusion, ce projet a démontré comment les technologies cloud et IoT peuvent être intégrées efficacement pour créer un système de sécurité innovant et robuste. En utilisant AWS IoT Core, nous avons mis en place une solution capable de traiter les signaux de présence détectés par des capteurs de mouvement, de les transmettre de manière sécurisée au cloud, et de gérer les données pour une surveillance en temps réel. Le choix d'AWS IoT Core a offert plusieurs avantages, notamment en termes de sécurité des données, de facilité de déploiement et de gestion, ainsi que de scalabilité. La capacité de gérer un grand nombre de dispositifs et de messages avec peu de latence a été essentielle pour la réussite du projet. À travers ce projet, nous avons acquis des compétences précieuses en matière de développement cloud et IoT, tout en contribuant à l'avancement des solutions de sécurité intelligente. L'expérience acquise pave la voie pour de futures recherches et développements dans le domaine de l'IoT et des applications cloud, en visant des solutions toujours plus intégrées, sécurisées et efficaces pour le bien-être et la sécurité des personnes et des biens. En définitive, le projet a non seulement atteint ces objectifs mais a également ouvert de nouvelles perspectives sur l'utilisation des technologies cloud pour améliorer les systèmes de détection de présence, soulignant l'importance de continuer à explorer et à innover dans le domaine de l'Internet des Objets.