Projet IOT

Explore the docs »

Le projet:

Schema

Le Fonctionnement

Nous avions pour but de créer un système IOT à l'aide de 3 M5stacks et de 3 Capteurs. Les capteurs sont: - Capteur PIR - Capteur light - Capteur TCS

Pour commencer, nous avons décidé de développer ce système pour être utilisé par la sécurité du musée. En effet, bien qu'il s'agisse de capteur aux données peut sensible, il est nécessaire que ces informations restent entre de bonnes mains. Afin de mener à bien le projet, notre reflexion s'est portée sur l'éthique et la sécurité de l'IOT dans un musée. Il a été nécessaire de bien comprendre les enjeux du système déjà existant, et les risques que nos apports peuvent amener. Comme dit plus tôt, nous avons défini que les données transmises par nos capteurs sont peu sensibles. Leur accès peut tout de même être utilisée à des fins malicieuse. (ex: éteindre la lumière pour voler une oeuvre). C'est pourquoi le contrôle des accès et l'encryption des données et cruciale à l'implementation de chaque dispositif que nous avons implémentés. Nous avons ensuite developpés un bot Telegram pour proposer une interface utilisateur. Enfin, nous avons utilisés packet tracer afin de simuler la surveillance du musée. Celui-ci contient

une Garage Door representant l'état du musée (Ouvert / Fermé) et une alarme. Ceux-ci peuvent être controlés via le Bot Telegram, ou par un ordinateur connecté au réseau simulé.

Voici une explication procédurale du fonctionnement de notre projet: - Les capteurs transmettent leur changement d'état à un Broker MQTT - Un client MQTT récupère les informations et les répartis en conséquences - Chaque log est envoyé à un serveur de log via TCP - Les status actuels des capteurs sont envoyés via UDP - Un bot telegram sert d'interface à l'utilisateur pour: - Afficher les logs par une requête TCP au serveur de log - Afficher l'etat actuel par une requete UDP - Controler les périphériques packet tracer

M5 Stack

- Capteur de présence:
  - L'objectif du capteur de présence est d'identifier le nombre de passage à l'entrée du musée.
  - Lorsque une personne passe devant le capteur, l'appareil renvoit la valeur 1. Dans le cas contraire, renvoit 0.
- Capteur de lumière:
  - Le capteur de lumière permet de capter le pourcentage de luminosité dans une pièce.
- Capteur de couleur:
  - Ce capteur permet quant à lui d'identifier quelle est la couleur actuellement déployée par les ampoules led.
- Les Main.py
  - Tous les codes des fichiers à l'intérieur des M5 ont été repris, et légèrement modifier dans le Main respectif de chaque appareil. Les commentaires permettent une meilleures compréhension du code. Ces dernier permettent un affichage sur l'écran de chaque appareil avec les valeurs retournées par les capteurs. Ainsi, chaque appareil est facilement reconnaissable de l'exterieur.

NB :

L'éthique et la sécurité des appareils IoT (Internet des objets) sont des préoccupations majeures dans le développement et l'utilisation de ces technologies. Les appareils IoT sont connectés à Internet, collectent des données sensibles et interagissent avec l'environnement physique, ce qui soulève des questions importantes sur la confidentialité, la sécurité et l'intégrité des informations.

Dans le contexte du M5, il est important de souligner que les code fourni précédemment ne mettent en place aucun protocole de sécurité ou de cryptage spécifique. Cela a été fait dans le but de faciliter les tests et la compréhension du fonctionnement du code. Toutefois, il est essentiel de comprendre que dans un contexte réel, la sécurité et la protection des données doivent être mises en place de manière rigoureuse.

Dans un déploiement réel de ces appareils IoT, plusieurs mesures de sécurité devraient être prises. Cela inclut l'utilisation de protocoles de communication sécurisés tels que MQTT avec TLS/SSL, l'authentification et l'autorisation des utilisateurs et des appareils, le chiffrement des données sensibles, ainsi que la gestion des mises à jour et des correctifs de sécurité.

La protection de la vie privée des utilisateurs est également un enjeu éthique crucial. Les données collectées par les appareils IoT doivent être traitées de manière responsable, en respectant les règles de confidentialité et en garantissant le consentement éclairé des utilisateurs.

Il est important de souligner que l'absence de protocoles de sécurité dans le code fourni ne reflète pas les pratiques recommandées dans un environnement réel. L'objectif ici était de simplifier le code pour les besoins de l'exemple. Dans un contexte réel, la sécurité et l'éthique doivent être considérées comme des priorités absolues, et des mesures appropriées doivent être prises pour garantir la confidentialité, l'intégrité et la disponibilité des informations et des systèmes IoT.

(back to top)

Serveurs et Protocoles

Pour chaque Serveurs/VM ou protocole utilisé, l'éthique doit être au centre de la reflexion. En effet, il est important de faire attention à la sécurité du système, à la protection des données et aux différentes question d'éthique liés à notre projet et à l'IOT.

Serveur de logs (Protocole TCP)

Afin de créer un server de logs, nous avons décidé d'utiliser Ubuntu Server 22.04 LTS. Ce choix part du besoin d'une distribution simple, efficace et facile à maintenir. De plus, nous avons pas besoin de Desktop Environnement puisque les informations du server ne sont que du texte. Nous avons choisi cette méthode (via TCP) plutôt que de faire un client MQTT qui enregistre directement les communications pour l'extensibilité du projet. En effet, les logs pertinent à un musée ne comprennent pas seulement des appareils communicant avec MQTT. Les logs pourrait comprendre des activités sur des ports/protocles différents. C'est pourquoi nous préférons centraliser les données quelle que soit leur provenance. De plus cela permets potentiellement de croiser automatiquement les différentes activités afin de créer des logs plus détaillés.

Les points de réfléctions importants du setup du serveur TCP sont: - Mise en place d'un parre-feu autorisant uniquement le port de notre communication TCP. - Création d'un script python recevant les informations du broker via TCP. Ensuite, le script affiche sur la console le log reçu et l'inscrit dans un fichier mqtt_logs.log. - Configuration du serveur pour utiliser des connexions

chiffrées en utilisant le protocole SSL/TLS. Cela permettra de garantir la confidentialité et l'intégrité des données transitant entre les clients et le serveur. - Implémentation d'une politique de mots de passe forts pour les utilisateurs du serveur et stockage de manière sécurisée, par exemple en utilisant des techniques de hachage et de salage des mots de passe. - Limitation de l'accès au serveur en configurant un pare-feu pour n'accepter les connexions que depuis des adresses IP autorisées. - Mise en place des mécanismes de surveillance des journaux d'accès et d'activité du serveur afin de détecter toute activité suspecte ou tentative d'intrusion.

Serveur de query (Protocole UDP)

Pour permettre aux différentes interface communicant avec les différents devices de notre système d'accéder rapidement aux informations des-dits appareils, le choix s'est porté sur un server de query. Celui-ci reçoit régulièrement le status des différents capteurs et périphérique. Les stock dans son code python, et les retourne lors d'une query. UDP permet l'envoie presque instané de l'information. Puisque les query sont fréquentes et que les status changent presque aussi souvent, les données nous pas besoin d'avoir une integrité parfaite à chaque communication. C'est pourquoi le choix s'est porté sur UDP plutôt que TCP pour les query.

Les points de réfléctions importants du setup du serveur UDP sont: - Mise en place de communications UDP entre les appareils et le serveur sécurisées en mettant en œuvre des mécanismes de chiffrement appropriés, tels que l'utilisation de protocoles VPN (Virtual Private Network) pour établir des connexions sécurisées. - Mise en place des mécanismes d'authentification et d'autorisation pour les appareils du réseau afin de garantir que seuls les appareils autorisés peuvent envoyer des informations au serveur. - Verification régulière des ports UDP ouverts sur le serveur pour identifier et fermer les ports non-nécessaires, réduisant ainsi la surface d'attaque potentielle. - Implémentation des mécanismes de journalisation pour enregistrer les activités des appareils du réseau et surveiller les comportements anormaux ou les tentatives d'intrusion.

Les clients

Bot Telegram (Protocole HTTP)

Afin de permettre à un utilisateur tel qu'un agent de sécurité, nous avons décidé d'utiliser un bot Telegram. L'user experience similaire à une télécommande nous semblait adaptée au besoin. De plus, le protocole HTTP permet un envoi et une récéption rapide des informations. Le problème principal de cette décision est vis-à-vis de la sécurité. En effet, les bots Telegram n'utilisent pas le même protocole d'encryption end-to-end que le service de Messagerie. Pourtant, puisque seul les 5 derniers logs, le status actuel d'une lampe, et le nbr de visiteurs sont disponible, nous avons estimé que le risque était acceptable. Il

faut tout de même noter que le risque demande une gestion afin de limiter les dégats s'il vennait à se réaliser. Cependant, plus tard dans le projet, nous avons mis en place une simulation packet tracer du musée. Celle-ci se controlle, elle aussi, par le bot Telegram. Dans ce nouveau cas, le risque est bien plus grand, et ne peut donc être pris que dans le cas de la Simulation. Dans un cas réel, l'activation/desactivation de ces contrôles ne devraient être faites uniquement "on-site" ou dans un environnement plus sécurisé. Pour limiter les risques, le bot n'accepte les connexions que d'utilisateurs dont l'ID a été préalablement ajouté à une liste.

Le bot à été codé en python à l'aide de la librairie python-telegram-bot. Et utilise asyncio pour permettre à plusieurs utilisateurs d'utiliser le bot en même temps.

Afin d'host le bot, nous utiliserions un VPS (tel qu'une machine sur linode) dans un cas réel. Il est nécessaire de prendre en compte que le bot demande une quantité de ressource proportionnelle au nombre d'utilisateurs. Dans le cas du projet, le bot est host sur sa propre machine/VM.

Client MQTT

Le client MQTT consitue le coeur du fonctionnement de notre projet. Celui-ci surveille les communications de certains topics de notre broker. Celui-ci se connecte au Broker à l'aide d'un mot de passe car l'accès au broker est limité. Pour faciliter la demonstration, nous avons décidé de retirer les identifiants.

Le client forme alors une texte de log qu'il envoie ensuite par un socket TCP au serveur de log. Il analyse ensuite le topic sur lequel le message a été posté. Selon le le topic, il assigne la valeur reçue à un capteur et envoie un message via un socket UDP au serveur de query.

Plutôt qu'un client, ce code agit comme un manager pour analyser, trier et repartir les données de notre système IOT.

(back to top)

Packet Tracer

Topologie du Packet Tracer comprend les éléments suivants : 2 PC, 3 routeurs, 1 commutateur, 1 serveur, 6 objets IoT, 1 carte à base de système (SBC) et 1 microcontrôleur (MCU). - Protocole RIP : Le protocole RIP (Routing Information Protocol) est utilisé pour la configuration du routage dynamique dans le réseau. Il permet aux routeurs d'échanger des informations de routage pour déterminer les chemins les plus courts vers différentes destinations du réseau. Grâce au RIP, une communication fluide et une connectivité sont assurées, avec des mises à jour automatiques des tables de routage des routeurs. - Réseau en DHCP : Le réseau est configuré pour utiliser le protocole DHCP (Dynamic Host Configuration Protocol), ce qui permet aux appareils connectés d'obtenir

automatiquement une adresse IP. Ce qui élimine la nécessité d'une configuration manuelle des adresses IP sur chaque appareil. Cela facilite la gestion du réseau et offre une flexibilité lors de l'ajout ou de la modification des appareils connectés. Les pools DHCP sont configurés sur le routeur R2 et sont liés à une interface loopback avec l'adresse IP 192.168.200.1/24. - Serveur : Le serveur0 est configuré avec une adresse IP 192.168.1.77/24 et avec une interface HTTP pour la création de comptes, y compris le compte administrateur. La sécurité du système et la protection des données sont des éléments essentiels dans notre réflexion. Nous avons pris des mesures pour garantir que les comptes utilisateurs sont créés de manière sécurisée et que les autorisations sont correctement gérées. La configuration du serveur avec une interface HTTP permet aux utilisateurs d'accéder de manière sécurisée aux fonctionnalités du système.

Pour rester dans le thème système IOT domotique:

- PC : Nous avons choisi d'utiliser un PC pour permettre aux utilisateurs de contrôler les objets IoT par le biais d'un navigateur Web. Cette approche offre une interface familière pour les utilisateurs. La sécurité et la confidentialité des données sont prises en compte pour garantir que seuls les utilisateurs autorisés ont accès aux fonctionnalités du système.
- Interrupteur marche/arrêt pour les caméras de sécurité : Un interrupteur est configuré pour activer ou désactiver les caméras de sécurité du système. Il se trouve directement dans la salle de sécurité. Cela permet aux utilisateurs de contrôler facilement les caméras en fonction de leurs besoins de sécurité. Nous avons pris en compte les questions d'éthique liées à la protection de la vie privée et à la surveillance en veillant à ce que l'activation des caméras soit soumise à des autorisations appropriées et à des politiques de confidentialité claires.
- Les fonctionnalités du système peuvent être contrôlées à la fois via une interface frontend web et via un bot Telegram. Pour permettre aux utilisateurs de contrôler les fonctionnalités du système. Les données nécessaires sont récupérées à partir d'un MCU programmé en utilisant une socket UDP en temps réel en Python. Dans notre réflexion éthique, nous avons veillé à protéger la sécurité du système, la confidentialité des données et à prendre en compte les différentes questions d'éthique liées au transfert des données.
- Contrôle d'ouverture et de fermeture du musée en fonction de la présence de lumière : Le musée est simulé par un "Iot Garage Door", qui affiche l'ouverture et la fermeture du musée en fonction de la présence de lumière. Lorsque le message "toggle_light" est reçu sur le MCU, le système réagit en conséquence. Cette décision a été prise dans un souci d'efficacité énergétique et de sécurité. En surveillant la présence de lumière, nous pouvons optimiser l'utilisation des ressources et assurer la sécurité en évitant l'ouverture du musée lorsque la lumière naturelle est suffisante.
- Contrôle de l'activation de la sirène : La sirène du système peut être activée ou désactivée en envoyant un message "toggle_alarm" via une socket

UDP. Dans notre réflexion éthique, nous avons pris en compte les impacts sonores sur l'environnement et les nuisances potentielles pour les personnes à proximité. L'activation de la sirène est soumise à des autorisations appropriées et à des politiques de sécurité pour éviter les utilisations abusives ou non nécessaires.

Nous avons conçu et mis en œuvre ces fonctionnalités en gardant à l'esprit l'importance de l'éthique. La sécurité du système, la protection des données, la confidentialité, la vie privée et les questions environnementales ont été prises en compte tout au long du processus de développement. Nous avons adopté des mesures de sécurité appropriées, telles que l'authentification des utilisateurs, le chiffrement des communications et la gestion des autorisations, afin de garantir une utilisation responsable et éthique du système.

Contact

Mathis Bourinet - mathis.bourinet@hes-so.ch

Kenan Henzelin - kenan.henzelin@hes-so.ch

Julien Frey - julien.frey@hes-so.ch

Project Link: https://github.com/Takeapill/IOT

(back to top)

Acknowledgments

- Python Telegram API
- Mosquitto
- Packet Tracer Real Servers

(back to top)