**IoT CO2**

By

Schlumberger

**Introduction :**

La qualité de l’air intérieur, qui est un important sujet de santé publique avec les divers polluants issus de notre environnement, a pris une importance cruciale ces derniers mois avec la découverte de la transmission par aérosol de la Covid 19. L’aération naturelle préconisée pour éviter la concentration du virus dans des espace clos n’est toutefois pas toujours possible. À SRPC, l’équipe HSE a condamné certaines salles de réunion car nous n’étions pas certains que l’aération y était suffisante. La surveillance du taux de CO2 est un moyen de vérifier la qualité de l’aération d’une zone utilisée par des groupes, et faire des préconisations sur son utilisation. Afin d’éviter de nombreuses et fastidieuses mesures ponctuelles, les ingénieurs impliqués dans le FabLab de SRPC ont décidé de développer un prototype de capteur de CO2 connecté, permettant une mesure en temps réel avec transmission WiFi des données acquises. Ce développement IoT pose toutefois l’enjeux de la sécurité informatique sur lequel nous vous proposons de travailler.

**Mots-clés :** Définitions de mots et concepts qui seront utiles lors de l’épreuve

* IoT : Internet Of Things
* MQTT : protocole de messagerie publish-subscribe basé sur le protocole TCP/IP
* TLS : Transport Security Layer, qui assure l’authentification du serveur, la confidentialité et l’intégrité des données.
* Flask : Micro-Framework open source de développement web en Python.
* OTA programming: Mise a jour du firmware à distance (Over-the-air)
* Arduino : plateforme de prototypage open source pour microcontrôleur.
* ESP32 : microcontrôleur intégrant WiFi, BlueTooth, etc…
* PostgreSQL : Système de Gestion de Base de Données Relationnelle open source.
* Platform IO : IDE basé sur Visual Code pour le développement de firmware.
* Python, C++ : langages de programmation utilisés pour ce projet.
* SINET : Schlumberger network

**Conditions de travail :**

Le produit est conçu pour fonctionner à l’intérieur sans contraintes climatiques particulières. Il doit pouvoir fonctionner sur batterie en autonomie pendant un mois, ou connecté à une prise USB. Il doit pouvoir être produit facilement dans un FabLab, avec une simple imprimante 3D. Ses composants (capteur, microcontrôleur) doivent pouvoir être éventuellement réutilisés pour d’autres projets. Le cout total de chaque unité doit être inférieur à 100 euros. Une trentaine pourront fonctionner simultanément sur un même site. L’environnement logiciel doit être Open-Source et « gratuit ».

**Tache :**

Un prototype du capteur a déjà été produit.Text

Description automatically generated with low confidence

Wifi antenna

[e-paper ESP32 kit](http://www.lilygo.cn/prod_view.aspx?TypeId=50061&Id=1393&FId=t3:50061:3)

magnet

e-paper screen

A picture containing text, electronics

Description automatically generated

USB

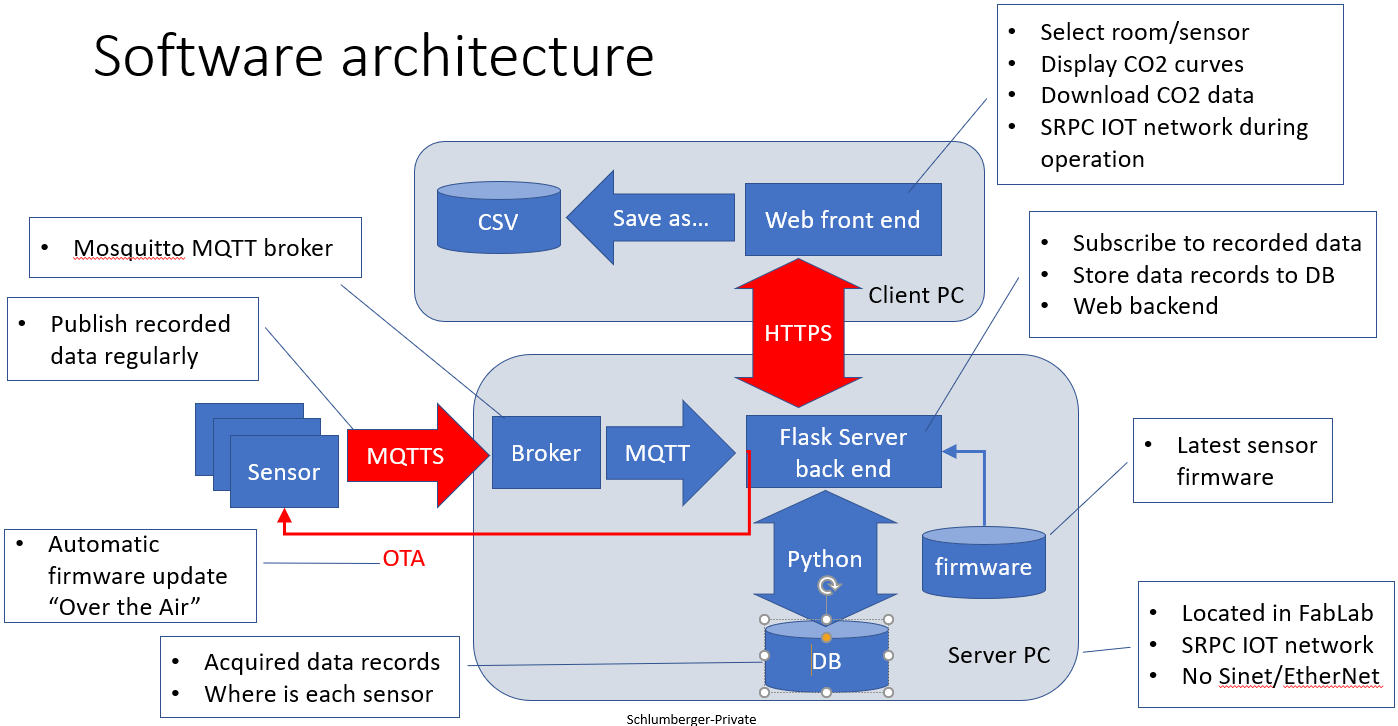
SD card slot

18650 battery holder

[Sensirion SCD41 kit](https://developer.sensirion.com/sensirion-products/scd4x-co2-sensors/) CO2 sensor

3D printed casing

L’architecture logicielle est en cours d’implémentation :



Le travail à réaliser consiste à élaborer et implémenter une stratégie cohérente permettant de sécuriser raisonnablement les communications suivantes :

1. Communication via le protocole MQTTS entre chaque capteur et le serveur. Pour l’instant, seul MQTT a été implémenté, sans login/passwd sur le broker…
2. Mise à jour par OTA du firmware des capteurs. Jusqu’à présent, l’OTA est réalisé via HTTP.
3. Sécurisation de la communication entre le Web front-end et le back-end Flask. Pour l’instant, il n’y a que du HTTP…

**Objectifs :**

L’objectif consiste à proposer une solution IoT difficilement « hackable », prenant en compte tous les aspects du problème, de la mise en service jusqu’à la maintenance. L’équipe étudiera prioritairement les points suivants :

1. Sécurisation des communications à la « frontière nord », entre le serveur Flask et le client web, en utilisant TLS pour passer de http à https.
2. Sécurisation des communications à la « frontière sud », en utilisant TLS dans le protocole MQTT entre les capteurs et le broker Mosquitto.
3. Sécurisation de la mise à jour du firmware « Over the Air » avec TLS.
4. Implémentation d’un contrôle d’accès des pages web par login/password.

**Sous-Objectifs :**

Au-delà des communications mentionnées ci-dessus, un « modèle de menaces » est fourni. L’équipe essaiera de proposer des réponses afin de minimiser le risque pour chaque menace envisagée, par une implémentation spécifique et/ou une procédure.

L’équipe pourra donc proposer et implémenter une solution simple pour la production et le déploiement des capteurs, notamment la mise en place des informations d’identification et les éventuels certificats utilisés par les capteurs. À cet effet, une carte SD pourrait éventuellement être utilisée au cours du déploiement.

L’équipe pourra prendre en considération les problèmes de sécurité liés à l’accès physique d’un capteur installé dans une zone de passage du site. On envisagera par exemple les scenarii suivants :

* Un individu charge un nouveau firmware dans le capteur pour simuler un « exploit » à l’occasion d’une importante visite de client, afin de décrédibiliser la société.
* Un individu remplace un capteur installé dans une salle de réunion par un faux, aux fonctionnalités identiques, mais pourvu d’un microphone pour faire de l’espionnage industriel.

**Liens utiles**:

* [ESP32 TLS](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mqtt.html)
* [ESP32 Flash Encryption](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/flash-encryption.html)
* [[ESP32 Secure Boot](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v2.html)](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v2.html)
* [ESP32\_ECO\_V3](https://www.espressif.com/sites/default/files/documentation/ESP32_ECO_V3_User_Guide__EN.pdf) [le microcontrôleur utilisé est le ESP32-D0WDQ6-V3]
* [HTTPS in Python](https://realpython.com/lessons/exploring-https-overview/)
* [Flask](https://flask.palletsprojects.com/en/2.0.x)
* [Flask\_mqtt](https://flask-mqtt.readthedocs.io/en/latest/usage.html)
* [CO2 sensor](https://developer.sensirion.com/sensirion-products/scd4x-co2-sensors/)
* [CP210x driver](https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers)
* [Platform IO](https://platformio.org/)
* [Lilygo t5 e-paper](http://www.lilygo.cn/prod_view.aspx?TypeId=50061&Id=1393&FId=t3:50061:3)

**Solution et exigences :**

* Implémentation avec code source relâché sur github.
* Etude des brèches de sécurité selon le « modèle de menace ».
* Présentation power point.

**Ressources et matériel**:

* Prototype du capteur ou mockup
* [Code source du firmware (C++)](https://github.com/ChristopheLaurent/AirQuality/tree/main/AirQualitySensor)
* [Code source du serveur (Python/Flask)](https://github.com/ChristopheLaurent/AirQuality/tree/main/AirQualityBackend)
* L’équipe aura la charge d’installer les logiciels open source suivants selon le document « prerequisite »:
  + PlatformIO
  + Installation du driver CP210x
  + Python 3
  + Flask
  + PostgreSQL

**Présentation :** A l’issue de l’épreuve, l’équipe fera une présentation power-point des solutions proposées pour réduire chaque menace, et fera une démonstration de l’implémentation réalisée. Les transparents seront rédigés en anglais, mais la présentation pourra être faite en français. Le temps imparti pour la présentation et la démonstration sera de 15 minutes, suivie d’une brève session de question réponse.

**Règles générales :** Il est vivement conseillé de faire toutes les installations requises selon le document « prerequisites », avant le début de l’épreuve. Chaque équipe relâchera son code source à l’issue de l’épreuve dans sa propre branche du [repo github](https://github.com/ChristopheLaurent/AirQuality).

**Pénalités et disqualification :**  Sans objet.

**Critères d’évaluation :** A chaque menace du modèle est associé un nombre de points. Les points seront accordés selon la pertinence de chaque réponse apportée (en anglais) dans la colonne « Remediation », et son implémentation. La somme des points obtenus constituera l’essentiel de l’évaluation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Critères à évaluer | Poids relatif (en pourcentage proportionnel à l’importance du critère) | Commentaires |
| Idée | Originalité/degrés d’innovation | 10 % | A quel point la solution a un caractère unique/innovant comparée aux autres ? |
| Accomplissement de la tache donnée | 80 % | Réalisations par rapport aux autres équipes : la solution s'attaque-t-elle à chaque sous-objectif défini pour la tâche ? Dans quelle mesure est-elle efficace par rapport aux autres solutions ? |
| Présentation | Pitch | 10 % | La présentation était-elle complète ? Est-ce que l'équipe a fait preuve de confiance envers ses idées ? Leurs idées étaient-elles bien fondées et expliquées ? Est-ce qu’ils ont vendu leur produit de manière convaincante ? |
| Sous-total (somme des points obtenus en pourcentage) |  |  |  |
| Pénalités |  |  | Si obtenues |
| Total (Sous-total moins les pénalités) |  |  |  |