1. PRESENTATION GENERALE DU PROJET

Vous devez concevoir un projet IoT répondant aux critères ci-dessous. Le projet sera réalisé individuellement.

Votre projet doit être innovant, actuel. L'objet IoT doit être **utile** et **utilisable**. Imaginez un scénario réaliste. Soyez ingénieux! Vous être le fondateur d'une startup et vos clients potentiels doivent avoir envie d'acheter votre objet connecté. Cela n'a pas de sens de prévoir un objet connecté qui coûte des centaines d'euros pour protéger une ressource qui coûte cinquante euros...

Cet objet connecté permettra de solutionner un problème que vous avez identifié dans un domaine spécifique. Les domaines d'utilisation de l'IoT sont variés et nombreux : Santé, Energie, Environnement, Mobilité, Education, Loisirs, Sports, Animaux de compagnie, Commerce, Agriculture,…

Il est souhaitable de parler de votre projet à un professionnel du domaine choisi. Par exemple, si votre objet concerne l'agriculture, les soins de santé,… , il est utile d'avoir un retour sur l'utilité de votre objet.

Une fois que vous avez défini un problème que vous voulez solutionner, vous devez trouver un **titre** pour votre projet. Ce titre doit être accrocheur et on doit pouvoir tout de suite savoir à qui s'adresse l'objet connecté (utilisation de "connecté" ou "smart" ou ...).

Votre startup va réaliser un prototype d'objet connecté que vous souhaitez commercialiser. Dans ce cadre, L'architecture de votre projet IoT doit comprendre **plusieurs niveaux.**

- Niveau 1 : les objets connectés et leurs capteurs : c'est à ce niveau que l'on retrouvera les cartes ESP32-WiFi, ESP32-LoRa et ESP32-H2-Thread.
  NB : souvent, on peut aussi retrouver l'utilisateur de l'objet connecté au niveau 1. Par exemple, si votre objet connecté concerne la gestion de trams. Les capteurs dans le tram ainsi que l'utilisateur final du tram sont au niveau 1. La société à laquelle vous vendez votre solution (TEC, STIB,…) est au niveau 2. Dans ce cas, l'utilisateur final au niveau 1 peut consulter des données du tram (localisation, taux d'occupation,…) qui est aussi au niveau 1.
  NB : Le Raspberry Pi sera au niveau 1 ou au niveau 2.
- Niveau 2 : l'utilisateur des objets connectés qui va pouvoir interagir avec les objets connectés pour fixer des seuils, pour recevoir des informations,… Les interactions s'effectueront par exemple via une VM Ubuntu. Le niveau 2 est connecté au niveau 1 par exemple en WiFi ou en Lora. On peut aussi imaginer que les données des capteurs partent dans le Cloud et que l'utilisateur se connecte via le Web pour interagir avec les objets connectés
- Niveau 3 : vous et votre entreprise. Vous n'avez pas qu'un seul client. Vous avez vendu votre solution à différents clients indépendants les uns des autres, mais qui ont tous été séduits par votre solution. Vous restez connecté avec les objets qui sont chez vos clients pour réaliser la télémaintenance, pour connaître l'utilisation qui est faite de vos objets, pour réaliser des mises à jour, pour pouvoir faire du reporting chaque semaine, chaque mois… C'est à ce niveau que se situe notamment une base de données. Le niveau 3 est connecté au niveau 2 à travers Internet.
- Niveau 4 : le Cloud : vous utilisez des services Cloud pour obtenir ou stocker des informations. Par exemple, vous utilisez des APIs dans le Cloud, un broker public MQTT,...

Par exemple, si votre objet connecté concerne les soins de santé, on pourrait retrouver au niveau 1 votre objet connecté qui est porté par le patient. Au niveau 2, on retrouve l'utilisateur, par exemple le médecin ou l'infirmière qui gère à son niveau différents objets et qui recoît les données pour par exemple ajuster le traitement des patients. Au niveau 3, on vous retrouve, vous et votre entreprise. Vous restez en contact pour offrir du service et du support à vos clients. Au niveau 4, on retrouve les

services Cloud que vous utilisez pour par exemple stocker les données et que vous accédez via des API REST.

2. ARCHITECTURE DU PROJET

Pour présenter votre solution, vous avez rendu une première version du schéma de l'architecture de votre projet (1 page A3 ou A4 – format paysage - pas manuscrit – schéma en couleurs – attention à la taille des caractères une fois le schéma imprimé - utilisez au mieux l'espace disponible – pas de zones vides – couleurs de fond claires différentes pour les 4 cadres représentants les différents niveaux). Vous avez utilisé par exemple https://app.diagrams.net pour réaliser ce schéma.

Ce schéma d'architecture doit comprendre tous les composants de votre projet et vous y indiquerez les interactions **entre tous ces composants** (lignes ou flèches) et sur ces lignes/flèches, indiquez les modes de transmission (au moins 2 logos par ligne ou flèche : par expl Lora et MQTT, WiFi et HTTPS, WiFi et VNC, WiFi et email, SMS et 4G, …). Utilisez des couleurs différentes pour ces lignes ou flèches et choisissez judicieusement l'épaisseur de ces lignes ou flèches. Ces lignes/flèches seront **unidirectionnelles**. En observant votre schéma, on doit pouvoir comprendre à quoi sert l'interaction. Au besoin, utilisez des petits logos supplémentaires (par expl : triangle avec point d'exclamation pour indiquer qu'il s'agit d'une alerte ou un logo battery low pour prévenir qu'il convient de recharger). Indiquez également sur les lignes ou flèches si la connexion est sécurisée en ajoutant par expl le logo d'un petit cadenas.

Quand vous envoyez un mail dans le cloud, il faut que quelqu'un reçoive l'info. Cela n'a pas de sens si personne ne vient lire ce mail… On doit pouvoir facilement visualiser le producteur (la source) d'un message et le consommateur correspondant (la destination). (idem pour SMS, Webex, MQTT, REST, …)

Quelques contraintes/remarques suite à votre avant-projet :

- Il n'est pas judicieux de mettre des photos de RPi, de capteurs,… Par contre, vous utiliserez des icones pour les différents composants (WiFi, LoRa, MQTT, Node-Red, Python, Webex, RPi, Mosquitto, base de données, Arduino IDE, Twilio ou équivalent, Ubuntu, Shodan, ESP32, VNC, caméra, capteurs,…). L'utilisateur de l'objet connecté sera également représenté par une icone. "Une image vaut mille mots". Utilisez également des icones/images pour les particularités de votre projet : moto, plante, enfant, maison, personnel soignant, animal, … En regardant votre schéma, on doit comprendre à quoi sert votre objet connecté.
- si le logo est bien choisi, il n'est pas souhaitable d'ajouter du texte (par expl : raspberry pi, caméra,…) Par contre, si le logo n'est pas suffisamment explicite, ajouter un texte bien choisi pour signaler de quoi il s'agit.
- Il s'agit d'un seul schéma et pas de plusieurs petits schémas. identifiez clairement les 4 niveaux. Il doit être possible de comprendre clairement où se situe chaque niveau. Par exemple, vous indiquerez : niveau 1 : poignet du patient, niveau 2 : bureau des infirmières,...
- A côté des cartes ESP32-WiFi, ESP32-LoRa, ESP32-H2-Thread et du RPi, indiquez quels capteurs/actuateurs sont connectés. Regroupez les entrées (capteurs) et les sorties (actuateurs). Rappel : un logo est meilleur qu'un mot.
- Les VMs Ubuntu 22.04 utilisées seront également représentées avec leur rôle (MQTT Broker, DB, …) ainsi que le smartphone utilisé pour visualiser les dashboards, ...
- Les langages/outils utilisés apparaîtront à côté des composants concernés: Python, Micropython, Node-Red, Arduino, C/C++,...
- On peut évidemment avoir plusieurs ESP32 ou plusieurs RPi par client. Pour la démo, vous n'en avez qu'un seul de chaque type, mais votre architecture peut en montrer plusieurs…
- Pour bien montrer qu'il existe plusieurs objets au niveau 1 contrôlés par un niveau 2 et plusieurs clients gérés par votre société (niveau 3), vous écrirez, à côté de niveau 2 : "gestion de x …." (par exemple, gestion de dizaines de caddies connectés). De même, à côté de niveau 3 (votre entreprise), vous écrirez : "gestion de x..." (par exemple, gestion de centaines de magasins). Remplacez "x" par un nombre réaliste !
- Note sur l'utilisation des logos : ne pas changer la couleur "officielle" d'un logo.

- L'impression de ce schéma doit être en couleurs et de qualité, sans numéro de page et entièrement en format paysage. Le titre de votre projet se situera au dessus de votre schéma.
- Si votre schéma est difficilement lisible notamment à cause d'un grand nombre de lignes/flèches, vous devrez changer l'agencement des composants reliés par ces lignes/flèches.
- Ne pas utiliser de légende en dessous de votre schéma d'architecture : un logo WiFi sur une flèche est nettement mieux qu'une ligne pointillée qui renvoie à une légende.
- Pour les **flèches qui montrent l'utilisation de MQTT**, faites apparaître le topic utilisé et le niveau de QoS sur les flèches. Le sens de la flèche montrera s'il s'agit d'un publish flèche du publisher vers le broker) ou d'un subscribe (flèche du broker vers le subscriber)

ATTENTION : Vous ne pouvez pas dessiner sur votre schéma des fonctionnalités que vous n'avez pas implémentées ! Si vous dessinez un capteur ou toute autre fonctionnalité, il vous sera demandé une démonstration concluante de ce capteur ou de cette fonctionnalité et si ce n'est pas implémenté, vous serez sanctionné...

3. CONTRAINTES TECHNIQUES MINIMALES

Ci-dessous, vous trouverez les contraintes techniques **MINIMALES**. Vous **devez** intégrer des composants additionnels. L'absence d'ajout de composants additionnels sera sanctionné. Soyez innovants !

Au minimum, vous devrez utiliser :

- Le raspberry pi avec carte GrovePi et adaptateur LA66-LoRaWan
- 2 machines virtuelles Ubuntu 22.04 avec utilisation d'au moins une application dans un container docker
- Un broker MQTT Mosquitto avec utilisation de certificats & port 8884
- Votre smartphone pour vous connecter à une interface Web (dashboards Node-Red) et également à une room web.webex.com. Concernant Webex par exemple, il n'est pas acceptable d'avoir par exemple une seule room partagée entre tous les clients !
- ESP32-WiFi qui communiquera en WiFi et en MQTT (utilisation de Micropython)
- Une communication en LoRa entre ESP32-LoRa et adaptateur LA66-LoRaWan
- Une communication Thread entre les 2 ESP32-H2-Thread et envoi en WiFi vers le raspberry pi.
- L'afficheur LCD connecté au GrovePi

La connexion au réseau WiFi sera sécurisée en WPA2-PSK (connexion via 4G ou WiFi IOT).
Note importante : une connexion 4G n'est nullement imposée. Les étudiants peuvent utiliser le WiFi IOT du laboratoire. En outre, une connexion 4G défaillante n'est nullement une excuse acceptable lors de votre démonstation.

L'accès à l'interface graphique du RPi s'effectuera en VNC à partir d'une VM Ubuntu 22.04.

Votre projet montrera que vous maîtrisez Node-Red, Python (utilisation de la librairie **Paho** et utilisation **judicieuse** de la librairie **multithreads "threading"** , micropython, **MongoDB** (utilisation du noeud MongoDB4) et les API REST.
En ce qui concerne les API REST, vous devez au moins ajouter un service web utile à votre projet et **non vu au cours**. L'accès s'effectuera avec la librairie Python "requests". Vous intégrerez également au moins une requête à l'API shodan. Attention : Il s'agit de prévoir une utilisation UTILE pour votre projet. Pas question, que ce soit pour Shodan, Webex ou autre de montrer l'utilisation de ces techniques dans un contexte extérieur à votre projet.
En outre, vous écrirez **votre propre API REST avec authentification et utilisation de OData** (utilisation du micro framework Flask)

Votre projet utilisera au moins 4 parmi les capteurs suivants : distance, luminosité, capteur angulaire, capteur de son, de température/humidité, bouton, GPS.

Votre projet utilisera au moins 2 sorties parmi les suivantes : Buzzer, LED, Relais.
Ces capteurs et sorties doivent être UTILES pour votre projet. Par exemple, on n'intègre pas un capteur qui ne sert à rien.

Les ESP32 feront au minimum l'acquisition d'une mesure via un capteur et transmettront les résultats via le réseau WiFi, LoRa ou Thread. Vous utiliserez un réseau privé (Peer-to-Peer) LoRa pour communiquer entre la carte ESP32-LoRa et l'adaptateur LA66-LoRaWan connecté au RPi. La carte ESP32-LoRa sera émetteur et l'adaptateur LA66-LoRaWan sera récepteur. Des messages pertinents seront affichés sur l'écran Oled. Pour Thread, envoi de données entre les 2 ESP32-H2-Thread et envoi en WiFi vers le raspberry pi.

Des notifications signaleront des événements par mail et également par SMS. Attention, il n'est pas pensable d'envoyer en boucle des SMS ou des mails. Ce que vous faites doit avoir du sens !

Du logging sera mis en place et sauvegardé dans la base de données sur une VM Ubuntu 22.04 (attention : le timestamp doit être présent). Il faudra pouvoir afficher un **historique des mesures (dates à encoder dans formulaire** via dashboard Node-Red ou interface web). Pensez à l'ergonomie !

Votre projet utilisera impérativement le RFID (utilisé en général pour donner un accès – peut aussi se trouver sur un animal), la puce NFC, l'afficheur LCD ainsi que la caméra Picam.

En ce qui concerne MQTT, vous devrez au moins une fois sécuriser les échanges à l'aide de certificats côté client et côté serveur. Vous devrez utiliser au moins une fois une QoS de 2 dans des échanges non-sécurisés par TLS. Utilisez **Wireshark** pour prouver que la QoS de 2 a été correctement implémentée. Vous utiliserez la librairie Python Paho.

En ce qui concerne Node-Red, vous devrez utiliser les dashboards et au moins une fois afficher des données sur une carte (**noeud Worldmap**). En ce qui concerne cette fonctionnalité, l'utilisation doit être, comme les autres fonctionnalités de votre projet judicieuse et utile. Par exemple, il peut être judicieux de montrer la position des trams si vous êtes sur le quai. Utiliser un formulaire pour récupérer des données de l'utilisateur. Les dashboards, c'est pour visualiser, mais également pour interagir avec l'environnement (**fixer des seuils**,…) via un ou des formulaires Node-Red.

Sur votre RPi, il ne s'agit pas de prendre la main au démarrage du RPi, et de lancer indépendamment chaque petit programme qui gère les capteurs. Votre application est démarrée automatiquement au démarrage du RPi (utilisation d'un script systemctl). De même, les cartes ESP32 démarreront automatiquement. Par exemple, il n'est pas acceptable de se rendre dans le code Node-Red et de démarrer une fonctionnalité manuellement ou de lancer un script python à partir de la ligne de commande ou de démarrer un container manuellement...

4. A RENDRE LE JOUR DE L'EXAMEN EN JANVIER.

- Le matériel prêté (tout le matériel prêté devra être rentré le jour de l'examen).
- Un dossier de quelques pages **agrafées** reprenant (présentation et orthographe soignées svp):
  - Une page de garde reprenant au minimum : Nom+Prénom+Date+Nom de votre projet
  - Descriptif de votre projet : Description du problème - Description de votre solution – Pourquoi nous avons choisi ce projet (1 page A4)
  - Ce document-ci imprimé sur lequel vous aurez surligné en fluo toutes les contraintes que vous avez respectées. Rappel, comme ce sont les contraintes minimales, il doit y avoir bcp de fluo! Attention : du fluo sur une ligne qui n'a pas été implémentée est considéré comme une tentative de fraude et sera lourdement sanctionné. Toutes les contraintes minimales doivent être implémentées. Il est donc inacceptable d'arriver à l'examen avec ce document non surligné.

- Le schéma en couleurs (1 page A4 ou A3) de l'architecture de votre solution reprenant les composants hardware et software utilisés. Attention, vous avez reçu un feedback concernant votre dossier. Il est essentiel de tenir compte des remarques qui ont été faites!
- Un screenshot de vos dashboards.
- Les diagrammes utilisant le modèle C4 vu au cours :
  - Le niveau 3 (composant) du modèle C4 pour les ESP32 + Raspberry Pi
  - Le niveau 4 (code) du modèle C4 : Les diagrammes de séquence et d'état des éléments pertinents du niveau 3. (Par exemple : appui sur un bouton par l'utilisateur, diagramme d'état de l'afficheur LCD, des LEDs….)
- Aspect sécurité : Relevé des actions prises pour sécuriser votre solution + utilisez https://sectigostore.com/blog/owasp-iot-top-10-iot-vulnerabilities/ et décrivez (Min 1 page et Max 2 pages) comment vous adressez le Top10 des IoT vulnerabilities. Soyez précis : il ne suffit pas par exemple de dire : "j'utilise des mots de passe complexes". Il convient d'expliquer comment vous pensez gérer les mots de passe (rappel : vous avez de nombreux clients).
- Une conclusion : quel était votre niveau de compétence en IoT avant de démarrer ce projet ? quel est votre niveau de compétence à la fin du projet ? Quelles ont été les difficultés rencontrées lors de ce projet ?

- Une archive (.rar ou .zip ou .7z) reprenant votre contribution personnelle : 6 répertoires : un répertoire RPi avec le code que vous avez écrit sur le RPi (node-red + python + …), un répertoire ESP32-WiFi avec le code que vous avez utilisé sur votre ESP32, un répertoire ESP32-LoRa avec le code que vous avez utilisé sur votre ESP32-LoRa, un répertoire ESP32-H2-Thread, un répertoire Ubuntu avec le code qui tourne sur vos VMs Ubuntu et un répertoire Dossier avec votre dossier au format pdf et également un fichier pdf séparé avec votre schéma d'architecture. Cette archive doit être prête avant l'examen sur votre ordinateur. (Il n'est donc pas acceptable de se présenter à l'examen sans cette archive prête).

Attention : la démonstration que vous ferez le jour de l'examen doit être **fonctionnelle** et comprendre **au strict minimum toutes** les contraintes techniques minimales décrites ci-dessus. Vous disposerez d'un peu de temps pour préparer votre démo (max 1h).
Pensez également à un packaging (boîtes en carton, en plastique,….) : c'est un prototype, mais il doit être présenté de manière correcte et adaptée à votre projet (il n'est pas acceptable d'avoir des capteurs reliés au Rpi, aux ESP32 sans aucun packaging). Par expl, si c'est un bracelet connecté, imaginez une manière d'attacher votre objet au poignet.
Les critères suivants seront également utilisés pour l'évaluation : originalité/ingéniosité du projet, pertinence du projet (would I buy it ?), utilité de l'objet, fiabilité de la solution, technologie(s) supplémentaire(s), niveau de difficulté + votre dossier.
L'ordre de passage vous sera envoyé par l'école virtuelle.

Remarque importante: ce n'est pas parce que votre projet est fonctionnel à **100%** et respecte toutes les contraintes minimales de ce document et que votre dossier technique est impeccable que l'examen sera réussi. Un projet fonctionnel à **100%** avec toutes les démos concluantes et un respect de toutes les consignes est nécessaire, mais ce n'est que **le point de départ de l'évaluation**. L'étudiant doit maîtriser toutes les facettes de son projet, dont le code et doit être capable d'apporter rapidement de petites modifications dans son code. Le code sera de qualité, digne d'un étudiant master en informatique. Un étudiant qui ne connaît pas son code ou qui ne sait pas y apporter de petites modifications ne réussit pas l'examen. Il n'y a pas une cote pour le projet et une cote pour la défense orale, mais une seule cote globale pour l'examen de laboratoire.

5. CONSIGNES POUR LA PARTIE EVALUATION ECRITE DU COURS

Le même jour que la présentation de votre projet, vous recevrez un questionnaire écrit portant sur la théorie (matière vue au cours) et sur votre projet.