CÉGEP MARIE-VICTORIN

420-410
Introduction aux plateformes IdO

Épreuve finale

Remise : 30 mai 2025

Prénom & Nom : _____

Matricule : _____

Signature : _____

/ 40

- ***Il est interdit d'aller chercher de l'information ailleurs que dans le matériel autorisé.***
- ***Le travail doit être fait durant les séances de cours, ou sous la supervision du professeur.***

Remise :

- ***Détails dans ce document.***

Matériel autorisé :

- ***Toute documentation externe est permise à l'exception des outils IA de génération de code (ChatGPT, Gemini, Perplexity etc.)***

Description générale

Vous devez faire une application IoT utilisant un *RaspberryPi,* un senseur DHT11, un bouton et 3 LED ordinaires. Cette application utilise le protocole MQTT et peut être contrôlée avec une API REST simple.

Spécifications

À chaque 30 secondes, l'application doit envoyer par MQTT les valeurs de température (en Celsius) et d'humidité (en pourcentage) lues sur le module DHT11. Les données doivent aussi être envoyées lorsqu'on appuie rapidement sur le bouton.

Le *topic* MQTT utilisé pour envoyer les données de température est **`final/HOTE/T`** et celui pour les données d'humidité est **`final/HOTE/H`**. Vous devez remplacer HOTE par le « hostname » de votre RaspberryPi.

Si votre Pi est celui où la température la plus élevée est lue, la LED rouge doit s'allumer.

Si votre Pi est celui où l'humidité la plus élevée est lue, la LED bleue doit s'allumer.

Lorsqu'on appuie plus de deux secondes sur le bouton, l'envoi de données est activé / désactivé. Une LED blanche est allumée lorsque l'envoi de données est activé.

Une API faite sur *Flask* donne accès à deux points terminaux :

URL /etat
TYPE POST
Attributs :

- **etat (int)** : 0 désactive l'envoi de données, 1 active l'envoi de données.

URL /donnees
TYPE GET
Attributs :

- **T (int)** : Nombre entier correspondant à la dernière température lue
- **H (int)** : Nombre entier de 0 à 100 correspondant à la dernière valeur d'humidité lue

Consignes

La communication avec le GPIO du RespberryPi doit se faire avec le module ***pigpio***, et le programme doit s'exécuter dans un environnement virtuel.

Vous devez faire un projet sur *GitHub* pour le code de votre application. Le projet doit contenir le fichier ***requirements.txt*** de votre environnement virtuel.

À la fin de chaque séance de travail, vous devez « pousser » sur *GitHub* le travail réalisé durant la session.

Chaque coéquipier doit travailler sur ses propres sections du programme, et vous devez identifier clairement dans le code, à l'aide de commentaires, les auteurs de chacune des sections de votre programme.

Vous devez citer vos sources en commentaires si vous utiliser du code basé sur des exemples provenant de sources externes.

Grille de correction

Fonctionnalité : le programme fait ce qui est demandé et le fait de manière efficace / rapide	/10
Robustesse : le programme fonctionne dans des conditions réalistes sans erreurs d'exécution	/10
Qualité du code : le code du programme est bien structuré (fonctions, structures de contrôle) et ne contient pas d'éléments inutiles ou inefficaces.	/10
Commentaires : Les commentaires sont clairs et expliquent, pour les différentes sections du code, à quoi elles servent et leur fonctionnement général	/10
TOTAL	**/10**