Lundi, 12 décembre 2022

**Internet des objets - Examen final, sur 30**

**Répartition des points de la session**

N'ayant pas pu faire l'examen 1 à la semaine 10 à cause de problèmes techniques, vous acceptez que les 20 points réservés pour cet examen soient appliqués au travail de fin de session que vous avez complété le 5 décembre et à l'examen final. Donc le travail de fin de session vaut 50% de la note finale et l'examen final 30%. Les 20% restant proviennent du premier travail remis (révision du code de la maison). L'examen final a une durée de trois périodes.

**Examen:**

**Question 1: 20 points**
Votre système devrait déjà pouvoir recevoir la commande d'allumer et éteindre la LED externe et le relais. Ce travail était la principale fonction à développer pour le travail de fin de session. Vous devez maintenant programmer 2 fonctions de commandes supplémentaires

**Partie a- 10 points** Votre système doit pouvoir accepter une commande "fan999" où 999 représente la vitesse de rotation du ventilateur. Le code existe déjà dans le programme du Arduino de la maison (commande w999#). La commande aura donc le format

Topic: commande/identifiant
Payload: fan999 (variant de fan0 à fan255)

Vous devez retourner une réponse du format

Topic: reponse/identifiant
Payload: "success" ou "succès" + la commande reçue

**Partie b- 7 points** Votre système doit pouvoir accepter un commande "x…x" où x…x représente une chaine de caractères de 1 à 16 caractères. Cette commande commence TOUJOURS par un "x", et le reste est une chaine de caractères (1 à 16 de long).Vous devez afficher ces caractères sur la première ligne de l'écran LCD de la maison. Le code sera similaire au code pour la fan (w) mais la fonction x n'existe pas. La commande aura le format

Topic: commande/identifiant
Payload (Exemple): x1234567890abcde

Vous devez retourner une réponse du format

Topic: reponse/identifiant
Payload: "success" ou "succès" + la commande reçue

**Partie c- 3 points**Si la commande reçue est inconnue, vous devez retourner

Topic: reponse/identifiant
Payload: "error" ou "erreur" + la commande reçue

Rappelez-vous que les commandes valides sont "a", "b", "c", "d", "fan", « x",
« AllumerLEDExterne », « EteindreLEDExterne », « AllumerRelais », « EteindreRelais »

**Question 2: 10 points**
Votre système devra pouvoir accepter des demandes d'état.

Votre système doit pouvoir accepter une demande d'état "lumiere" ou "gaz". Le code existe déjà dans le programme du Arduino de la maison (commandes "h" et "i") qui retournent 1 à 3 chiffre représentant un nombre de 0 à 1023. La commande aura le format

Topic: etat/identifiant
Payload: "gaz" ou "lumiere" ou "i" ou "h"

Vous devez retourner une réponse du format

Topic: retour/identifiant
Payload: nombre de 0 à 1023 **sous forme de string**

**Notes importantes:**

**—Vous devez remettre l'équipement ET un fichier zip dans LÉA qui contient votre code Arduino de la maison et du Mega.**

**—Les commandes et demandes d'état arrivent en continu à peu près à toute les 5 ou 10 secondes.**

—Vous pouvez utiliser le code que j'ai placé dans mon Github au besoin. Le code "maison 2 arduino" fonctionne pour les commandes et demandes d'état (il faut bien sûr le modifier un peu…)

—J'utilise une petite fonction qui place mon identifiant dans une variable de contexte à chaque "deploy" avec un "inject"

```
global.set("identifiant", "maison_9"); //maison_1 à maison_23
return null;
```

Je récupère mon identifiant au besoin avec

```
global.get("identifiant");
```

—Il pourrait être utile de comprendre ceci:

```
            msg.payload = msg.payload.substr(3,5); //à partir de 3 (n'oubliez pas qu'on compte le 0) lire
5 charactères
            msg.payload = 'w' + msg.payload + '#';
```

—Si vous avez de la difficulté à traiter l'information numérique qui revient de la maison, vous pourriez utiliser ce bout de code dans une fonction Node Red. Comme les nombres arrivent un caractère à la fois, il faut être certain qu'on a tout reçu pour continuer. Pour que ça fonctionne bien, j'utilise une fonction "trigger" qui envoie "Fin!" pour compléter la séquence de décodage.

```
input = msg.payload;
input = String(input);                          //juste au cas!
storage = context.get("storage")||[];           //rappel des derniers input. Si vide, on initialise un
array vide

if (input == "Fin!") {                          //Le signal de fin, pour procéder au .join()
    output = storage.join()                     //mettre toutes les parties ensemble
    output = output.replace(/,/gi,"");          //enlever les virgules
    msg = {payload:output};                     //créer l'output
    node.send(msg);                             //on fait node.send au lieu de return
    context.set("storage",[]);                  //on "vide" storage
}else if (input != "Fin!") {                   //Si il y a d'autre input qui viennent
    input = input.trim();                       //enlever les charactères 10 et 13 (aussi " " au besoin)
de l'input
    input = input - 48;                         //transformer en chiffre (en soustrayant 0)
    storage.push(input);                        //rajouter à la fin du storage
    context.set("storage",storage);             //remettre le storage dans le "context"
}
return null;
```