

Réseau loT Chenillard KNX

Théo Delagarde - Corentin Grosos 24 mai 2022





Sommaire

1. Introduction

2. Interface Web & chenillard

3. Architecture & communications

4. Synchronisation

5. Conclusion

Introduction

Objectif: Faire communiquer une maquette knx et une interface web par le biais d'une API Rest.

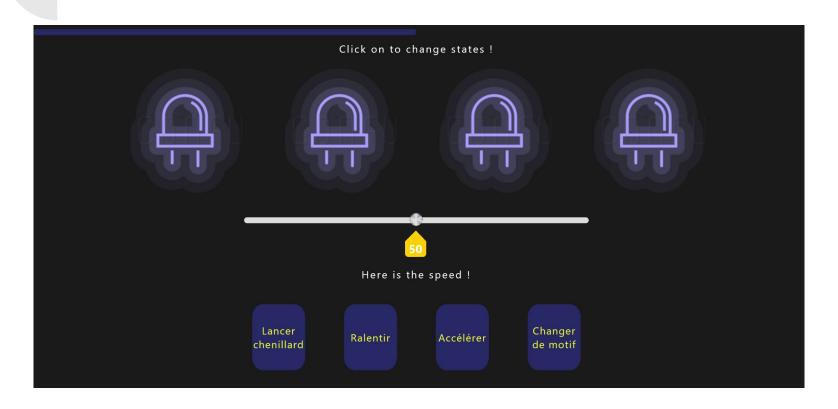
Langages de programmation choisis :

Client: HTML / CSS / JS Server: Javascript

Organisation du projet :

Interface Web/API
Connexion KNX
Communication client / serveur / KNX

Interface Web



Interface Web

Lancer chenillard

Gestion du chenillard

Actif quand chenille_On == true Appel de la fonction récursive chenilleMOTIFS()

Ralentir

Gestion de la vitesse définie par

ActualSpeed = minSpeed + IntervalSpeed + IntSpeed

Accélérer

Avec minSpeed = 300, Interval Speed = 80 et IntSpeed défini entre 0 et 10 selon les boutons Ralentir et Accélérer ⇒ 300 ms < ActualSpeed < 1100 ms

(chenille_On == true)

traitement si

Changer de motif

Gestion du motif parmi 6 implémentés



Fonctionnement du chenillard :

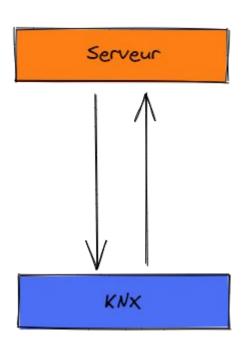
Fonction récursive appelée toutes les 300 - 1100 millisecondes

- Chenillard Simple
- Chenillard inversé
- Chenillard B2B
- Allumage aléatoire
- Allumage en quinconce
- Clignotement



```
function chenilleMOTIFS(){
  if(chenille On == true){
     sleep(actualSpeed).then(() => {
       switch(decideMotif()){
          case "chenillardSimple":
             /* traitement */
             /* modification des leds */
             /* calcul de l'indice pour le prochain appel */
             return chenilleMOTIFS();
          case "chenillardInverse":
             /* traitement */
             /* modification des leds */
             /* calcul de l'indice pour le prochain appel */
             return chenilleMOTIFS();
          case "chenillardBackToBack":
          case "allumageAleatoire":
          case "parityQuinquonce":
          case "everyLEDsOn":
```

Communication Serveur - KNX



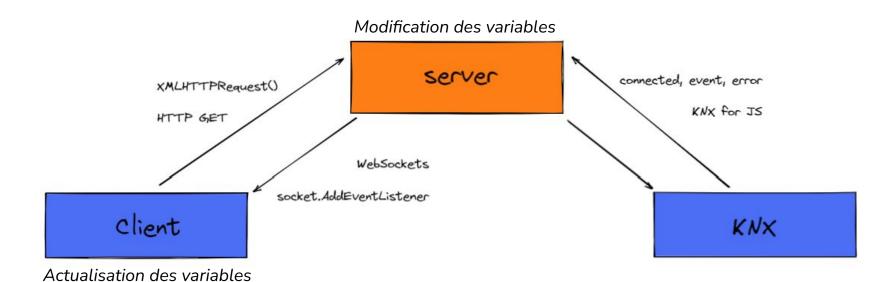
Librairie KNX pour JS

- → Renseignement de l'adresse IP de la maquette
- → Handlers contenant 3 évènements :
 - connected
 - event
 - error

Librairie de déconnexion

async-exit-hook

Architecture



Synchronisation client / serveur

server

Lancement du programme :

Initialisation des variables chenille_on, actualspeed, numMotifs, etc.

Appui sur les boutons de la maquette :

Modification des variables Envoi des variables modifiées à tous clients en Websocket

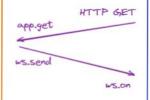
Appui sur les boutons de la page web :

Modification des variables Envoi des variables modifiées à tous clients en Websonket

Appel de chenilleMOTIFS() connexion, write("0/0/"+ledIndice, 1)

Communication Synchronisation

ws.send ws.on



Client

Initialisation des variables chemille. On, actualspeed, numMotifs, etc.

Récupération de la valeur des variables côté serveur

Actualisation des variables

Demande de modification des variables

Actualisation des variables

Appel de chenilleMOTIFs()
tabLeds[ledIndice].src = "images/led-orange"

Conclusion

Développement API Rest

Pas d'utilisation de framework

Implémentation simple du KNX

Améliorations possibles

Démonstration