Institut National des Sciences
Appliquées de Lyon
&
KompleXKapharnaüM

Stage de 4ème année du département Génie Électrique

Projet Do Not Clean

Réalisation d'une carte multimédia programmable et contrôlable via wifi

Proposition pour le système de scénarios

Auteur :
Olivier Radisson

Tuteur de stage :
Gilles Gallet

Chef de projet :
Pierre Hoezelle

- 31 septembre 2014 -

Dernière édition le 3 octobre 2014

Résumé

Ce document présente une proposition pour le système de scénario. Chaque carte, en plus de pouvoir être contrôlable via des ordres reçus en wifi, se doit de pouvoir conserver un fonctionnement autonome. C'est ce fonctionnement là qui est décrit par les scénarios. La proposition suivante découpe chaque scénario en plusieurs parties :

— Des scènes, qui sont jointes entre elles via des transitions.
— Des suites d'étapes, qui sont ordonnées et jouées à l'intérieur des scènes.
— Des étapes, éléments les plus élémentaires permettant de déclencher des actionneurs via la carte.
— Des interruptions, qui permettent de déclencher certain comportement quelque soit l'étape en cours.
— Des transitions, qui sont franchies lorsque leurs pré-requis sont accomplis, permettant de gérer le flux du scénario.

Table des matières

1 Introduction

Cette proposition est là pour essayer de mettre une première fois au clair les besoins et les solutions qui concernent le système de scénario de la carte. C'est donc une proposition qui est vouée à évoluer, voir être totalement remodelée.

L'attention est portée uniquement sur le système de scénario, pas sur les capacités directes de la carte ni sur l'interface finale de la régie. Pour finir ce document à vocation d'être lu autant par des technicien(ne)s que par les artistes, et un maximum de retours est souhaité.

2 Le système de scénario

2.1 Motivations

Le système de scénario trouve sa raison d'être par le fait que la carte doit pouvoir être autonome et agir sans forcément attendre des ordres de la régies. Pour pouvoir se faire, il est donc nécessaire de prévoir *en amont* de la représentation une *marche à suivre* pour que la carte sache quoi faire en l'absence d'ordres directes, mais aussi savoir comment perturber se fonctionnement autonomes lorsque des ordres lui sont envoyés.

2.2 Besoins identifiés

De la réunion du mercredi 17 septembre sur le projet de DNC[1], des divers échanges avec Gilles, Katia et Pierre ainsi que des quelques discutions avec certaines des personnes qui sont vouées à utiliser cette carte, j'ai commencé à identifier quelques besoins élémentaires qui doivent être satisfaits par le système de scénario.

Éditable par des non-techniciens : Premier besoin, les scénarios doivent pouvoir être éditables assez facilement sans faire appel à l'un des créateurs du système. En revanche si le système de scénario nécessite une petite documentation et un introduction rapide avant usage, cela semble ne pas être bloquant.

Éditable facilement via une régie en amont du spectacle : Le système d'édition des scénarios doit pouvoir être accessible facilement de-

1. Do Not Clean

puis une régie. Pas besoin de matériel spécifique (câble de liaison FTDI ou JTAG par exemple), une simple connexion WIFI et un terminal [2] avec l'interface de gestion du système de scénario installée doit suffire.

Logique simple pour répondre rapidement à de petits cahiers des charges : Le système doit pouvoir créer des scénarios simples rapidement. Il ne faut pas que celui-ci demande beaucoup d'investissement avant de pouvoir faire fonctionner un scénario élémentaire.

Logique assez souple pour pouvoir répondre à des fonctionnements plus complexes : Le système doit en revanche pouvoir être assez souple pour permettre, malgré sa simplicité, de prévoir des cas de figure plus complexes qu'une simple liste d'actions avec conditions.

Fonctionnement rapide pour permettre une prise de contrôle sans latence via les ordres WIFI : Le système se doit d'être assez rapide pour permettre à des ordres WIFI d'êtres interprétés rapidement et ne pas afficher de latence particulière, notamment sur certaines taches précises.

2.3 Contraintes techniques identifiées

Cette partie peut-être sautée par ce(lles)(ux) qui ne sont pas intéressé(e)s par les aspects purement technique du système

Le fonctionnement du système doit faire face à plusieurs contraintes liées à l'organisation du projet, aux contraintes matérielles de la carte, aux spécificités du réseau, etc.

Contrainte de vitesse d'exécution : La carte possède deux unités programmables [3] : un Atmega644 et un processeur Dragino. Le premier est cadencé à 1.6 MHz le second à 400 MHz. En revanche, bien que le processeur Dragino soit plus de 200 fois plus rapide, il fait tourner un système d'exploitation et une interface WIFI qui réduiront tout deux forcément le temps processeur disponible pour notre programme. Toutefois, Linux, le kernel qui tourne sur le processeur Dragino, est particulièrement efficace et permet un ordonnancement précis des taches permettant de prioriser

2. Terminal incluant tout dispositif informatique récent avec un écran pouvant faire tourner le logiciel de régie

3. Hormis les cartes radios qui n'en possède qu'une car elle n'ont pas le shield wifi

à notre souhait une action particulière. Je pense donc que le système de scénario sera tout de même plus rapide sur le processeur Dragino.

Contrainte liée au langage de programmation Le processeur Dragino permet d'utiliser n'importe quel langage de programmation, notablement le langage python. L'Atmega lui n'accepte que du code compilé, donc principalement en C/C++ [4] qui, pour un système de scénario qui se veut souple ce n'est pas vraiment les langages de prédilection.

Contrainte liée à la mémoire disponible : Les scénarios pouvant être longs et complexes prendront forcément une place qui peut devenir importante, il est donc nécessaire de tenir compte de ce facteur là. Le processeur Dragino vient avec 16 Mo de mémoire Flash et 64 Mo de mémoire vive (RAM), l'Atmgea possède une mémoire programme de 64 ko et une 4 Ko de mémoire vive (SRAM).

Contrainte liée au réseau WIFI : Le réseau WIFI induit des ralentissements dans l'échange d'informations entre les différents membres du réseau, mais peut aussi impliquer : des pertes de messages ou pire, la perte totale du signal. Il faudra donc forcément garder ces contraintes à l'esprit.

4. Ce n'est pas tout à fait vrai, il est possible de faire tourner un interpréteur python sur l'Atmega mais c'est une manipulation extrême et peu souhaitable.

3 Proposition

Pour répondre à ces besoins en tenant compte des contraintes évoquées, voici une première proposition d'un système de scénario.

3.1 Idée globale

Dans l'ensemble le système de scénario découperait l'univers [5] en plusieurs entités :

- Les cartes DNC WIFI
  - Les cartes DNC WIFI seulement
  - Les cartes DNC Gateway [6] WIFI-Radio
- Les cartes DNC Radio
- Les régies DNC

et en plusieurs phases :

- La phase de création
- La phase de représentation

Chaque élément ayant un comportement donné pour une phase donnée.

Durant la phase de création une régie peut se connecter à une carte WIFI pour lui injecter un scénario créé via l'interface proposée. Durant cette phase là, les cartes WIFI ne font qu'attendre de recevoir un scénario envoyé par une régie. Les cartes Radio n'ont pas de fonction pendant la phase de création.
Durant la phase de représentation les cartes WIFI exécutent leur scénario pré-établi et réagissent en fonction des entrées [7] pouvant également envoyer elles-mêmes des ordres selon leur scénario, la régie elle n'a plus que vocation à être un élément capable d'envoyer des ordres aux cartes WIFI, elle peut également leur demander de présenter leur état et permet une visualisation des paramètres de chaque membre du réseau. Les cartes Radio sont elles contrôlées directement via une carte WIFI ayant en plus un module Radio. Elles sont donc dépendantes de cette carte, mais cela permet la multiplication à faible cout de petites installations [8].

5. Façon de voir les choses. L'univers qui représente l'ensemble des éléments qui ont un lien avec le projet
6. Porte d'entrée
7. Ordres WIFI, capteurs, signal radio, timers, etc
8. Le module WIFI étant la partie la plus onéreuse de la carte

3.2 Organisation des scénarios

Chaque scénario pourra être créé par une interface disponible sur les régies. Cette interface est encore à définir, mais pour l'instant elle serait une combinaison d'une partie graphique permettant d'organiser la structure du scénario, et d'une partie textuelle permettant de décrire le comportement de chaque bloc de la structure.
Les scénarios pourront être enregistrés et sauvegardés côté régie dans des fichiers, permettant de garder des archives ou de faire différents essais sans pour autant supprimer les tests déjà faits.
Chaque scénario sera découpés en scènes distinctes reliées entre elles par des transitions.

Le scénario ne pourra se trouver que dans une scène à la fois, il ne sera pas possible de faire du parallélisme via les scènes. Ceci pour ne pas complexifier le système et pour éviter les innombrables problèmes liés au parallélisme et enfin pour laisser le système utilisable sans une trop grande préparation et une documentation excessivement lourde.
En revanche le scénario ne sera pas totalement bloquant et des évènements extérieurs pourront déclencher certaines actions quelque soit l'état en cours, ce seront les interruptions.
Enfin chaque scène sera composée d'une liste ordonnée d'étapes reliées elles aussi par des transitions. Ces étapes seront composées d'actions élémentaires permettant d'utiliser les capacités de la carte : allumer un projecteur, jouer un son, changer le volume, déclencher un *chaser* etc..

Au final le scénario sera toujours entrain de jouer un étape, appartenant à une scène, et de regarder si des changements permettent soit de :

- Changer d'étape
- Changer de scène
- Déclencher une interruption pour jouer une action prédéfinie

La séparation entre scènes et étapes peut paraitre inutile à cause de la proximité de leur fonctionnement, mais elle permettra de mieux structurer le scénario pour éviter de se retrouver à éditer des blocs très long, pouvant devenir trop complexes, ou pour avoir un contrôle assez fin du flux (i.e. de l'étape en cours).

3.3 Les scènes

Les scènes représentent un état du spectacle dans son ensemble. Par exemple une scène pourra correspondre au comportement de journée, une autre au comportement de soirée et d'autres à différents comportements liés à des performances en cours. C'est ici par exemple que ce trouve une des différences avec les étapes, qui elles sont vouées à représenter la logique de ces comportements, dans des échelles de temps souvent plus courtes et ayant tendance à se répéter jusqu'au changement de scène.
Chaque scène aura :

- **Une action d'initialisation**, lorsque le scénario se met à jouer cette scène.
- **Une action de sortie**, lorsque le scénario quitte cette scène.
- **Une série d'étapes**, qui décriront le comportement de la scène.
- **Un ensemble de transitions**, permettant de quitter la scène lorsque certaines conditions sont remplies.

Voici un exemple de scénario sur trois jours. Seul l'enchainement des scènes est montré pour plus de clarté, mais le scénario proposé est volontairement complet.

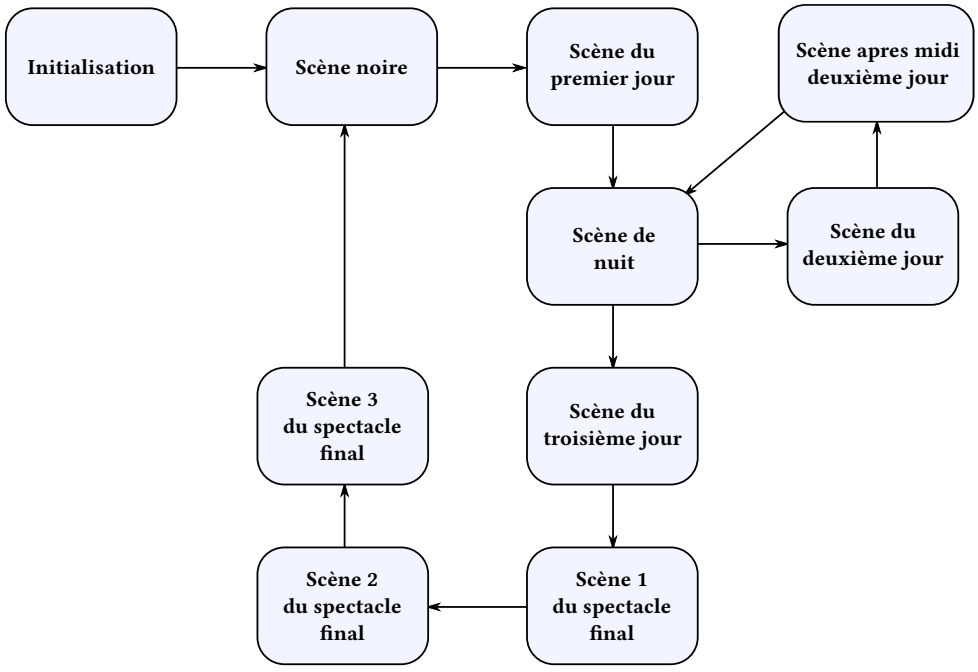

Figure 1 – Exemple d'un scénario complexe

3.4 Les séries d'étapes

Les séries d'étapes sont ce qui va décrire le fonctionnement de la carte à l'intérieur d'une scène. Dans chaque série d'étapes on trouve :

— Une étape de début (l'action d'initialisation de la scène)
— Une étape de fin (l'action de sortie de la scène)
— Aucune, une ou plusieurs étape(s) intermédiaire(s)
— Aucune, une ou plusieurs transition(s)

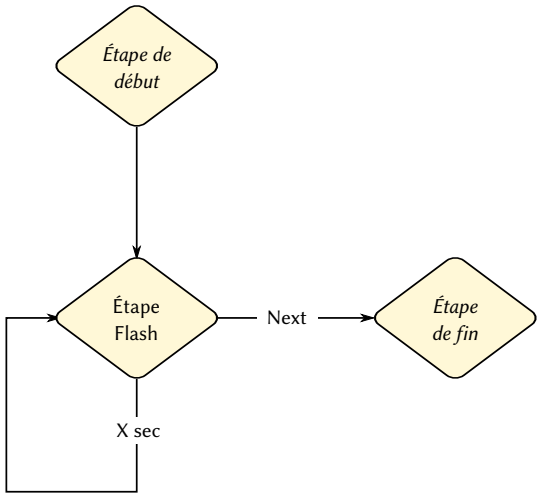

FIGURE 2 – Série d'étapes simples

La figure 2 montre une série de taches toutes simples. Le fonctionnement de cet exemple sera détaillé plus loin, mais cette structure permet de faire clignoter une lumière toutes les X secondes, X étant défini auparavant.

3.5 Les étapes

Une étape est un élément très simple qui comporte un nom, un comportement de début d'étape et un comportement de fin d'étape. Ce sont ces comportements qui font intervenir un langage textuel pour leur définition. L'utilisation d'un langage textuel et non un langage graphique sera argumenté plus tard.

Voici un exemple d'étape :

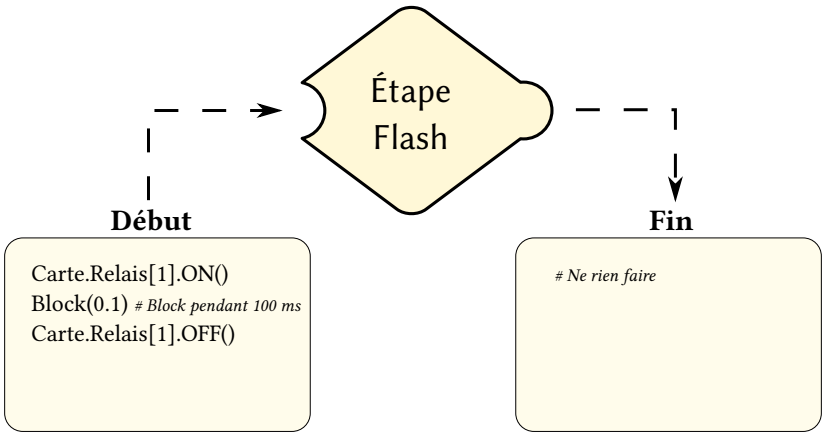

Figure 3 – Étape flash détaillée

Le code de début d'étape est rédigé dans une syntaxe simple, avec des actions toutes préparées.
Ici la première ligne permet de dire à la carte d'actionner sont premier relais. La seconde permet de bloquer le fonctionnement pendant 0.1 seconde, soit 100 millisecondes. La dernière permet d'éteindre le premier relais.
Ce qui est écrit en italic après le symbole # sont des commentaires pour rendre le code encore plus compréhensible si besoin est.

3.6 Les transitions

Les transitions permettent de passer d'une étape à une autre ou encore d'une scène à une autre lorsque certaines conditions sont réunies.

Une transition attendra toujours que l'action de début d'une étape soit finie avant de faire quoi que ce soit. Ceci dans le but de ne pas provoquer de comportements chaotiques ou inattendus. Par exemple, dans l'étape flash présentée plus haut, si la transition *Next* se déclenche alors que l'étape est sur sa seconde ligne, *Block(0.1)*, le relais ne sera jamais éteins car la ligne suivante : *Carte.Relais[1].OFF()* ne sera pas interprétée.

En revanche, les transitions entre scènes peuvent être déclenchées quelques soit l'étape en cours, tant que l'action de début d'étape est terminée évidemment.

Ensuite les transitions entre deux blocs (étape ou scène) sont forcément orientées de l'un à l'autre.

Enfin les transitions sont reliées à des signaux pour que celles-ci sachent quand il est utile de vérifier leur condition ou non, par exemple, un signal se déclenche dès qu'un ordre WIFI arrive, la transition qui attends l'ordre *Next* va donc être lié à ce signal et vérifiera si l'ordre reçu est bien celui-ci, ou un autre.

Une transition ne pourra avoir que deux états, soit oui, soit non. Permettant dans le premier cas le franchissement de celle-ci et dans le second, laissant le scénario tel quel.

Voici un exemple de transition :

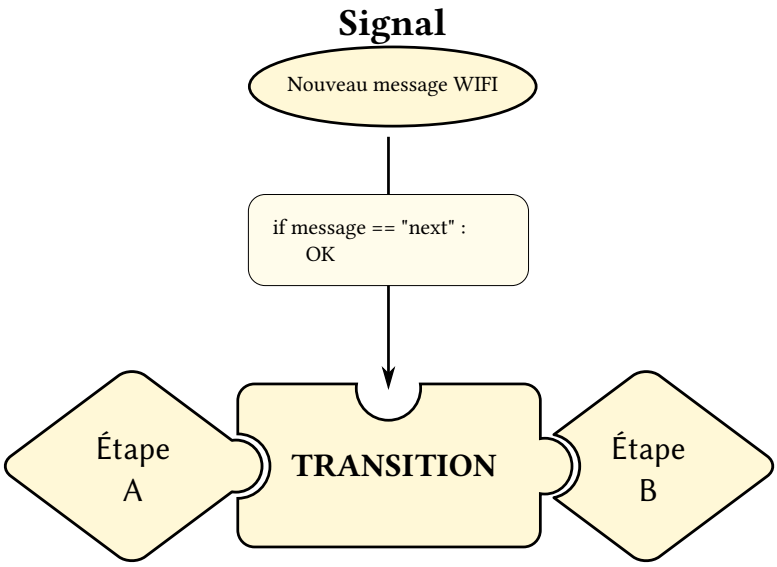

FIGURE 4 – Exemple de transition

3.7 Les interruptions

Les interruptions permettent de garder une grande souplesse dans ce qu'il sera possible de faire via le système de scénario. En effet les étapes se déroulent dans une logique assez linéaire et les transitions ne sont franchies que si l'étape précédente est active. Comment alors perturber ce flux

et par exemple mettre en pause tout le système sur un ordre WIFI, quel que soit l'étape en cours ?

Les interruptions sont là pour répondre à ce type de besoin. Une interruption est proche d'une transition, mais celle-ci peut :
— Être déclenchée, quelque soit l'étape en cours
— Faire une action sans forcément changer l'étape en cours.

Les interruptions sont elles aussi connectées à des signaux qui les réveillent. Une fois réveillées elles vérifient leur condition et si celle-ci est remplie elles exécutes des ordres définis.

Par exemple dans notre petite scène qui fait clignoter une lumière toutes les X secondes, une interruption peut être utilisée pour changer cette valeur de X en fonction d'un ordre reçu.

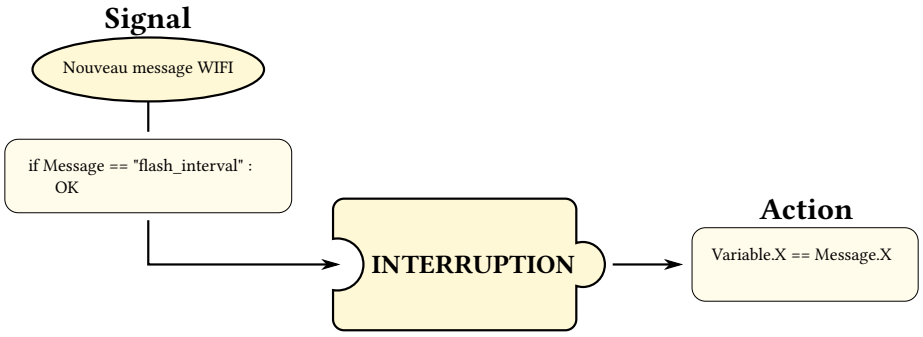

Figure 5 – Exemple d'interruption

4 Exemple d'application : Une poubelle intrigante

Pour illustrer comment répondre à un besoin donné avec ce système de scénario voici un exemple concret.
Imaginons une poubelle qui possède un fonctionnement assez simple. Le jour elle joue une piste sonore en continue et grâce à un capteur de présence diminue le volume du son lorsque quelqu'un se rapproche et l'augmente lorsque celui-ci s'éloigne.

La nuit, la poubelle coasse [9] de temps à autres, et émet une lumière diffuse. Lorsque personne n'est a proximité la lumière est faible, mais lorsque quelqu'un s'approche celle-ci se met à briller de plus en plus.

Voyons comment il est possible de faire en sorte que la carte réponde à ce scénario.

4.1 Le scénario

Tout d'abord on peut identifier deux fonctionnements bien différents. L'un de journée, l'autre de soirée, il y aura donc deux scènes.

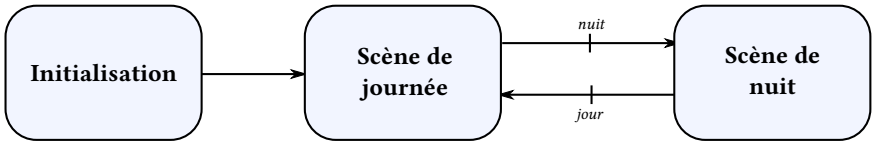

Figure 6 – Scènes de l'exemple poubelle intrigantes

En plus de la scène d'initialisation qui correspond au démarrage de la carte, nous avons bien deux scènes. Une pour le comportement de journée, l'autre pour celui de nuit. Les deux scènes sont liées par deux transitions, une dans chaque sens. Nous allons seulement en détailler une, l'autre étant presque identique.

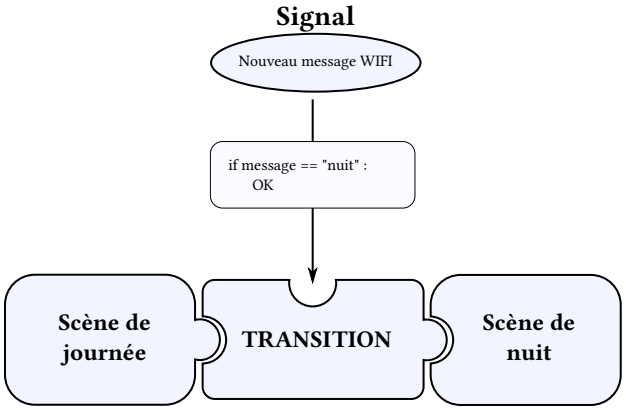

Figure 7 – Transition entre la scène de journée et la scène de nuit

9. Du verbe coasser qui désigne le chant de la grenouille : [lien]

4.2 La scène de journée

Nous allons détailler la série d'étapes de la scène de journée :

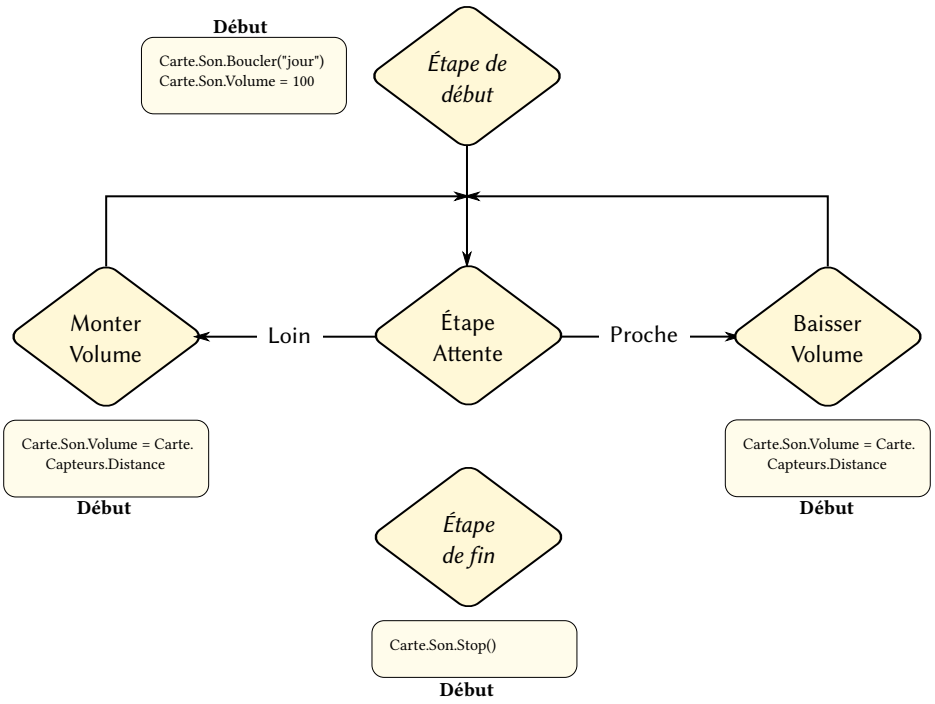

Figure 8 – Série d'étapes de la scène de journée

Plusieurs choses sont intéressantes à remarquer ici :

- Des transitions peuvent être sans condition, elles ne font qu'attendre que l'action de début de l'étape qui les précède soit finie.
- Des étapes peuvent ne recevoir ou n'émettre aucune transition. Ici l'étape de fin est de toute façon jouée lorsque le scénario quitte la scène.
- Une étape peut être vide de toute action. Ici l'étape d'attente est justement là pour attendre que le capteur de distance change de valeur et renvoyer soit sur l'étape qui monte le volume soit sur celle qui le descend.

Nous allons maintenant détailler la transition *Proche*, la transition *Loin* étant très similaire.

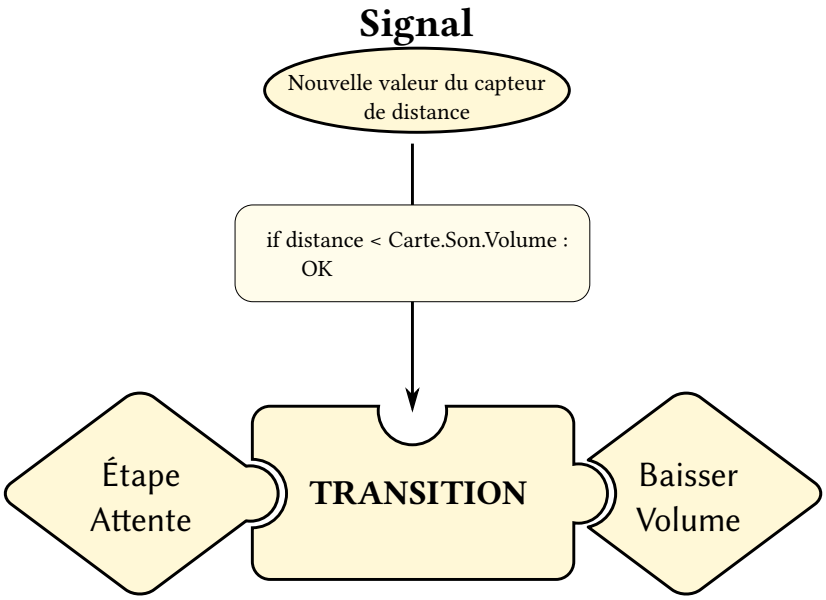

FIGURE 9 – Transition entre l'étape d'attente et l'étape baisser le volume

Rien de surprenant, la transition attends que le signal *Nouvelle valeur du capteur de distance* la réveille, vérifie que cette nouvelle valeur est bien inférieur à l'ancienne et si c'est le cas, déclenche le passage à l'étape *Baisser le volume*.

4.3 La scène de nuit

La scène de nuit est très similaire à la scène de journée, il suffit donc de reprendre la structure et d'en changer quelques éléments.
Voici à quoi ressemblerait la scène de nuit :

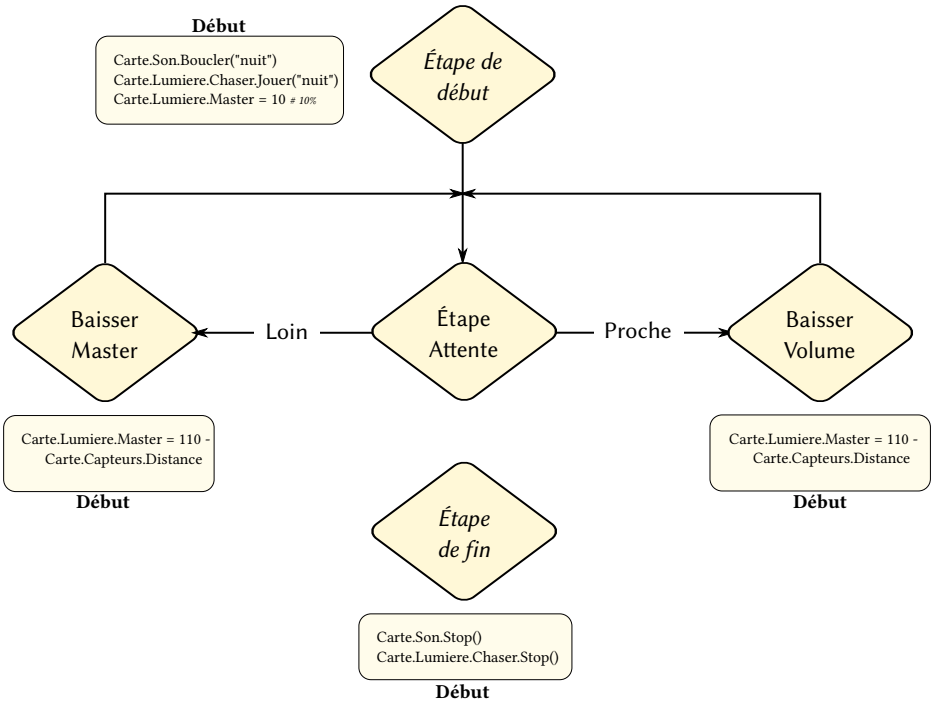

Figure 10 – Série d'étapes de la scène de nuit

4.4 Les interruptions

Aucune interruption n'est nécessaire pour ce scénario. En revanche on peut vouloir figer les poubelles pour que celles-ci ne fassent plus un bruit et n'émettent plus de lumière en leur ordonnant, puis leur permettre de reprendre leur fonctionnement.

Voici une proposition avec une interruption pour la pause, une autre pour reprendre le fonctionnement.

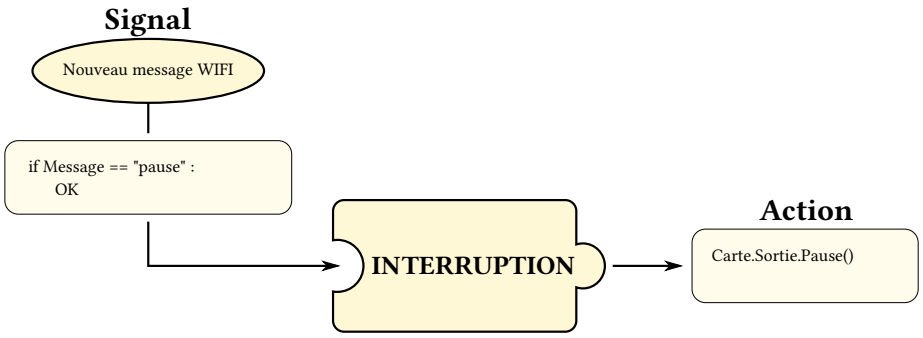

FIGURE 11 – Interruption pour la mise en pause de la carte

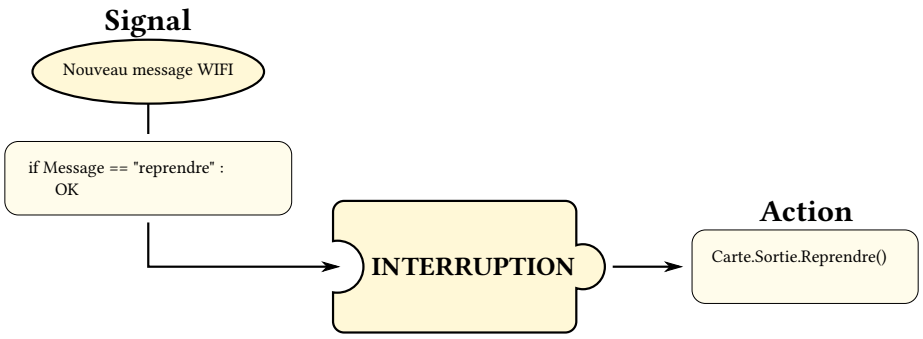

FIGURE 12 – Interruption pour la reprise de la carte

Les interruptions sont simples et permettent de perturber le flux sur un ordre WIFI directe.

5 Considérations techniques

Cette partie peut-être sautée par ce(lles)(ux) qui ne sont pas intéressé(e)s par les aspects purement technique du système

5.1 Latence du système de scénario

Le système en lui même semble assez rapide sur le processeur Dragino. En effet la latence moyenne d'une interruption est de l'ordre de 2 ms. En imaginant qu'une interruption est souvent lié à un ordre il faut pour cela doubler ce temps. En rajoutant le traitement de l'information, même si celui-ci reste dans la majorité extrêmement simple, la plupart du temps une simple condition, pour obtenir une latence de l'ordre de 5 ms pour le système de scénario tournant sur le processeur Dragino.

5.2 Latence de la communication Dragino-Atmega

Il faut également considérer la latence due à la communication entre le processeur Dragino et la carte Atmega. Faute d'avoir pu réussir un essai directe je dois me fier aux informations trouvées sur le Web [10]. Les taux de transfert pour de simples ordres semblent court : dans une fourchette comprise entre 0.1 ms et 5 ms en fonction des technologies utilisées. Majorons donc à 5 ms.

5.3 Latence due au traitement de l'information de l'Atmega

L'Atmega n'aura presque aucun calcul à faire, seulement à exécuter des ordres. Vu que la partie communication à déjà été prise en compte ci-dessus, je pense que l'on peut facilement prévoir des temps de réponses compris entre 1 et 5 ms en fonction de la tache à effectuer. Par exemple un *chaser* sera beaucoup plus lourd à gérer que le déclenchement d'un flash.

5.4 Latence due au réseau WIFI

Cette latence sera la plus variable et aura comme principal paramètre l'état du réseau, la qualité des antennes et le nombre de bornes. Des essais que j'ai pu réaliser avec une seule borne, un réseau d'une qualité moyenne et une antenne peu adaptée j'obtiens une latence d'environs 2 ms. Avec

10. `https://www.pjrc.com/teensy/td_libs_AltSoftSerial.html`

plus de bornes et un signal moins bon je pense que cette latence peut atteindre entre 5 et 8 ms.

5.5 Latence globale

En réunissant toutes les latences majorées on obtiens un système avec un temps de réponse avoisinant les 20 ms ce qui semble un assez bon résultat. Pour s'en convaincre voici quelques ordres de grandeur :

Fréquence d'échantillonage de l'œil :

$$f_{œil} \sim 25 \text{ Hz}$$

Période d'échantillonage de l'œil :

$$T_{œil} = \frac{1}{f_{œil}} \sim 40 \text{ ms}$$

Vitesse du son dans le vide :

$$v_{son} = 340 \text{ m.s}^{-1}$$

Temps de propagation du son dans le vide à 15 mètres

$$t_{son} = \frac{15}{v_{son}} = 44 \text{ ms}$$

Sachant que la plupart de ces latences sont majorées et que tous les déclenchements ne passerons pas par le WIFI, le système pourra paraitre d'une bonne fluidité au yeux du public et des artistes.
De plus, pour les opérations demandant une synchronisation très poussée il sera possible de définir des *latences tampons* permettant d'accuser les différents retards dues à la propagation des ordres et d'obtenir un déclenchement synchronisé avec une grande précision

6 Choix et orientations

Une partie de cette section peut-être sautée par ce(lles)(ux) qui ne sont pas intéressé(e)s par les aspects purement technique du système

Plusieurs choix dans cette propositions ont été fait sans contraintes particulières. Ce sont des cadres qui permettent de donner une forme au système et qui ont été motivés principalement par soucis de simplicité et de temps de développement. En revanche certains choix ont été fait par extrapolation et peuvent être erronés, c'est pourquoi je vais expliquer un petit peu plus ces choix, ce qui les a motivés pour qu'ils puissent être discutés et éventuellement changés.

6.1 Un système pas 100% graphique

Le système de scénario sera en quelque sorte 90% graphique et 10% textuel. Pourquoi garder une petite partie en textuel et ne pas faire tout en graphique ?

Avantages

- Pour la même souplesses dans les possibilités, une solution 100% graphique demande un grand surcout en temps de développement par rapport au compromis de laisser une faible partie textuelle.
- L'avantage du langage textuel est qu'il n'est pas figé et qu'il permet de pousser le système au delà de ses capacités prévues si besoin, sans pour autant modifier la structure de celui-ci.

Inconvénients

- La partie textuelle demande la connaissance d'une syntaxe. Mais cette syntaxe est simple, assez réduite et sera fortement documentée.
- Le plus gros désavantage que je vois est le fait que si il y a une faute de syntaxe (par exemple *Carte.Lumier.Master* à la place de *Carte.Lumiere.Master*) le système provoquera une erreur.

6.2 Pas deux étapes actives en parallèle

Le système sera prévue pour qu'il n'y ai toujours qu'une seule étape active.

Avantages

- Cela réduit grandement les risques de comportements non souhaités, le parallélisme est toujours une grande source de problèmes.
- Cela rend le système plus simple, au niveau de la création et de sa gestion. Il a donc de meilleures chances d'être plus réactif.

L'inconvénient pourrait être de se fermer beaucoup de possibilités car les cartes sont capables de faire du son, de la lumière, déclencher des actionneurs, etc et que de ne pas pouvoir en profiter serait dommage.

En réalité la carte Atmega sera capable de faire plusieurs actions en même temps et les étapes du scénario ne font que déclencher des événements qui ensuite pourront jouer en simultané. De plus le système

d'interruption permet de créer un comportement pseudo-parallèle au niveau des scénarios sans apporter de désavantages majeurs comme le parallélisme.

6.3 Python comme langage de programmation

Rien n'est fixé, mais je pense que python est le langage tout à fait adapté aux besoins du système.

Avantages

- Souple et permet un développement rapide
- Syntaxe claire et variables non typées ce qui permettra de rendre les 10% de syntaxe comprit dans le système plus simple
- La librairie Bridge du Yun Shield est écrite en Python !
- C'est un langage que je maitrise bien et qui est simple à apprendre, ce qui réduira le temps de développement.

Inconvénient

- Langage interprété donc forcément un peu plus lent que le C. Mais c'est un langage qui est très utilisé dans le calcul scientifique et qui est depuis devenu très mature. Il présente donc d'excellentes performances en générale.

6.4 TCP comme protocole de transport selon le modèle OSI

Le modèle OSI [11] propose plusieurs protocoles pour le transport de l'information.
Je propose d'utiliser le protocole TCP pour cette fonction.

Avantages

- Le protocole TCP permet de s'assurer qu'un message a bien été reçu et si n'est pas le cas le protocole se charge de le renvoyer. On a donc quelque chose de robuste et la quasi certitude que chaque ordre sera reçu.
- Le protocole TCP est plus simple à mettre en place, plus utilisé et plus robuste de manière générale (il évite de surcharger le réseau lorsque celui-ci est saturé etc.)

11. `http://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI#Quelques_protocoles`

— Le protocole TCP est intègre. Il s'assure que les données n'ont pas été corrompues pendant le transfert, on ne risque pas d'avoir un ordre Y reçu pour un ordre X envoyé.

Inconvénient

— Le protocole TCP est plus lent que le protocole UDP en général. En revanche il ne présente pas d'énormes latence pour un réseau comme le notre et surtout permet dans de nombreux cas d'être plus rapide que l'UDP de part sa bonne gestion du réseau [12]. Cela reste tout de même à vérifier sur des tests à plus grande échelle.

7 Conclusion

Ce document a donc essayé de présenter le plus clairement possible une solution pour le système de scénario. Cette solution est bien évidement une proposition, et pleins de choses restes à définir. En revanche c'est une proposition qui est techniquement viable et j'ai déjà fait plusieurs essais de code pour tester cette logique d'interruption de transition et d'autres concepts mis ne jeu par le système présenté.

Pour que je puisse avancer au plus vite sans non plus me lancer dans une direction qui ne serait pas la bonne il me faut donc savoir si ce système, aux modifications prêt évidement, correspond bien aux attentes du projet.

12. http://stackoverflow.com/questions/47903/udp-vs-tcp-how-much-faster-is-it