Avec l'aide de l'intelligence artificielle

CINQUIÈME PARTIE : UN BRIN D'INTELLIGENCE ARTIFICIELLE

EX_MACHINA

Un réseau automatisé

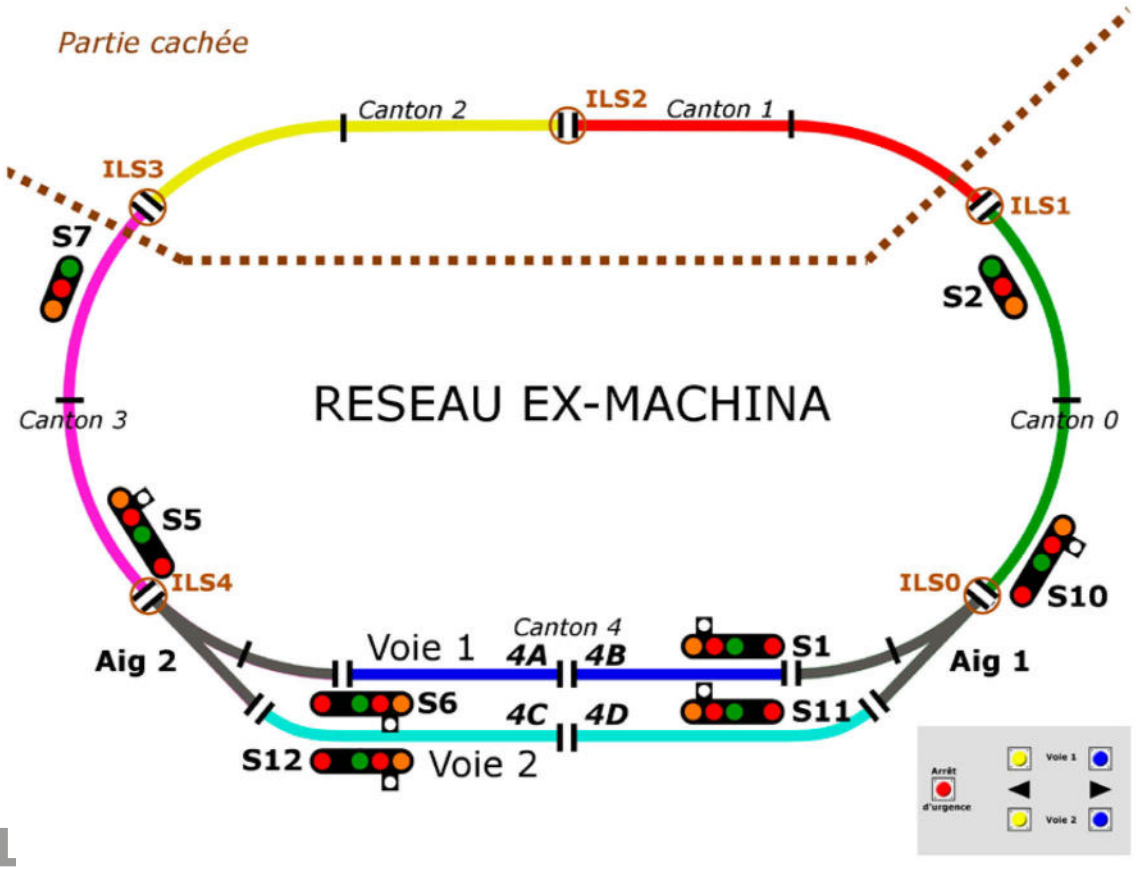
Jusque-là, nous avons fait appel à l'Intelligence Artificielle pour nous aider à écrire des fonctions ou des programmes et pour trouver des algorithmes. Nous allons aujourd'hui gérer le mouvement des deux trains en fonction des choix effectués, quels que soient les sens de circulation demandés.

Texte et illustrations: Christian Bézanger

Il est évident que sur une voie unique, deux trains ne peuvent pas se déplacer en même temps dans des sens différents : le deuxième train doit attendre en gare que le premier train ait fini son circuit avant de pouvoir entamer le sien. Par contre, si les sens de circulation sont les mêmes, le deuxième train peut sortir dès que possible pour suivre le premier. La carte Arduino va le permettre.

État des lieux

Si vous avez été suffisamment courageux pour tenter l'aventure que je vous proposais (à partir de LR), vous avez entre les mains EX_MACHINA, le premier réseau conçu et géré par une IA. La **figure 1** montre ses différents éléments constitutifs : la voie unique composée de 4 cantons, deux aiguilles motorisées à mouvement lent (réglable en durée), deux voies en gare où les trains s'arrêtent en étant approximativement centrés par rapport au quai, une signalisation lumineuse fonctionnelle, et le tableau de commande permettant au joueur de passer ses requêtes. Les signaux peuvent être à cathodes ou à anodes communes et les relais peuvent être commandés par un signal HIGH ou LOW. Tout a été pensé pour une certaine évolutivité, notamment les cantons qui sont isolés sur les deux files de rails comme l'exigent certains équipements modernes. Si vous en êtes arrivés là avec l'aide d'une IA, je suis sûr



1

que cette expérience vous a énormément appris et je ne doute pas que vous pourrez continuer à améliorer ce réseau, lui installer des aiguilles à pointes de cœur polarisables, ou bien lui adjoindre un TCO virtuel sur écran d'ordinateur où la progression des trains serait parfaitement suivie ainsi que le report de la signalisation. Il nous reste une chose à faire : rendre notre carte Arduino intelligente pour prendre les décisions qui s'imposent.

Que veut-on faire ?

Une fois de plus, il faut se poser la question du but que nous voulons atteindre. Notre tableau de commande doit permettre de choisir le premier train au départ (voie 1 ou voie 2) et son sens de circulation, puis le sens de circulation du deuxième train. Une fois ces données connues, le système fait faire un « tour de circuit » à chaque train et ensuite attend vos nouvelles consignes. EX_MACHINA propose huit scenarii de jeux possibles : quatre possibilités pour choisir le premier train et son sens de circulation et pour chaque possibilité, deux choix possibles pour le sens de circulation du deuxième train. Et à chaque fois, l'automatisme prend en charge la circulation des deux trains. Pour un si petit circuit, ce n'est déjà pas si mal !

La première chose à faire est de demander à l'IA de modifier la fonction qui analyse le tableau de commande, afin de déterminer le scénario complet avant de faire partir le premier train. Arduino décide ensuite si les deux trains peuvent ou non circuler en même temps. Si oui, il lance les mouvements avec les opérations qui s'imposent. Si non, il suit le mouvement du premier train jusqu'à son retour en gare et quand il est arrêté, il fait partir le deuxième train et il le suit jusqu'à son retour en gare. Facile non ?

Problèmes à résoudre

Jusqu'à présent, un train du locodrome part dès qu'on a appuyé sur un des quatre poussoirs pour choisir la voie et le sens de circulation. Cette fois, nous voulons donner nos ordres pour les deux trains : un premier problème à résoudre est d'attendre que tous les choix soient faits avant le moindre mouvement de train.

Un deuxième problème à résoudre, c'est à quel moment peut-on faire partir le deuxième train ? Si les sens de circulation sont les mêmes, le premier train doit être complètement sorti de la gare, voire un peu éloigné avant de positionner les aiguilles pour faire partir le deuxième train. Si les sens de circulation sont opposés, à quel moment sait-on que le premier train a fini son tour et est rentré en gare ?

Un troisième problème à résoudre consiste aussi à ne pas louper de survol d'I.L.S pendant le mouvement des aiguilles. Autrement dit, le mouvement du deuxième train ne doit pas interférer avec la surveillance du mouvement du premier et réciproquement.

Fonction d'acquisition des ordres de départ

Cette fonction ne doit être appelée qu'une seule fois au début de chaque nouveau scénario et ne plus être appelée ensuite dès que les ordres de départ ont été donnés. Elle sert à définir les paramètres suivants :

- Premier train à partir (ce qui permet de déduire le deuxième train)
- Sens de circulation du premier train
- Sens de circulation du deuxième train

Ces données peuvent être affichées dans le moniteur pendant la phase de mise au point de la fonction afin de contrôler que l'automate comprend bien ce qu'on attend de lui.

L'IA va donc définir de nouvelles variables pour chacun des deux trains : sens de circulation, voie de départ, etc. ainsi que pour contrôler si les choix sont terminés ou encore en cours. ▶▶

2

Repère	Actions
Sens de circulation identiques	
Fin de examConsignes()	voieActive = voieDepartT1 sensCircul = sensCirculT1 voieDepartT2 sensCirculT2 mouvementAiguilles(voieDepartT1) departTrain() ***Le premier train part sur le locodrome***
T1 sur ILS2 ?	mouvementAiguilles(voieDepartT2) departTrain() ***Les deux trains tournent sur le locodrome***
Sens antihoraire T1 sur ILS4 ? Top arrivee en gare	arriveeGare(T1) ***Le premier train s'arrête en gare sur voieDepartT2***
Sens horaire T1 sur ILS0 ? Top arrivee en gare	arriveeGare(T1) ***Le premier train s'arrête en gare sur voieDepartT2***
10 secondes après top arrivee en gare	mouvementAiguilles(voieDepartT1), arriveeGare(T2) ***Le deuxième train entre et s'arrête en gare sur voieDepartT1***
Fin du tour de circuit	**Arduino attend de nouvelles consignes de jeu**
Sens de circulation contraires	
Fin de examConsignes()	voieActive = voieDepartT1 sensCircul = sensCirculT1 voieDepartT2 sensCirculT2 mouvementAiguilles(voieDepartT1) departTrain() ***Le premier train part sur le locodrome***
Sens antihoraire T1 sur ILS4 ? Top arrivee en gare	arriveeGare(T1) ***Le premier train s'arrête en gare sur voieDepartT1***
Sens horaire T1 sur ILS0 ? Top arrivee en gare	arriveeGare(T1) ***Le premier train s'arrête en gare sur voieDepartT1***
10 secondes après top arrivee en gare	voieActive = voieDepartT2 sensCircul = sensCirculT2 mouvementAiguilles(voieDepartT2) departTrain() Remise à 0 flagEvent. ***Le deuxième train part sur le locodrome***
Sens antihoraire T2 sur ILS4 ? Top arrivee en gare	arriveeGare(T2) ***Le deuxième train s'arrête en gare sur voieDepartT2***
Sens horaire T1 sur ILS0 ? Top arrivee en gare	arriveeGare(T2) ***Le deuxième train s'arrête en gare sur voieDepartT2***
Fin du tour de circuit	**Arduino attend de nouvelles consignes de jeu**

▸▸ Certaines variables sont à réinitialiser à chaque début de scénario. Finalement, l'IA arrive assez bien à écrire cette fonction qui scrute les poussoirs et détermine les paramètres. Cette fois, des boucles while permettent d'attendre que les consignes soient passées : on les utilise quand on ne connaît pas le nombre d'itérations que la boucle doit effectuer, et tant que la condition entre parenthèses est vraie, la boucle continue de s'exécuter.

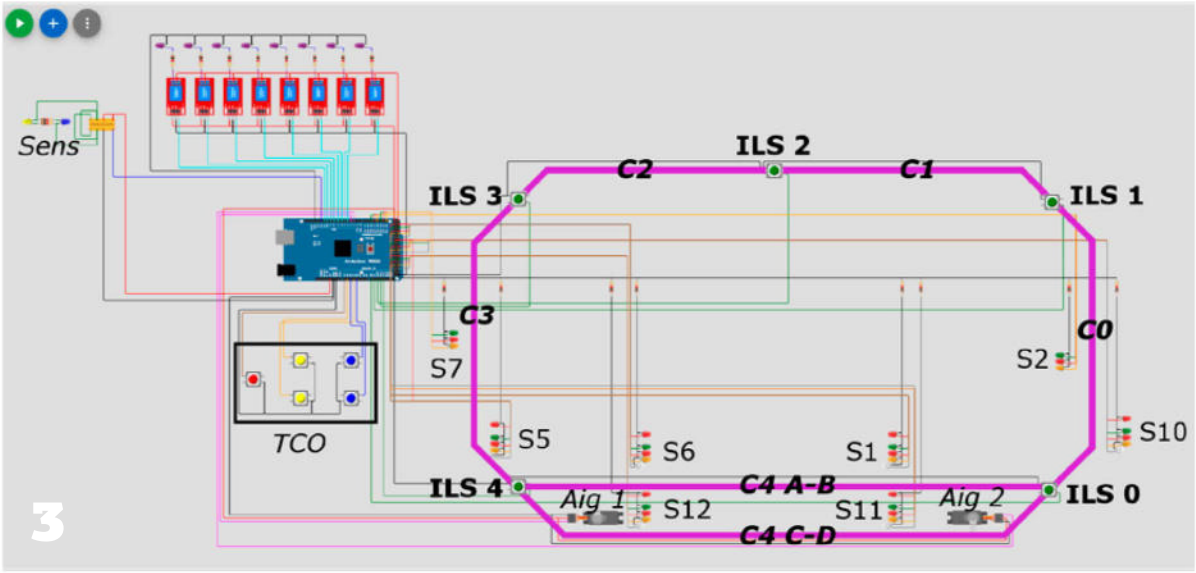
Le premier problème semble résolu et il ne reste plus qu'à agir dans la fonction loop pour exécuter toutes les opérations nécessaires à la conduite des trains. Pour cela, il faut demander à l'IA de découper les « tours de circuit » en différentes opérations et d'utiliser les fonctions déjà écrites :

- Scrutation des I.L.S
- Détermination du tableau des états des cantons
- Alimentation ou non des cantons
- Traitement des signaux

Algorithme d'intelligence artificielle pour Arduino

Pour contrôler ce que l'IA va produire, il faut définir exactement ce que doit faire l'automatisme pour gérer les deux trains. Si les sens de circulation sont identiques, on peut faire sortir le deuxième train, ce qui implique une action sur les aiguilles puis mettre le courant sur la voie. Mais pour que cela se fasse en toute sécurité, il faut laisser le premier train s'éloigner un peu. L'avancement du premier train est surveillé par le déclenchement des ILS : lorsque l'I.L.S 2 est déclenché, c'est que le train est le plus éloigné de la gare quel que soit le sens de circulation et il est temps de faire sortir le deuxième train. On peut ensuite laisser les deux trains se suivre jusqu'à un possible retour en gare. L'IA aura à produire une fonction pour faire partir un train de la gare et une autre fonction pour le faire arriver en gare, sachant que l'endroit où le train s'arrête dépend de son sens de circulation. Le niveau de protection du locodrome arrêtera le deuxième train jusqu'à ce qu'un mouvement d'aiguilles lui permette de rentrer sur la voie qui reste libre pour s'y arrêter aussi.

Si les sens de circulation sont contraires, on laisse le premier train tourner jusqu'à son retour en gare où l'arrêt se fera si l'alimentation est coupée sur le demi-canton concerné. On peut alors, après un certain délai, faire partir le deuxième train, après avoir positionné les aiguilles et alimenté la voie. Le train effectue son tour puis vient se ranger là d'où il est parti et l'automatisme attend alors de nouveaux ordres. La **figure 2** résume les actions à faire. Si on regarde la colonne « Repère » de cette **figure 2**, on voit que les actions sont à faire lorsque les trains passent l'I.L.S 2 ou bien arrivent en gare. L'Intelligence Artificielle propose de repérer la progression des trains sur le circuit : pour cela, elle utilise un drapeau (flag en anglais) qui vaut 0 avant déclenchement d'un I.L.S et 1 une fois que l'I.L.S a été déclenché. Les valeurs sont stockées dans un tableau appelé flagEvent []. Par exemple flagEvent [2] est mis à 1 lorsque l'I.L.S 2 est déclenché, et le remplissage du tableau flagEvent permet de situer où se trouve le train (plus il y a de 1 et plus le train progresse). Pour agir lorsque l'I.L.S 2 est déclenché, il suffit de regarder flagEvent [2] : s'il vaut 0, on est dans la configuration de départ permettant au premier train de partir, s'il vaut 1, c'est que l'I.L.S 2 a été déclenché et on peut faire partir le deuxième train. Il n'y a qu'à la fin du tour de circuit qu'on remet le drapeau à 0 (pour une autre séquence de jeu). Avec ce système, l'IA a résolu le deuxième problème. Pour être sûr qu'un train est bien entré en gare, on peut utiliser des temporisations : il est même possible d'utiliser la fonction delay () mais uniquement si aucun train ne circule.



Résoudre le troisième problème est assez facile : le temps pour effectuer les actions nécessaires (alimentation des voies, mouvement des aiguilles) doit être inférieur au temps de parcours d'un canton. Ainsi, on est certain que les opérations seront bien terminées avant que le train n'atteigne l'I.L.S suivant. Un mouvement d'aiguilles se fait en deux secondes (temps réglable) ce qui laisse du temps avant que le train n'arrive à l'I.L.S suivant. Quant au traitement des relais ou des signaux, c'est extrêmement rapide et ce n'est donc pas un problème.

Il est facile de demander à l'IA d'écrire le programme pour respecter les actions définies dans la **figure 2**. Il faut ensuite tester le tout soit dans le simulateur Wokwi (voir plus loin), soit directement sur le réseau, ce qui est préférable. On peut alors noter la moindre anomalie et chercher à la réparer avec l'aide de l'IA. Parfois, il faut peu de choses, juste un petit détail oublié. Par exemple, en faisant des tests sur le prototype, j'ai rencontré un bug qui ne se produisait que pour un scénario : un appel d'une fonction était mal placé. Plutôt que de demander à l'IA de réécrire le programme, j'ai corrigé de moi-même. Comme je l'ai dit dans le premier article, l'IA a bien écrit plus de 90 % du code.

Concernant le réseau EX_MACHINA

Sur le plan matériel, le réseau EX_MACHINA ne représente pas une grosse dépense pour celui qui veut pratiquer le modélisme ferroviaire. Construire ce réseau, c'est aussi une excellente occasion pour découvrir les techniques appliquées à Arduino (ou autres cartes à microcontrôleur) avec un professeur (l'IA) dont la patience vous étonnera. Ces techniques vous seront utiles pour un autre projet plus conséquent. Je ne peux pas, dans cette série d'articles, vous commenter toutes les lignes de code écrites par l'IA (il y en a presque un millier !) et de toute façon, l'IA donne déjà des explications lorsqu'elle fournit un programme. C'est pourquoi je me suis contenté de vous expliquer les grands principes techniques du réseau ainsi que la façon de travailler avec une IA. J'ai parfois commenté certaines façons d'écrire le code qui peuvent dérouter un non spécialiste en programmation. Pour d'autres explications de ma part, je vous donne rendez-vous sur le forum de *Loco-Revue* (https://forum.lrpresse.fr/viewtopic.php?t=98474). Voici maintenant la liste des fonctions écrites par l'IA : j'ai parfois modifié le nom et celui-ci est assez explicite pour comprendre l'utilité de la fonction :

examConsignes – arretURGENCE – inversionFeeder – initLocodrome – departTrain – arriveeGare – extinctionSignaux – setSignal – setBAL – affichageSignaux – mouvementAiguilles – moveServosSimultaneously – scrutationILS – determinationEtatsCantons – gererCompteurs – mettreAJourRelais

La **figure 3** montre l'ensemble des composants d'EX_MACHINA sur le simulateur Wokwi. Contrairement à la **figure 8 de l'article 4 (LR 939)**, les composants (I.L.S, signaux) ont été placés comme sur le locodrome et les voies sont sommairement représentées en violet. Les servomoteurs des aiguilles pointent vers la voie sélectionnée (à l'horizontal : voie 1, vers le bas : voie 2). Le lien < https://wokwi.com/projects/435830312682088449 > vous permettra de tester ce simulateur, donc ce que ChatGPT et moi avons réalisé pour faire tourner EX_MACHINA. Passez en mode plein écran et réglez la dimension de l'image de manière à voir tout le circuit. Passez en mode simulation et faites avancer vos trains en cliquant sur les poussoirs I.L.S dans le bon ordre selon le sens de circulation. Vous pourrez comparer nos solutions avec celles de votre IA et aussi récupérer certaines fonctions pour les utiliser dans votre projet, quel qu'il soit.

***Attention** : ce logiciel est livré en l'état. Ni moi ni LR-Presse ne serons tenus responsables en cas de dysfonctionnement ou en cas de dommage matériel ou corporel si vous n'avez pas suivi les règles de l'art en électricité et électronique.*

Conclusion

Cette série d'articles a essayé de démontrer que concevoir un automatisme à base d'Arduino n'est pas si compliqué, même si on n'est pas un expert d'Arduino. L'Intelligence Artificielle est un outil que nous devrons dans l'avenir apprendre à utiliser. Un champ nouveau de possibilités s'ouvre à nous et il faut en profiter. J'espère que développer en collaborant avec une IA vous a fait découvrir quelques techniques possibles dont vous pouvez vous inspirer pour d'autres choses. Avec un peu de méthode et de pugnacité, vous serez capable de concevoir tous les automatismes que vous voulez pour votre réseau, des plus simples au plus compliqués. Lancez-vous et bon développement ! ■