FONDATION La main à la pâte
POUR L'ÉDUCATION À LA SCIENCE

Cycle 4

Améliorer le trafic routier : robotique avec Thymio

Une séquence du projet *1,2,3... CODEZ !*

Résumé

Cette séquence de technologie utilise des robots Thymio pour simuler un événement courant de la circulation routière : le dépassement d'un véhicule par un autre. Elle permet de réfléchir aux caractéristiques de la « voiture autonome ». Le travail peut bien évidemment se prolonger par une séance d'ASSR (attestation scolaire de sécurité routière).

Projet « robotique avec Thymio »

Ce projet repose quasi-exclusivement sur la programmation d'un robot Thymio II, que nous appellerons désormais simplement « Thymio ». Il existe bien évidemment de nombreux autres robots éducatifs (Lego Mindstorm, mBot, Poppy, etc.) auxquels il est possible d'adapter le principe de cette séquence (attention toutefois : une telle adaptation nécessite de nombreuses modifications pour tenir compte des spécificités de ces robots).
Notre choix s'est porté sur le robot Thymio car il est utilisable de façon intéressante de la maternelle au lycée et même au-delà : des étudiants de Master l'utilisent aussi, comme nous le verrons en Séance 2 ! Thymio est robuste, non anthropomorphe – ce qui permet d'aller au-delà de la première intuition des élèves sur ce qu'est un robot – et se programme à l'aide de différents langages : en *VPL* dès le cycle 2, en *VPL* avancé ou *Blockly* dès les cycles 3 et 4, et dans un langage de programmation textuelle, *Aseba*, au-delà[1].

Discipline concernée et liens avec les programmes

Les programmes 2016 de technologie comportent un chapitre « informatique et programmation » qui justifie parfaitement un projet de robotique (nous ne citons pas ici les compétences transversales qui sont travaillées tout au long de ce projet) :

L'informatique et la programmation

– Écrire, mettre au point et exécuter un programme
• Analyser le comportement attendu d'un système réel et décomposer le problème posé en sous-problèmes afin de structurer un programme de commande.
• Écrire, mettre au point (tester, corriger) et exécuter un programme commandant un système réel et vérifier le comportement attendu.
• Écrire un programme dans lequel des actions sont déclenchées par des événements extérieurs.
  - Notions d'algorithme et de programme.
  - Déclenchement d'une action par un évènement, séquences d'instructions, boucles, instructions conditionnelles.
  - Capteur, actionneur, interface.
• Observer et décrire le comportement d'un robot ou d'un système embarqué. En décrire les éléments de sa programmation.
– Autres compétences :
• Développer les bonnes pratiques de l'usage des objets communicants.
• Logiciels de *mind-mapping*.

1. À l'heure où nous écrivons ces lignes, une version de *Scratch* pour Thymio est en cours de développement. *Scratch* et *Blockly* partagent de nombreux principes communs (programmation par blocs) : il sera facile d'adapter la séquence *Blockly* en une séquence *Scratch*.

Le professeur de technologie peut mener le projet seul, dans une version compacte ou plus développée selon le temps qu'il souhaite consacrer aux éléments de programme correspondants. En effet, les quatre premières séances constituent à elles seules un mini-projet de robotique cohérent. Nous conseillons néanmoins de mener l'ensemble du projet, afin d'approfondir et de consolider les notions en jeu, et de laisser les élèves exprimer davantage de créativité.

Objectifs

Cette séquence de technologie utilise des robots Thymio pour simuler un évènement courant de la circulation routière : le dépassement d'un véhicule par un autre. Elle permet de réfléchir aux caractéristiques de la « voiture autonome ».
Cette séquence peut bien évidemment se prolonger par une séance d'ASSR (Attestation scolaire de sécurité routière).

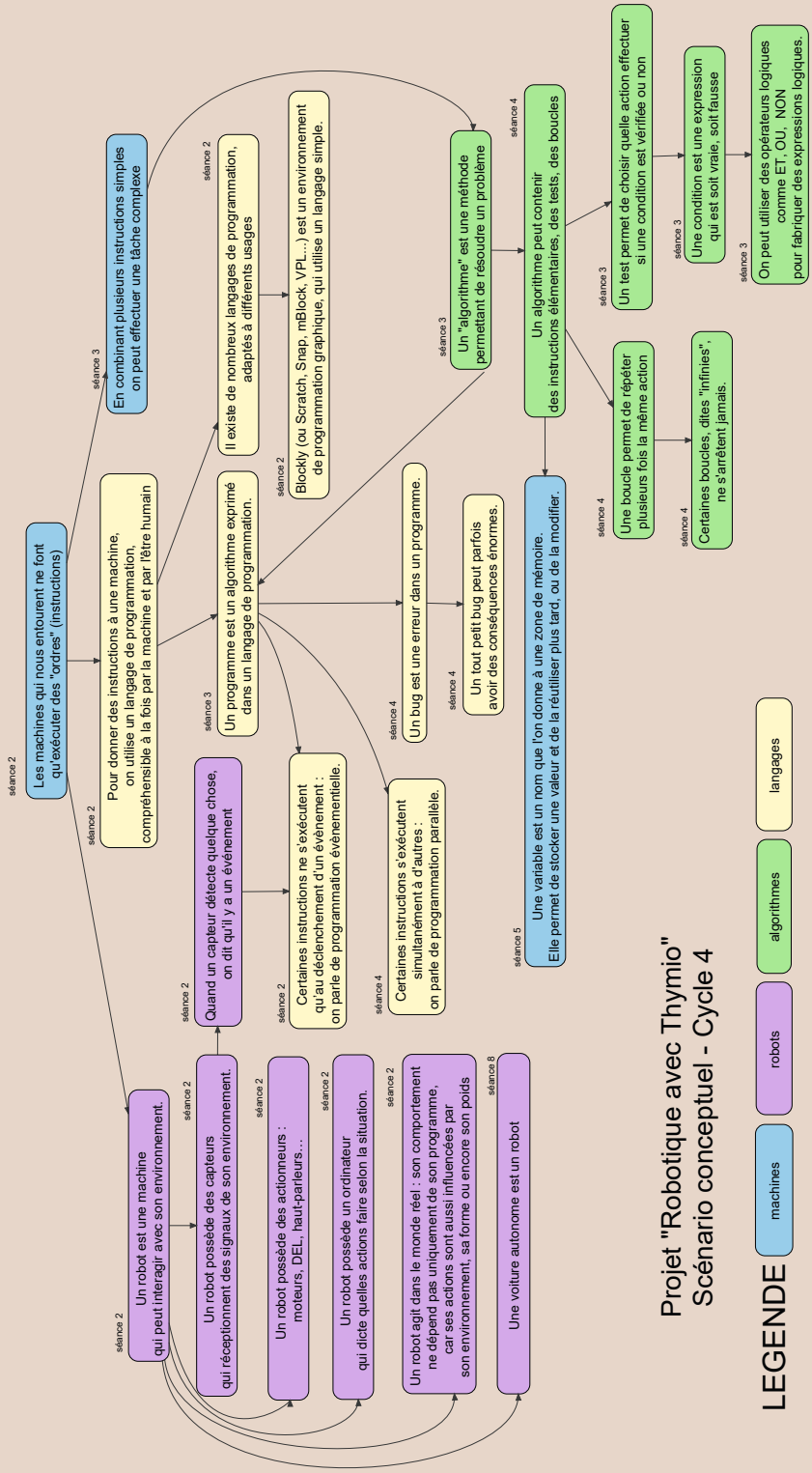
Notes pédagogiques

• La programmation avec *Aseba/VPL* ou *Aseba/Blockly* repose sur les mêmes principes, cependant chaque langage propose une approche ou des raccourcis qui lui sont propres, et qu'il faut donc apprendre au cas par cas.
• Nous proposons ici une approche avec *Blockly*, ce qui suppose que les élèves ont déjà quelques notions de programmation par blocs (*Scratch*, par exemple) et ont déjà manipulé des variables ou des opérateurs logiques. Pour les classes qui souhaiteraient programmer Thymio en *VPL* plutôt qu'en *Blockly*, nous proposons également une variante axée sur ce langage (page 338).

	Séance	Titre	Page	Résumé
	Séance 1	Comment améliorer le trafic routier ?	303	Suite à la visualisation de documents sur les embouteillages, les élèves réfléchissent aux différentes solutions qui permettraient d'améliorer le trafic routier. Ils les organisent ensuite sous la forme d'une carte mentale.
	Séance 2	Découvrir Thymio	306	Les élèves découvrent Thymio, un robot que des étudiants de Master utilisent pour simuler un réseau routier. Ils se l'approprient, et commencent à le programmer par eux-mêmes avec *Blockly*, un langage de programmation graphique.
	Séance 3	Programmer un évitement d'obstacle	311	Les élèves apprennent à manipuler des opérateurs logiques, et programment Thymio pour lui permettre d'éviter un obstacle.
	Séance 4	Programmer un suiveur de ligne	315	Les élèves programment eux-mêmes un comportement qui imite le mode « suiveur de ligne » de Thymio. Ils décrivent tout d'abord l'algorithme de ce mode, puis l'implémentent avec *Blockly*.
	Séance 5	Programmer un suiveur de ligne amélioré	319	Pour que Thymio puisse recueillir des informations sur son chemin via des codes-barres, les élèves modifient leur suiveur de ligne : il n'utilise désormais qu'un seul capteur de châssis au lieu de deux.
	Séance 6	Programmer le lièvre et la tortue	327	La première simulation de circulation routière des élèves reprend le thème d'une fable de La Fontaine. Il n'y a pas de véritable enjeu tant que le lièvre et la tortue sont sur des pistes séparées. En revanche, lorsqu'ils partagent la même piste, les élèves doivent prévenir les accidents.
	Séance 7	Autoriser le lièvre à doubler la tortue	331	Pour laisser une chance au lièvre, tout en évitant les accidents, les élèves doivent programmer le changement de piste du Thymio rapide, s'il est trop ralenti par le Thymio lent.

	Séance 8	Bilan : avantages et inconvénients de la voiture autonome	336	Les élèves organisent un débat pour échanger sur les avantages, les inconvénients, les risques et les possibilités des voitures autonomes.
	Variante	Faire le projet avec *Aseba*/VPL	338	Corrigé des principales étapes du projet avec VPL avancé.

Scénario conceptuel



Séance 1 – Comment améliorer le trafic routier ?

Discipline dominante	Technologie
Résumé	Suite à la visualisation de documents sur les embouteillages, les élèves réfléchissent aux différentes solutions qui permettraient d'améliorer le trafic routier. Ils les organisent ensuite sous la forme d'une carte mentale.
Notions *(cf. scénario conceptuel, page 302)*	
Matériel	Pour la classe : • Un vidéoprojecteur • Un logiciel de *mind-mapping*

Situation déclenchante

Afin de déclencher un brainstorming sur le thème du trafic routier, le professeur diffuse au choix :
– une photo ou une vidéo d'embouteillage (voir par exemple la Fiche 1) ;
– une simulation de trafic routier (les animations multimédia sont très bien faites sur ce site : http://www.traffic-simulation.de/index.html) ;
– toute autre source documentaire sur le même thème.

Le professeur demande aux élèves ce que le document leur inspire. Le professeur rebondit sur les propositions des élèves concernant la sécurité routière ou le trafic routier, en posant la question suivante : *Comment pourrait-on améliorer le trafic routier ?*

Brainstorming : comment améliorer le trafic routier ? (classe entière)

Les élèves débattent librement des pistes qu'ils envisagent. Le professeur les note au tableau, sans classement. Voici un exemple d'idées qui peuvent être évoquées. On peut les regrouper, mais ce travail sera réalisé à l'activité suivante :

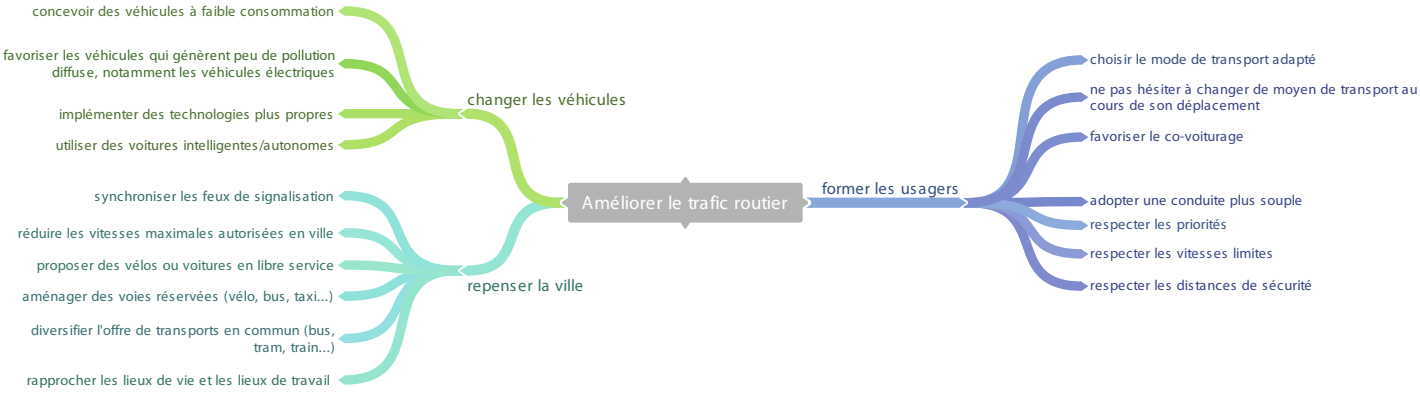
- réduire la vitesse limite
- imposer des distances de sécurité
- synchroniser les feux de signalisation
- aménager des voies dédiées
- utiliser des voitures intelligentes/autonomes
- s'arrêter aux obstacles
- respecter les priorités à droite
- privilégier les véhicules/voies prioritaires
- implémenter des technologies plus propres

- éduquer à une conduite plus souple (pas de freinages ni d'accélérations brusques)
- mieux utiliser son véhicule (covoiturage)
- prendre les transports en commun ou le vélo (c'est l'intermodalité)
- ne pas hésiter à changer de moyen de transport au cours de son déplacement (c'est la multimodalité)

Réalisation d'une carte mentale

Les élèves doivent maintenant organiser leurs idées sous forme d'une carte mentale. Il existe de nombreuses possibilités : papier-crayon, post-it, logiciels de *mind-mapping*, etc. Selon l'organisation de la classe et le degré d'autonomie des élèves, ce travail peut s'effectuer en classe entière ou par groupes.

Voici un exemple de réalisation, produit avec le logiciel en ligne gratuit Coggle :



Notes pédagogiques

- Il est tout à fait possible de réaliser un travail similaire en reprenant l'approche, plus classique en technologie, du diagramme-pieuvre©.
- Il n'existe évidemment pas de solution unique, chaque classe peut aboutir à sa propre carte, la solution proposée ici n'est qu'un exemple de ce qui peut être produit.

Conclusion

Pour améliorer le trafic routier, il faut commencer par le comprendre. Cette compréhension passe par l'observation, mais aussi par la modélisation. La séquence qui va suivre se concentrera justement sur les bases de la modélisation de la circulation automobile grâce à la robotique.

FICHE 1
Embouteillage monstre à Delhi (Inde)

Séance 2 – Découvrir Thymio

Discipline dominante	Technologie
Résumé	Les élèves découvrent Thymio, un robot que des étudiants de Master utilisent pour simuler un réseau routier. Ils se l'approprient, et commencent à le programmer par eux-mêmes avec *Blockly*, un langage de programmation graphique.
Notions *(cf. scénario conceptuel, page 302)*	Machines : • Les machines qui nous entourent ne font qu'exécuter des « ordres » (instructions). Robots : • Un robot est une machine qui peut interagir avec son environnement. • Un robot possède des capteurs qui réceptionnent des signaux de son environnement. • Un robot possède des actionneurs : moteurs, DEL, haut-parleurs… • Un robot possède un ordinateur qui dicte quelles actions faire selon la situation. • Un robot agit dans le monde réel : son comportement ne dépend pas uniquement de son programme, car ses actions sont aussi influencées par son environnement, sa forme ou encore son poids • Quand un capteur détecte quelque chose, on dit qu'il y a un événement Langages : • Il existe de nombreux langages de programmation, adaptés à différents usages • Pour donner des instructions à une machine, on utilise un langage de programmation, compréhensible à la fois par la machine et par l'être humain • *Blockly* (ou *Scratch, Snap, mBlock, VPL*…) est un environnement de programmation graphique, qui utilise un langage simple. • Certaines instructions ne s'exécutent qu'au déclenchement d'un évènement : on parle de programmation évènementielle.
Matériel	Pour la classe : • Un vidéoprojecteur Par groupe : • Un robot Thymio • Un ordinateur avec *Aseba Studio* installé

Avant-propos

Il existe de nombreuses façons de programmer Thymio. Nous utilisons ici *Aseba Studio/Blockly*, que nous appellerons désormais plus simplement « *Blockly* » même si c'en est une version très spécialisée (l'interface *Blockly* n'est disponible que depuis la version 1.5 d'*Aseba*). Nous fournissons également des corrigés utilisant *Aseba Studio*/VPL, spécifiquement conçu pour Thymio, en variante (page 338). Il est de même possible d'utiliser des plugins pour *Scratch* ou *Snap !* ou *mBlock* pour ce faire, mais le professeur devra adapter de lui-même les corrigés que nous proposons ici.

Préparation

Télécharger le logiciel *Aseba Studio* à partir de la page https://www.thymio.org/fr:start, qui permet de choisir son environnement de travail. Ce logiciel est gratuit et disponible pour les systèmes d'exploitation Windows, Mac et linux. L'installation ne pose aucun problème : il suffit d'accepter les options par défaut (en particulier, choisir l'option « pour Thymio II (recommandée) ».

Pour lancer *Aseba Studio/Blockly*
1. Brancher Thymio sur l'ordinateur avec le câble USB (il s'allume) ; **2.** Lancer *Aseba Studio* pour Thymio ; **3.** [Optionnel] dérouler les onglets prox.horizontal (ce qui permet de lire les valeurs des 7 capteurs sur le pourtour du robot) et prox.ground.delta (ce qui permet de lire les valeurs des 2 capteurs de châssis) ; **4.** Cliquer sur « Démarrer *Blockly* ».
Pour programmer
1. Écrire le programme **2.** Sauvegarder le programme **3.** Exécuter le programme

Si le robot est relié à l'ordinateur par un câble et que le logiciel est fermé par erreur, l'ordinateur risque de ne plus détecter le robot : dans ce cas, débrancher et rebrancher le robot.

Situation déclenchante

Suite aux débats en classe de la séance précédente, le professeur diffuse la vidéo d'un projet de robotique d'étudiants de master de l'EPFL qui simule un réseau routier : https://www.thymio.org/creations-fr:thymio-reseau-routier

Les élèves décrivent ce qu'ils observent : des robots qui peuvent changer de couleur, certains jouant le rôle de voitures, d'autres celui de feux de circulation. Il y a aussi des marquages sur le bord des pistes sur lesquelles circulent les robots.

Découverte de Thymio

Le professeur distribue alors un exemplaire de ces robots à chaque groupe. Les élèves doivent découvrir, en autonomie, ce que peut faire Thymio.

Au bout de 5 minutes, les élèves ont découvert comment l'allumer ou l'éteindre : ils ont également remarqué que Thymio pouvait changer de couleur, et qu'il ne réagissait pas de la même façon selon la couleur qu'il arborait. Le professeur explique alors que chaque couleur correspond à un mode préprogrammé, et que les élèves doivent donc chercher à quoi correspond chaque mode.

Au bout de 10 minutes, et en l'absence de certains matériels, ils ne peuvent décrire que 4 des six modes, en leur trouvant à chaque fois un surnom, comme par exemple :

– Thymio « amical » (vert) : il suit l'obstacle placé devant lui. Si l'obstacle est trop près, il recule.

– Thymio « peureux » (rouge) : il fuit l'obstacle placé devant lui.

– Thymio « obéissant » (mauve) : il avance ou tourne quand on appuie sur les flèches de son capot. Appuyer plusieurs fois sur la même flèche accélère le mouvement.

– Thymio « explorateur » (jaune) : il avance seul, et esquive les obstacles placés devant lui.

Note pédagogique

Les élèves auront sans doute repéré deux autres modes (cyan et bleu), mais sans réellement pouvoir les interpréter.

• Nous utiliserons le Thymio cyan à la séance suivante.

• Le Thymio bleu est sensible au son : pour des raisons évidentes nous déconseillons son utilisation en classe. Cependant, il est possible de l'explorer moyennant quelques précautions : un seul élève pourra faire des tests devant la classe entière priée de rester complètement silencieuse.

Les élèves auront remarqué qu'aucun des quatre modes étudiés jusqu'ici ne reproduit ce qu'ils ont vu dans la vidéo. Certains vont peut-être émettre l'hypothèse qu'il faudrait des pistes sur lesquelles faire évoluer leurs robots, et c'est effectivement une réponse, bien que partielle.

Dans l'immédiat, les élèves peuvent désormais définir ce qu'est un robot. Le professeur leur demande : *à votre avis, pourquoi met-on Thymio dans la catégorie des robots ?* Les idées sont notées au tableau, sans pour l'instant donner la réponse.

Un premier programme en *Blockly*

Le professeur précise que les Thymio de la vidéo ont été reprogrammés pour les besoins de la simulation. Et les élèves vont eux-mêmes reproduire une simulation routière, qui nécessitera de modifier les comportements par défaut de leur(s) Thymio, en utilisant un langage de programmation dédié, « *Aseba Studio* ».

Le professeur démontre le fonctionnement de l'interface *Blockly* pour *Aseba Studio* en créant en direct le programme suivant :

Il demande alors aux élèves d'essayer de deviner ce que signifie ce programme, et ce que Thymio fera si on lui demande de l'appliquer. Certains élèves proposent que Thymio prendra une couleur verte s'il détecte quelque chose devant lui. Le professeur exécute alors le programme pour vérifier cette hypothèse.

La démonstration permet d'aborder dès maintenant deux notions :

1. Thymio est un robot, car il est doté de capteurs, d'actionneurs, et d'un ordinateur. Les élèves peuvent essayer de les repérer :

• Capteurs (représentés dans le langage *Blockly* par des déclencheurs bordeaux ou des valeurs de capteurs bleues) comme les 9 détecteurs à infrarouge (carrés noirs dotés d'une DEL pour signaler leur activité) sur le pourtour et le châssis, mais aussi le *touchpad*, (les flèches et le bouton central) ou encore, bien qu'invisibles : le microphone et l'accéléromètre.

• Actionneurs (représentés dans le langage *Blockly* par des blocs beiges) : plusieurs DEL d'ambiance, pour le capot et pour le châssis, deux moteurs indépendants pour les roues, un haut-parleur.

• Ordinateur : associe des évènements liés aux détections par les capteurs et des actions liées aux actionneurs, via un programme. Sans démonter Thymio, ils ne peuvent pas observer le microprocesseur qui l'applique.

2. Thymio obéit strictement au programme fourni : ici, on ne lui a pas dit quoi faire quand il ne détectait plus rien face à lui. Une fois teinté en vert, il garde donc cette couleur quoi qu'il arrive. Il faut donc souvent

initialiser ses programmes, et inclure des portes de sortie, ou contrordres, pour revenir à l'état initial. Les élèves dictent alors comment compléter ce programme pour revenir à l'état initial :

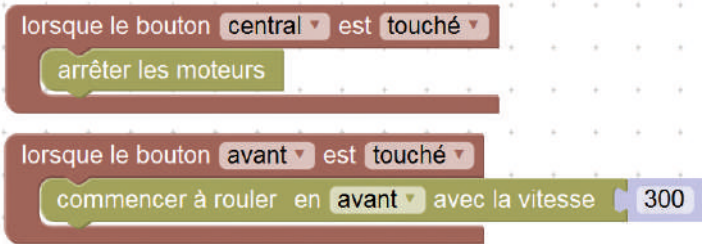
Le professeur démontre ainsi deux états fondamentaux des capteurs de Thymio : actif (« détecte quelque chose ») et inactif (« détecte rien »). Il passe sous silence pour l'instant les autres valeurs disponibles (« détecte du noir » ou « détecte du blanc ») qui serviront implicitement plus loin.
Le professeur précise enfin que *Blockly* permet une programmation évènementielle et parallèle : chaque action est déclenchée par certains évènements. Des actions différentes peuvent démarrer simultanément si les évènements qui les déclenchent (typiquement sur deux capteurs distincts) sont eux aussi simultanés.

> **Note pédagogique**
> En arrière-plan, il est possible de voir *Aseba Studio* écrire en temps réel l'équivalent textuel du programme graphique réalisé. Cela permet d'aborder le fait qu'il existe de nombreux langages de programmation, certains graphiques, certains textuels, mais que tous doivent être interprétés par la machine.

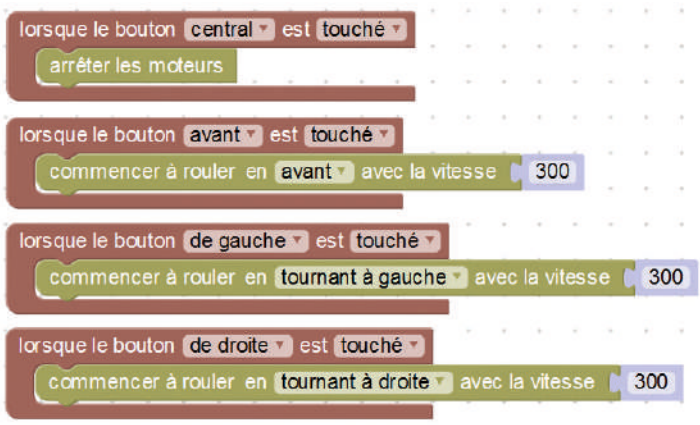
Quelques défis pour prendre *Blockly* en main

Les élèves vont se familiariser avec *Blockly* en relevant plusieurs défis : il va s'agir de reproduire partiellement les comportements observés en mode mauve (obéissant) et en mode jaune (explorateur). Nous supposons que chaque défi permet d'améliorer le programme petit à petit, en ajoutant progressivement de nouveaux éléments, plutôt que de recommencer de zéro à chaque fois.

Défi 1 : Thymio avance grâce à la flèche du haut et s'arrête grâce au bouton central
Voici une solution possible. On peut ajouter de nouvelles instructions pour complexifier le programme et ajouter des fonctionnalités du robot.

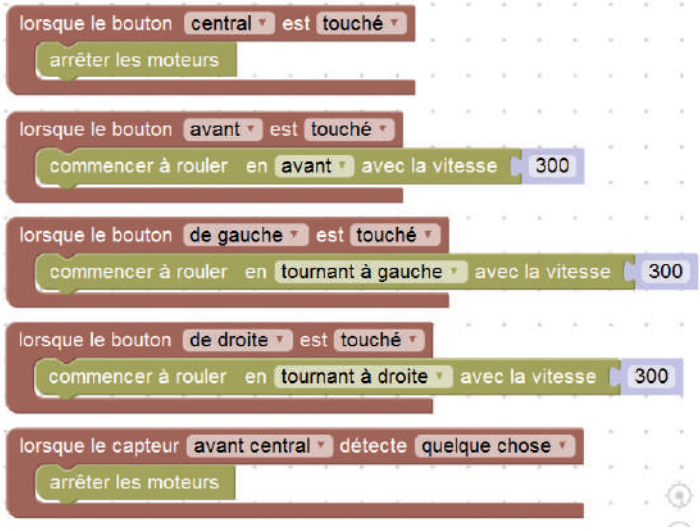


Défi 2 : Thymio tourne grâce aux flèches



Défi 3 : Thymio s'arrête si un obstacle est devant lui

À ce stade, nous arrêtons l'imitation du mode mauve pour incorporer une détection d'obstacles.



Conclusion

Les élèves élaborent une conclusion commune, par exemple :

– Un robot est une machine qui peut interagir avec son environnement.

– Un robot possède un ordinateur, des capteurs et des actionneurs, tous connectés entre eux.

– Pour donner des instructions à une machine, on utilise un langage de programmation, compréhensible à la fois par la machine et par l'être humain.

– Certaines instructions ne s'exécutent qu'au déclenchement d'un évènement : on parle de programmation évènementielle.

– Blockly *est un langage de programmation que Thymio peut comprendre.*

Ils peuvent y ajouter un bref descriptif des modes préprogrammés de Thymio qu'ils auront étudiés.

Séance 3 – Programmer un évitement d'obstacle

Discipline dominante	Technologie
Résumé	Les élèves apprennent à manipuler des opérateurs logiques, et programment Thymio pour lui permettre d'éviter un obstacle.
Notions *(cf. scénario conceptuel, page 302)*	Machines : • En combinant plusieurs instructions simples on peut effectuer une tâche complexe. Langages : • Un programme est un algorithme exprimé dans un langage de programmation. Algorithmes : • Un "algorithme" est une méthode permettant de résoudre un problème. • Un test permet de choisir quelle action effectuer si une condition est vérifiée ou non. • Une condition est une expression qui est soit vraie, soit fausse. • On peut utiliser des opérateurs logiques comme ET, OU, NON pour fabriquer des expressions logiques.
Matériel	Par groupe : • Un robot Thymio • Un ordinateur avec *Aseba Studio* installé • Fiche 2, page 314

Situation déclenchante

Lors de la séance précédente, les élèves ont découvert Thymio, un robot qu'ils vont programmer pour reproduire une scène de circulation automobile. Ils ont commencé à programmer des comportements simples avec *Blockly*: le défi 3 rappelle une version très simplifiée du mode mauve.

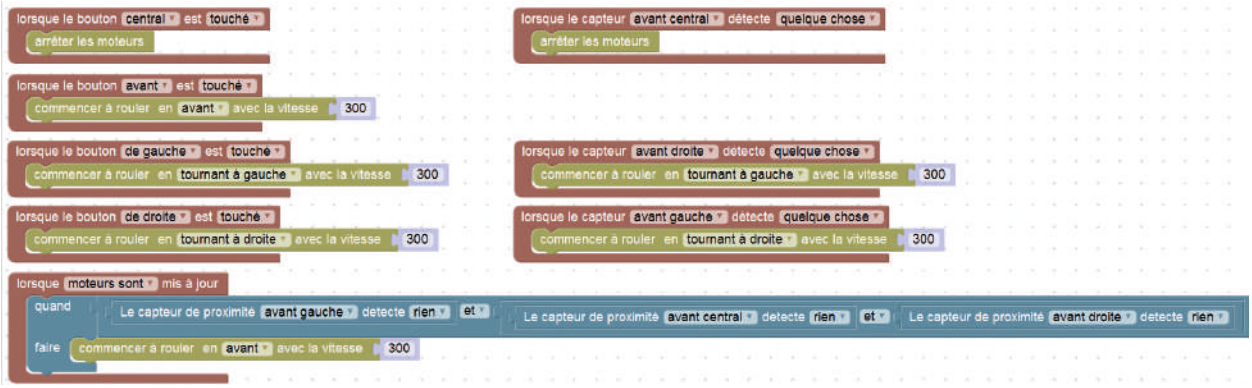
Défi 4 : Thymio contourne les obstacles

Le professeur propose de reprendre leur dernier programme, et d'y incorporer des éléments caractéristiques du mode jaune, l'explorateur. Thymio doit contourner les obstacles situés à sa droite ou à sa gauche, en tournant dans la direction opposée. La fonctionnalité d'arrêt face à un obstacle programmée au défi 3 est maintenue.

Pour relever le défi 4, il faut prévoir une porte de sortie (bloc inférieur) : si Thymio ne détecte explicitement rien devant lui, alors il peut aller tout droit (sinon il ne redémarrerait jamais). Dans ce cas, nous impliquons ici trois capteurs (avant-gauche, centre, avant-droite), par souci de lisibilité : un programme similaire mais plus robuste impliquerait les cinq capteurs avant.

Note pédagogique

L'emploi de l'instruction « quand moteurs sont mis à jour » est nécessaire pour pouvoir imbriquer plusieurs conditions simultanément (*cf.* bloc bleu, ci-dessus).



Que les élèves découvrent ou non la programmation par blocs, il peut être bénéfique d'utiliser les exercices de la Fiche 2 (page 314) pour que les élèves s'approprient ou consolident les opérateurs logiques.

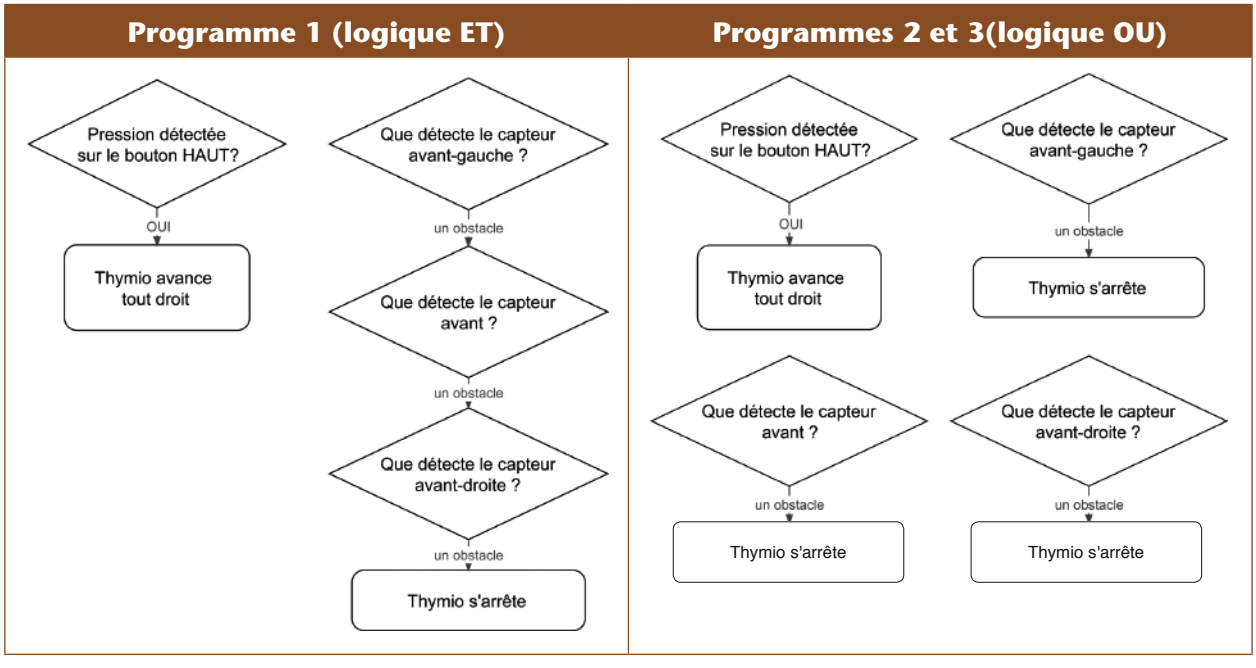
• **Programme 1** : Thymio avance tout droit lorsqu'on appuie sur la flèche du haut ; SI les trois capteurs centraux détectent un obstacle simultanément, ALORS Thymio s'arrête.

• **Programme 2** : Thymio avance tout droit lorsqu'on appuie sur la flèche du haut ; SI l'un des trois capteurs centraux détecte un obstacle, ALORS Thymio s'arrête.

• **Programme 3** : ce programme donne un résultat identique au programme 2 !

Le programme 1 illustre une logique « ET » : il faut que les trois capteurs détectent un obstacle ; si un seul capteur est dans ce cas, alors Thymio continue tout droit. Les programmes 2 et 3 illustrent une logique « OU », une seule détection d'obstacle suffit à stopper Thymio (et 2 détections simultanées ou 3 détections simultanées l'arrêtent également).

Le professeur peut vouloir à ce moment introduire les logigrammes que nous utiliserons souvent dans le courant de cette séquence. Il construit avec la classe les logigrammes des deux algorithmes de la Fiche 2 :



Conclusion

Les élèves élaborent une conclusion commune, comme celle-ci :

– Un « algorithme » est une méthode permettant de résoudre un problème.

– En combinant plusieurs instructions simples on peut effectuer une tâche complexe.

– Un test permet de choisir quelle action effectuer si une condition est vérifiée ou non.

– On peut utiliser des opérateurs logiques comme ET, OU, NON pour fabriquer des expressions logiques.

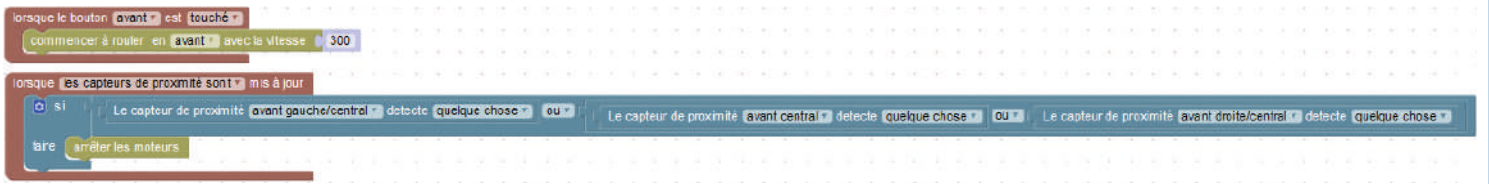
– Un programme est un algorithme exprimé dans un langage de programmation.

Programme 1

Hypothèses

Observations

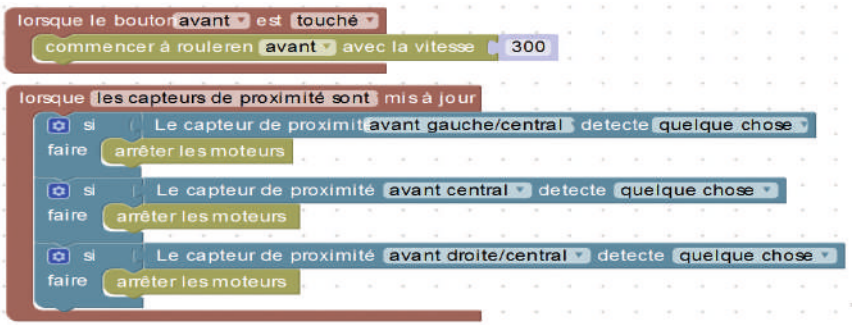
Programme 2



Hypothèses

Observations

Programme 3



Hypothèses

Observations

Consigne : Dans les trois cas suivants, écris d'abord tes hypothèses, en fonction de ce que tu devines du comportement probable de Thymio. Ensuite, reproduis ces programmes sur ton propre Thymio et écris alors tes observations. Qu'en penses-tu ?

FICHE 2
Thymio et les opérateurs logiques de *Blockly*

Séance 4 – Programmer un suiveur de ligne

Discipline dominante	Technologie
Résumé	Les élèves programment eux-mêmes un comportement qui imite le mode « suiveur de ligne » de Thymio. Ils décrivent tout d'abord l'algorithme de ce mode, puis l'implémentent avec *Blockly*.
Notions *(cf. scénario conceptuel, page 302)*	Langages : • Certaines instructions s'exécutent simultanément à d'autres : on parle de programmation parallèle. • Un bug est une erreur dans un programme. • Un tout petit bug peut parfois avoir des conséquences énormes. Algorithmes : • Un algorithme peut contenir des instructions élémentaires, des tests, des boucles. • Une boucle permet de répéter plusieurs fois la même action. • Certaines boucles, dites "infinies", ne s'arrêtent jamais.
Matériel	Par groupe : • Une piste pré-imprimée* sur feuille A3 • Un robot Thymio • Un ordinateur avec Aseba Studio installé

Situation déclenchante

Forts des acquis de la séance précédente, les élèves vont pouvoir désormais reproduire un autre mode : le mode cyan, qui fait partie des modes qu'ils n'ont probablement pas su sonder auparavant. Le professeur distribue alors des pistes préparées à l'avance qui vont aider les élèves à caractériser le mode cyan « suiveur de piste ». Ces pistes devraient leur rappeler la simulation de l'EPFL démontrée en vidéo (*cf.* séance 2, page 306), qui reposait sur un réseau routier symbolisé par des pistes de couleur sombre.

Observer le Thymio cyan

La mission des élèves est simple : *mettez votre Thymio en mode cyan et observez ce qu'il fait sur cette piste.* Les élèves comprennent vite que si Thymio tournait en rond, c'est qu'il cherchait bel et bien une piste à suivre. Ensuite, une fois sur une piste, Thymio la parcourt en redressant toujours sa trajectoire pour ne pas la perdre dans les virages ou les croisements.

Défi 5 : imiter le Thymio suiveur de piste

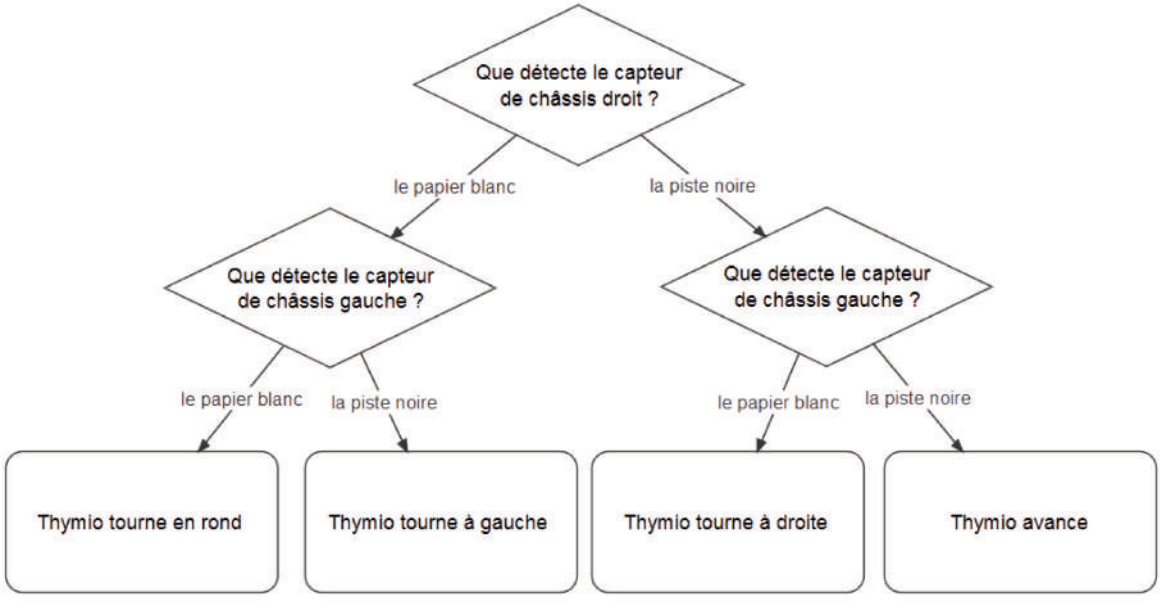
Maintenant, les élèves doivent programmer une imitation simplifiée de ce mode avec *Blockly*.

* Il existe de nombreuses pistes pour Thymio disponibles sur Internet. Celle-ci convient parfaitement : https://aseba.wdfiles.com/local--files/en:thymioaccessory/piste%208.pdf

Note pédagogique

Il est trop ambitieux de vouloir reproduire à l'identique les subtilités de ce mode. Le but est ici d'acquérir une méthode pour en faire une imitation acceptable. Pour être précis, tous les modes préprogrammés ont été rédigés en programmation textuelle (ce que permet *Aseba Studio*), qui autorise beaucoup plus de fonctionnalités et de finesse que la programmation graphique. Les compétences de programmation textuelle relèvent davantage du lycée.

Dans un premier temps, les élèves réfléchissent à l'algorithme à implémenter. Pour cela, ils peuvent indifféremment décrire leur méthode en français, en dessiner un logigramme (si le professeur a introduit cet outil), etc. Ils ont évidemment le droit d'observer Thymio cyan autant que nécessaire pour bien comprendre quels capteurs et actionneurs sont en jeu, et dans quelles conditions.
Une mise en commun des propositions des groupes permet de s'accorder sur l'algorithme à implémenter :



Note scientifique

Ce logigramme, tout à fait représentatif du fonctionnement de *Blockly* ou de VPL, passe sous silence un ingrédient implicite : ces conditions sont testées en permanence, indéfiniment. On parle de « boucle infinie ».

Chaque groupe doit désormais implémenter cet algorithme sous *Blockly*, pour que Thymio le comprenne. Lorsque Thymio ne trouve pas de piste, il tourne en rond : la direction est tout à fait arbitraire (dans la solution que nous proposons, il tourne à gauche : dans l'algorithme ci-dessus la formulation est volontairement floue). Voici une solution possible :

Note pédagogique

Comme illustré ici, on peut associer plusieurs actions à un seul et même évènement. C'est facultatif pour l'instant, mais cela deviendra vite obligatoire, dès la séance suivante. Il n'est pas indispensable de colorier Thymio de façon différente à chaque étape, mais c'est une bonne habitude de programmation, qui facilitera le débogage par la suite : pourquoi est-il bloqué dans telle couleur ? pourquoi telle couleur n'apparaît-elle jamais ?, etc.

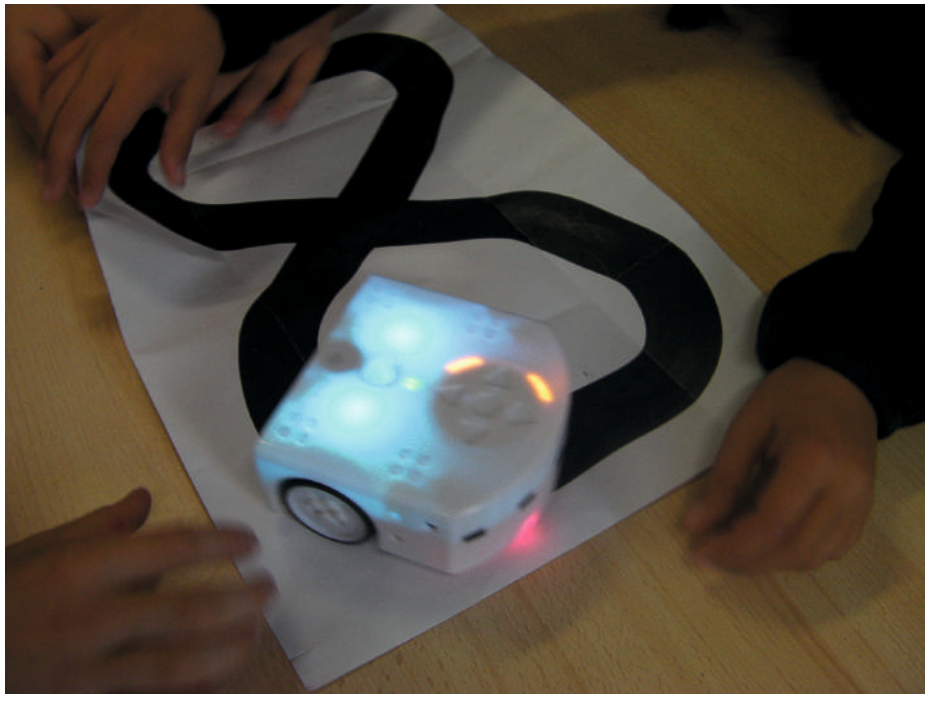
Notes scientifiques

• Les élèves peuvent se rendre compte que l'absence d'événement est aussi, d'un point de vue programmatique, un événement à part entière. En effet, les capteurs de Thymio ne sont pas capables stricto sensu de détecter un sol noir : précisément, dans ce cas-là ils signalent qu'ils ne détectent pas de sol blanc. La programmation *Blockly* permet toutefois de passer outre ces subtilités : il existe quatre détections possibles, « quelque chose », « du blanc », « du noir », ou « rien ».

• L'aspect le plus délicat à peaufiner ici est le suivi des virages très serrés. Dans ce cas-là, il faut certes tourner pour revenir sur la piste, mais surtout il ne faut pas non plus se déporter (au risque de quitter la piste) : il faut donc faire avancer une roue et simultanément faire reculer l'autre, pour que Thymio pivote sur place ! C'est la distinction que nous faisons ci-dessus en précisant si Thymio « roule en tournant à gauche » ou « tourne dans le sens horaire ».

Mise en commun

Chaque groupe présente sa solution à la classe, qui identifie les meilleures idées pour élaborer le meilleur programme possible. Comme souvent, en informatique, il n'y a pas une solution unique à ce problème. Tant que la solution choisie par la classe répond au cahier des charges (« imiter le comportement de Thymio cyan »), c'est une bonne réponse.



Thymio sur piste

Conclusion

Les élèves élaborent une conclusion commune, comme celle-ci :

– Un bug est une erreur dans un programme.

– Introduire des repères dans le comportement du robot (émission d'un son, changement de couleur…) permet de faciliter la correction des bugs.

– Quand un capteur détecte quelque chose, que ce soit blanc ou noir, on dit qu'il y a un événement.

Séance 5 – Programmer un suiveur de ligne amélioré

Discipline dominante	Technologie
Résumé	Pour que Thymio puisse recueillir des informations sur son chemin via des « codes-barres », les élèves modifient leur suiveur de ligne : il n'utilise plus désormais qu'un seul capteur de châssis au lieu de deux.
Notions *(cf. scénario conceptuel, page 302)*	Machines : • Une variable est un nom que l'on donne à une zone de mémoire. Elle permet de stocker une valeur et de la réutiliser plus tard, ou de la modifier.
Matériel	Par groupe : • Un robot Thymio • Un ordinateur avec *Aseba Studio* installé • La Fiche 3 Pour la classe : • Un vidéoprojecteur • Une piste réalisée avec les tronçons des Fiches 4 à 6

Préparation

Le professeur prépare à l'avance une piste avec les tronçons des Fiches 4 à 6 : Thymio devra la parcourir avec le bord noir à sa droite et le bord blanc à sa gauche. La piste doit inclure des virages dans les deux sens pour bien tester la robustesse des programmes proposés.

Situation déclenchante

Les élèves rappellent ce qui a été vu à la séance précédente : Thymio peut suivre des lignes pour aller où on veut l'emmener. Le professeur demande alors ce qui peut se passer si Thymio « suiveur de ligne » rencontre une intersection.

Les élèves débattent rapidement et plusieurs hypothèses font surface :

– selon son orientation et sa vitesse, il peut aller parfois à droite ou parfois à gauche ;

– son programme pourrait lui dire quoi faire : aller toujours à droite, ou toujours à gauche, ou choisir aléatoirement.

Le professeur catégorise ces propositions selon deux axes : dans le premier cas, cela dépend de l'environnement, dans le second cas, cela dépend de la programmation du robot. C'est ce second cas qui va être étudié.

Pour cela, les élèves doivent trouver comment les étudiants de Master ont pu donner des informations complémentaires à leurs robots. Ils visionnent à nouveau la vidéo de l'EPFL, pour trouver des éléments de réponse : https://www.thymio.org/creations-fr:thymio-reseau-routier

Les élèves peuvent repérer trois choses :

– Les pistes ne sont pas noires comme la leur…

– Il y a des codes-barres sur le côté des pistes…

– Les Thymio « feu de circulation » sur les ronds-points donnent des informations d'une manière ou d'une autre aux autres Thymio… (pourquoi certains s'arrêteraient-ils au feu alors que d'autres pourraient aller tout droit ou tourner ?)

Le professeur leur confirme qu'il y a effectivement un dialogue Wifi qui s'engage entre les différents Thymio, mais qu'il est possible, en classe, de reproduire des comportements complexes en utilisant simplement un système de codes-barres. Mais comment faire, alors que les deux capteurs de châssis du Thymio sont concentrés sur le suivi de piste ?

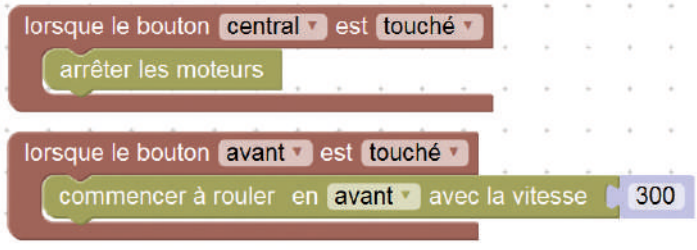
Les élèves arrivent à la conclusion qu'il faut dédier un seul capteur au suivi de la piste (disons le gauche) et libérer l'autre (ici le droit) pour la lecture du code-barres. Il faut donc apprendre à :

– suivre une piste avec un seul capteur ;
– lire un code-barres ;
– retenir quel mode utiliser en fonction de l'information du code-barres.

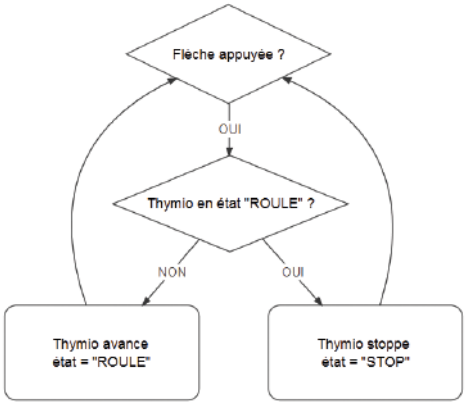
> **Note pédagogique :**
> La programmation d'un vrai lecteur de codes-barres pour Thymio est :
> – impossible avec VPL ;
> – difficile avec *Blockly* ;
> – conseillée en mode textuel (ce qui permet de régler la fréquence de lecture des codes-barres en fonction de la vitesse de déplacement des Thymio).
> Nous ne programmons donc pas, dans ce qui suit, un vrai lecteur de code-barres : nous nous contentons de « bornes » simples, noires, qui joueront le rôle de « bip ».

Thymio dans tous ses états (classe entière)

Le professeur annonce que Thymio a effectivement une petite mémoire interne qu'il est possible d'utiliser pour mémoriser des sous-programmes différents à utiliser dans des conditions différentes. Il reprend alors le programme du premier exercice effectué par les élèves :



Il leur demande de trouver l'algorithme qui permettrait d'avoir le même comportement, mais en appuyant exclusivement sur la flèche « haut ». Ils élaborent un algorithme comme celui-ci, que le professeur traduit aussitôt sous *Blockly* :



Les élèves appliquent ce programme à leur propre Thymio, pour vérifier qu'il répond bien à la consigne.

Note scientifique :

Dans *Aseba/Blockly*, les seules variables possibles sont des tableaux, ce qui complique légèrement l'écriture mais ne change pas l'algorithme.

Exploration : la sensibilité de Thymio (par groupes)

Les élèves peuvent maintenant programmer le suiveur de ligne amélioré. Pour cela, ils doivent tout d'abord rendre leur Thymio borgne, en masquant le capteur droit (qui sera réservé à la lecture d'informations) avec de la patafix® ou du scotch opaque. Puis le professeur leur demande comment faire pour n'utiliser qu'un capteur, mais néanmoins savoir précisément où Thymio se trouve par rapport à la piste.

Les élèves auront probablement remarqué que les pistes de la vidéo n'étaient pas tout à fait noires, à bord francs : elles sont effectivement dégradées, allant du noir sur le bord droit au gris clair sur le bord gauche. En effet, les capteurs de Thymio ne sont pas « binaires » : ils peuvent tout à fait détecter des nuances de gris, et pas seulement du noir ou du blanc.

Note scientifique

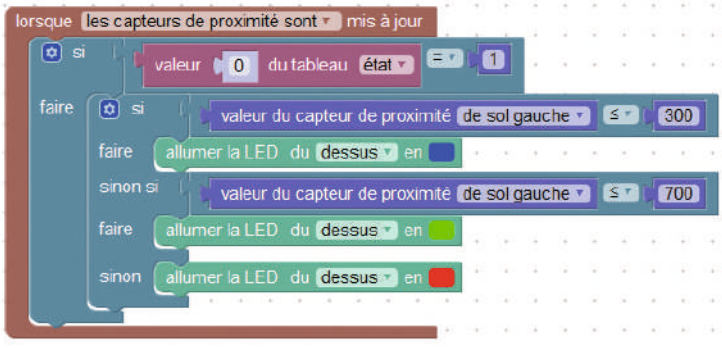
Les capteurs de Thymio sont sensibles à l'infrarouge : nos notions de « blanc » et de « noir » définies visuellement ne sont qu'extrapolables à ce domaine ; le choix du matériau (encre, papier) peut donc poser problème. De plus, seul le mode textuel d'*Aseba Studio* permet de profiter pleinement de toute la gamme des mesures de ces capteurs.

Le premier défi des élèves consiste à leur faire étalonner leur Thymio. Dans ce cas précis, le professeur distribue la Fiche 3 et demande aux élèves : *Faites avancer Thymio tout droit sur cette feuille, et faites-le changer de couleur en fonction de la nuance de gris détectée.*

Note pédagogique

Dans le cadre de ce calibrage, il n'est évidemment pas obligatoire de faire avancer le robot, et encore moins d'utiliser des variables, mais c'est une bonne occasion de s'approprier l'outil vu à l'instant. De plus, cela permet d'écrire des programmes plus propres, avec des comportements distincts « repos » et « actif » : Thymio ne commence à agir que lorsqu'on le déclenche.

Les élèves peuvent compléter le programme de leur Thymio avec ces instructions :

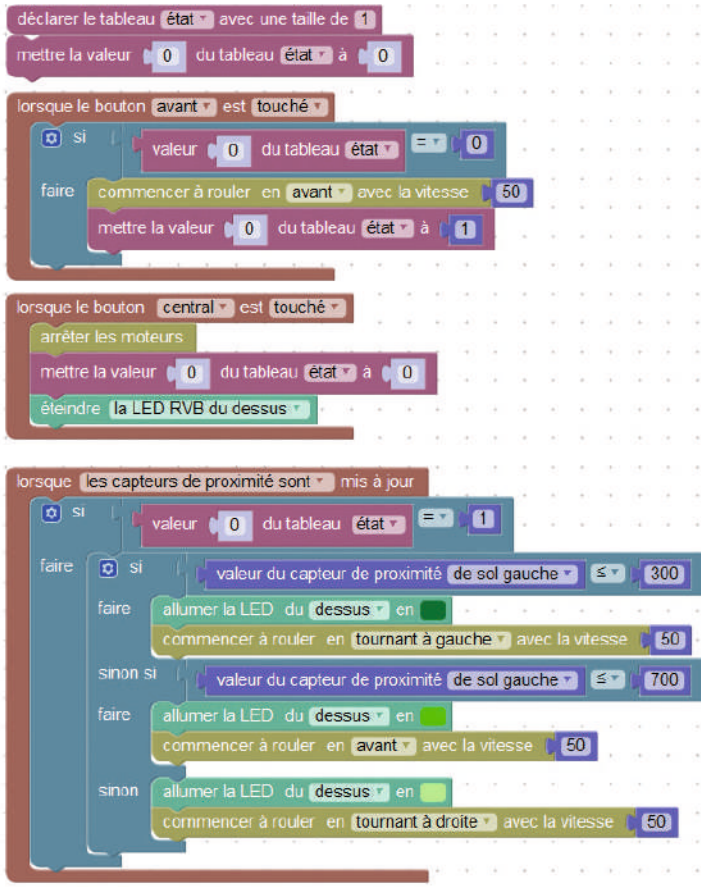


Note scientifique

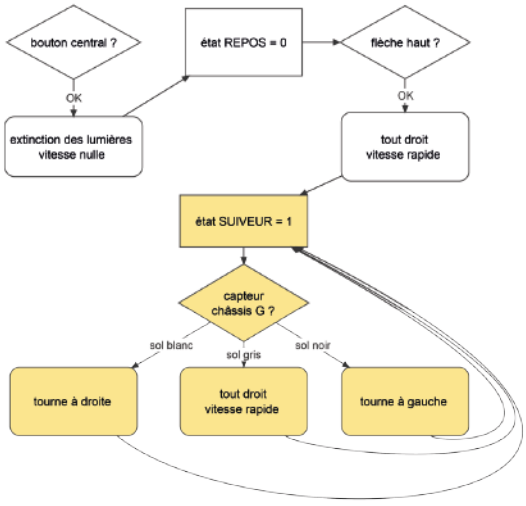
Il ne s'agit pas d'un réel calibrage : au contraire, les élèves vont devoir adapter leur programme à la sensibilité du Thymio qu'ils ont entre les mains. Il sera donc idéal de leur laisser le même robot à chaque séance pour ne pas avoir de mauvaise surprise lors des séances suivantes, à moins de recommencer cette étape à chaque reprise. Il faut donc sauvegarder ce programme pour les « calibrages » futurs.

Conclusion : suiveur de ligne amélioré (par groupes)

Les élèves peuvent enfin repartir de ce programme pour améliorer leur suiveur de ligne. Ils testeront leur Thymio sur la piste commune.
Le programme complet devient :



On voit bien ici que l'état 0 correspond à l'état « repos » : il faut appuyer sur la flèche « haut » pour que le suiveur de ligne démarre (état 1). Dans tous les cas, le bouton central remet Thymio au « repos » (état 0). Encore une fois, il n'y a pas de solution unique : tant que le Thymio « borgne » (avec le capteur droit masqué) réussit à suivre la piste de test, quelle que soit la position initiale où on le place, le défi est réussi.
Ici, les couleurs choisies (trois nuances de vert) préfigurent la mise en situation de la séance suivante : le suiveur ci-dessus sera la « tortue » de la fable de la Fontaine.
Les élèves reproduisent enfin dans leur cahier de projet le logigramme du suiveur de ligne :

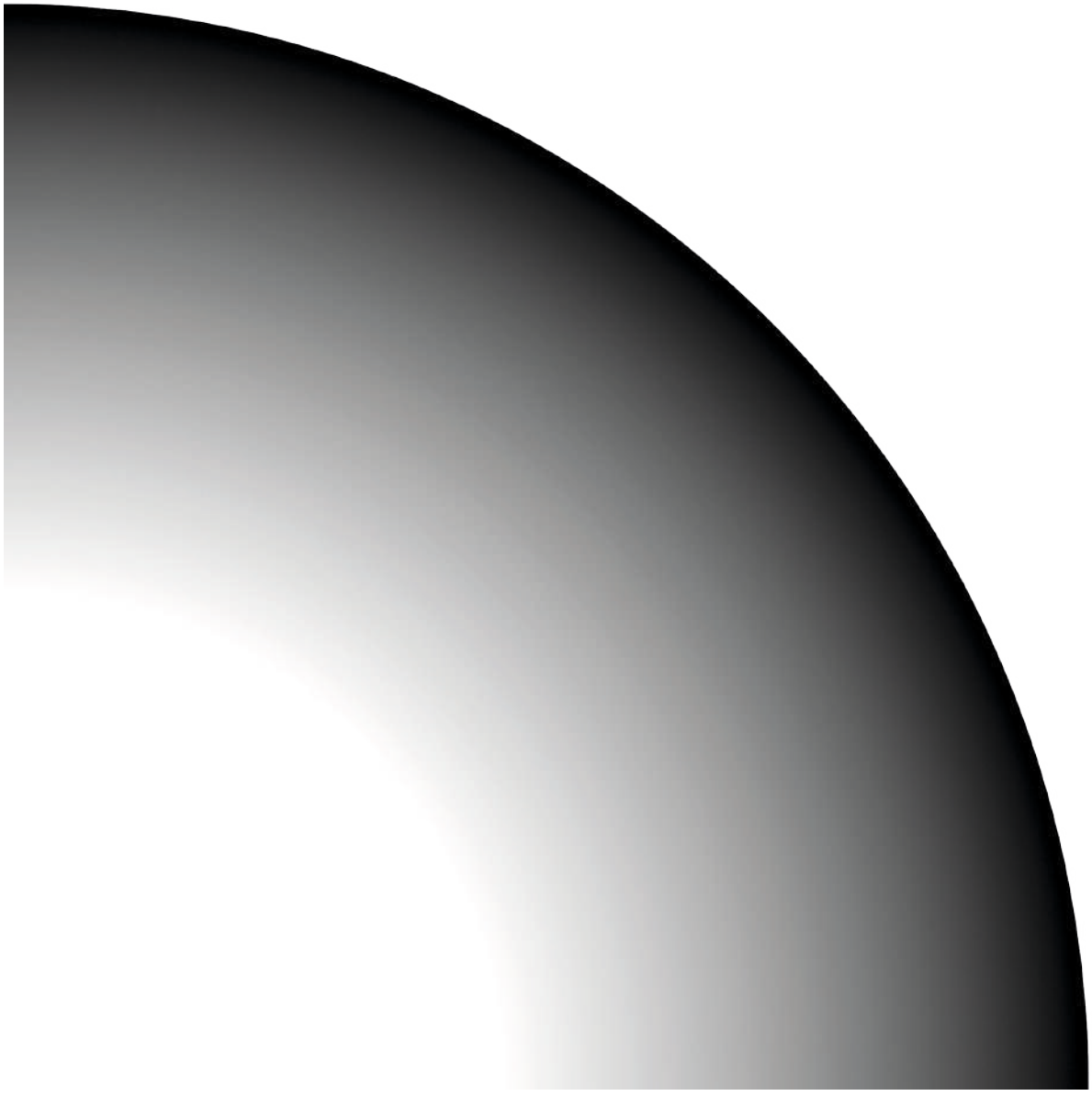


FICHE 3
Etalonnage : 50 nuances de gris pour Thymio

FICHE 4
Piste dégradée pour Thymio : tronçon rectiligne

FICHE 5
Piste dégradée pour Thymio : virage à droite

FICHE 6
Piste dégradée pour Thymio : virage à gauche

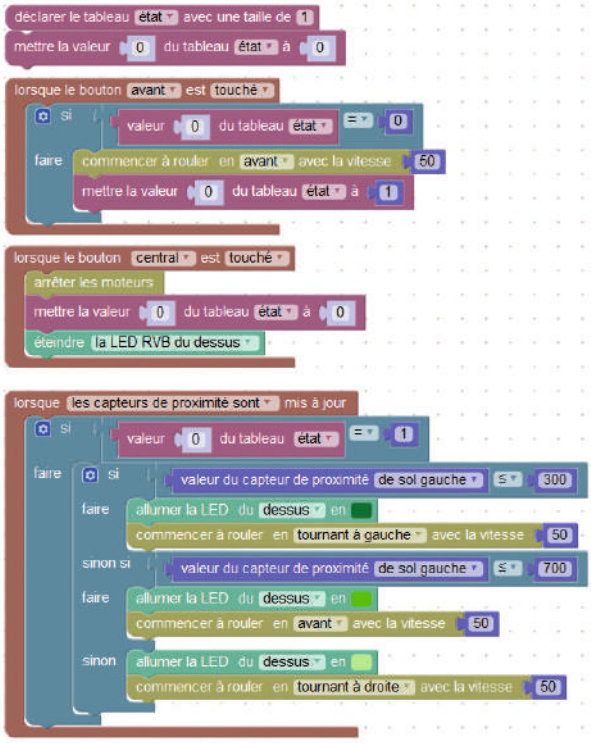


Séance 6 – Programmer le lièvre et la tortue

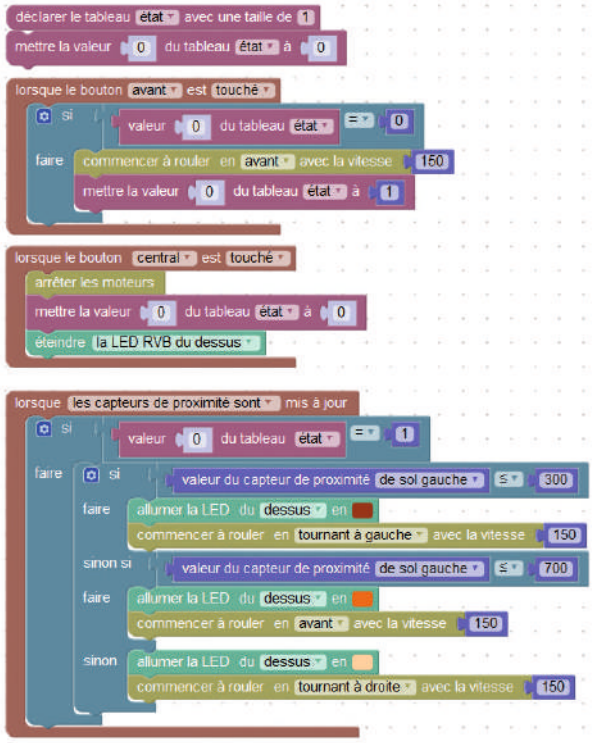
Discipline dominante	Technologie
Résumé	La première simulation de circulation routière des élèves reprend le thème d'une fable de La Fontaine. Il n'y a pas de véritable enjeu tant que le lièvre et la tortue sont sur des pistes séparées. En revanche, lorsqu'ils partagent la même piste, les élèves doivent prévenir les accidents.
Notions *(cf. scénario conceptuel, page 302)*	Idem séances précédentes
Matériel	Pour la classe • Un vidéoprojecteur • Deux longues pistes parallèles disjointes, constituées de plusieurs tronçons rectilignes (Fiche 4) • Un robot Thymio pour jouer la tortue Par groupe • Un robot Thymio pour jouer le lièvre • Un ordinateur avec *Aseba Studio* installé

Situation déclenchante

Le professeur propose de revivre la fable *Le Lièvre et la Tortue* de Jean de La Fontaine : il y a deux pistes dégradées, parallèles, et il faut programmer deux suiveurs de piste, avec des vitesses de croisière différentes. Il est préférable de colorer le Thymio suiveur « lent » et le Thymio suiveur « rapide » de façon différente pour bien les distinguer. Le professeur gèrera le suiveur « lent », tandis que chaque groupe programmera son suiveur « rapide ». Les élèves arrivent rapidement à deux programmes similaires :

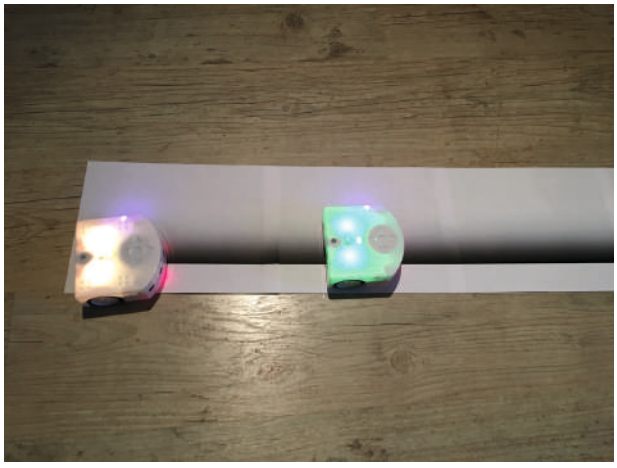
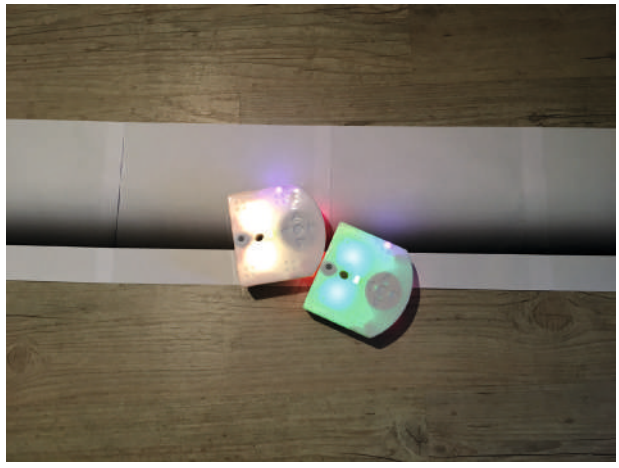


Programme de la tortue



Programme du lièvre

Les groupes organisent une course sur les deux pistes parallèles, et, sauf erreurs de programmation (bugs) ou aléas de l'environnement (pli d'un papier, défaut d'impression de la piste, etc.), le lièvre arrive toujours le premier.
Maintenant, le professeur retire une des deux pistes, et annonce que, comme dans la fable, le lièvre laisse de l'avance à la tortue avant de partir. Que va-t-il se passer ?

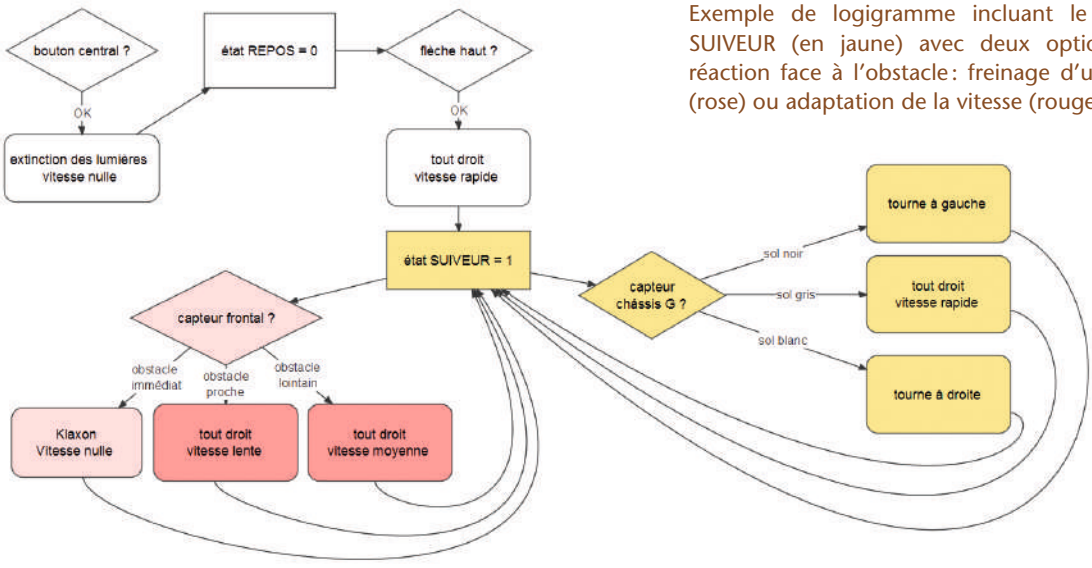



À gauche : situation initiale, la tortue (en vert) a un peu d'avance sur la piste unique, le lièvre (en orange) la suit de peu. À droite : le lièvre a rattrapé la tortue et cause un accident en la poussant sur le bas-côté.

Evidemment, le lièvre rattrape la tortue et cause un accident : selon les cas, il la pousse jusqu'à la ligne d'arrivée, la fait dévier vers le fossé, dévie lui aussi, etc.
Le défi proposé aux élèves permet de gérer ce problème.

Défi : le lièvre rattrape la tortue sans causer d'accident (par groupes)

Le professeur demande alors : *dans une situation automobile, comment peut-on gérer ce problème ?*
Les élèves proposent en particulier de freiner, ou d'adapter sa vitesse. Le professeur propose alors à chaque groupe d'appliquer chacune des deux méthodes, en commençant par le freinage.
A chaque fois, les élèves sont encouragés à mettre au propre leur algorithme avant de se lancer dans la programmation.



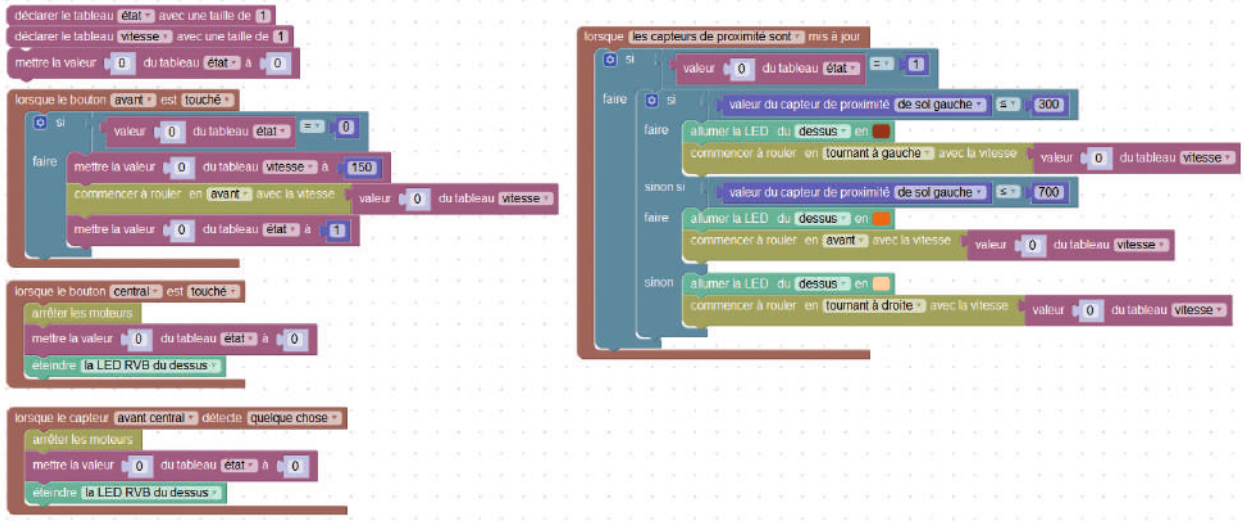
Exemple de logigramme incluant le mode SUIVEUR (en jaune) avec deux options de réaction face à l'obstacle : freinage d'urgence (rose) ou adaptation de la vitesse (rouge)

Note pédagogique

Nous conseillons à tous les groupes de commencer par la programmation du freinage d'urgence même s'ils ont envisagé des solutions plus fines. En effet, le freinage permet une première approche de l'utilisation des capteurs frontaux, qui sera approfondie lors de la programmation de la seconde solution.

Solution 1 : freinage d'urgence

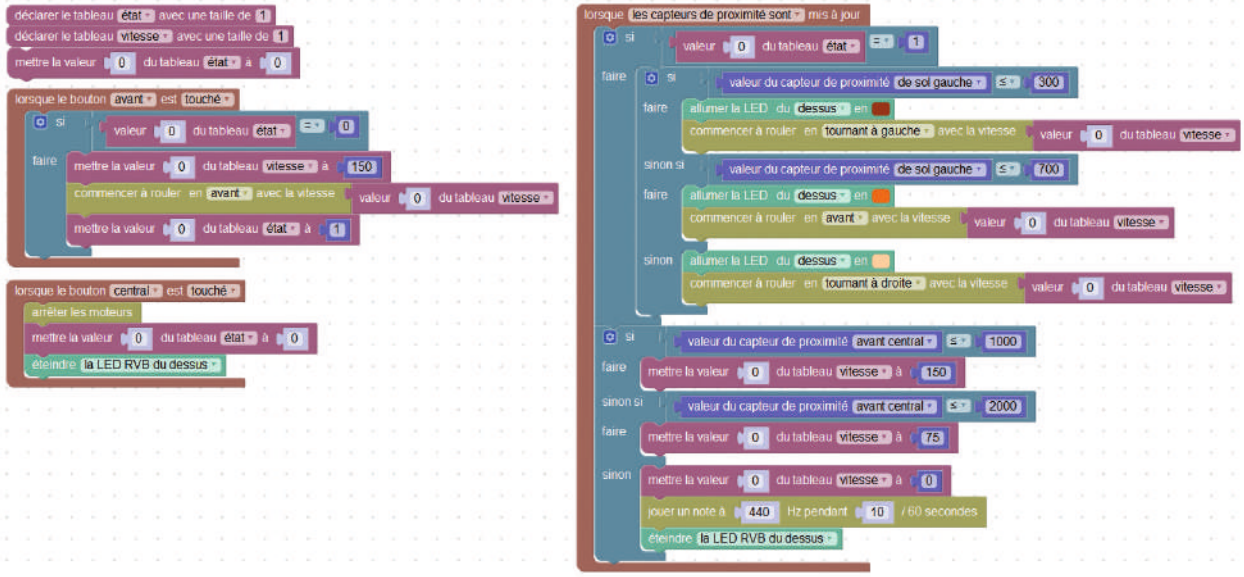
Une première solution consiste à piler net devant l'obstacle. Si c'est déconseillé en ASSR, c'est la solution la plus rapide à programmer.



Seul le bloc en bas à gauche est nouveau.

Solution 2 : suivre la tortue

Une seconde solution est d'adapter sa vitesse à celle de la tortue, en gardant une distance de sécurité. Là encore, il faut faire plusieurs essais pour bien jauger la sensibilité du capteur avant. Il est tout à fait possible (et encouragé) d'utiliser l'écran d'*Aseba Studio* qui reste inactif en arrière-plan à bon escient : si, avant de « Démarrer *Blockly* », on déplie d'office les onglets prox.horizontal (capteurs 5+2) et prox.



ground.delta (2 capteurs de châssis) dans *Aseba Studio*, on peut lire en temps réel les valeurs retournées par les capteurs.
Dans la correction proposée ci-dessus, le lièvre ralentit lorsqu'il se rapproche trop de la tortue, jusqu'à piler et klaxonner s'il s'en approche trop.

> **Note pédagogique**
> Encore une fois, il n'est pas demandé de reproduire le comportement du Thymio « amical » (mode préprogrammé vert) car ce mode est plus subtil qu'il n'y paraît : Thymio est capable d'adapter sa vitesse en fonction de la position, mais aussi de la vitesse de l'obstacle. Ici, les élèves doivent comprendre le principe de la programmation plus qu'aboutir à des programmes parfaits.

Conclusion

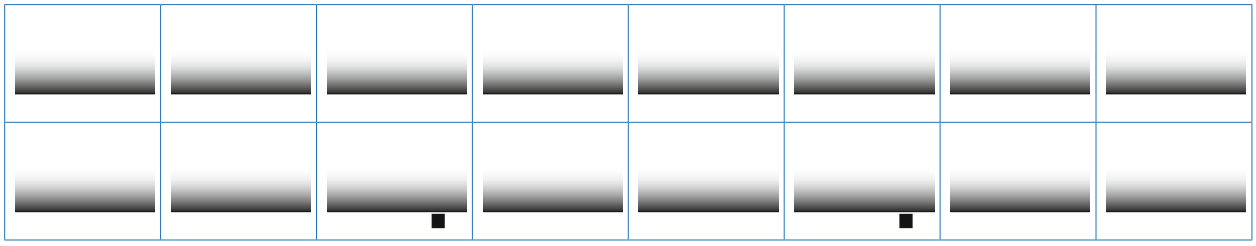
Avec ces nouvelles conditions, l'accident est évité, mais le lièvre est furieux. *Que se passe-t-il dans la vraie vie, sur la route, dans des situations similaires ?*
Les élèves répondent vite : le véhicule rapide cherche à doubler le véhicule lent. Ce sera l'objet de la séance suivante.
Pour l'instant, les élèves reproduisent le logigramme du lièvre dans leur carnet de projet.

Séance 7 – Autoriser le lièvre à doubler la tortue

Discipline dominante	Technologie
Résumé	Pour laisser une chance au lièvre, tout en évitant les accidents, les élèves doivent programmer le changement de piste du Thymio rapide, s'il est trop ralenti par le Thymio lent.
Notions *(cf. scénario conceptuel, page 302)*	Idem séances précédentes
Matériel	Pour la classe : • Un vidéoprojecteur • Deux longues pistes parallèles collées côte à côte, constituées de plusieurs tronçons rectilignes (Fiche 4), avec deux signaux « ligne de guidage » issus de la Fiche 7 Par groupe : • Deux robots Thymio • Un ordinateur avec *Aseba Studio* installé

Préparation

Le professeur reprend une piste rectiligne de la séance précédente, et lui adjoint une autre piste, composée majoritairement de tronçons de la Fiche 4 et de quelques tronçons de la Fiche 7, selon un schéma similaire à celui-ci (qui modélise une route à 2 voies) :



Situation déclenchante

Les élèves rappellent ce qu'ils ont élaboré lors de la séance précédente : deux suiveurs de ligne entrent en collision s'ils empruntent la même voie à des vitesses différentes. Il faut donc que le « lièvre » puisse doubler la « tortue », ce qui est impossible sur une piste unique.

Le professeur présente alors le terrain qu'il a préparé : deux pistes parallèles, simulant une route à double-sens à deux voies séparées par une ligne blanche continue. Des codes-barres (minimalistes : ce sont les blocs noirs) représentent le début de lignes discontinues (« lignes de guidage ») permettant au lièvre de changer de voie. Le professeur insiste sur le scénario à respecter. La tortue part sur la voie inférieure et la parcourt de gauche à droite. Le lièvre la suit à quelques secondes, sur la même voie. Lorsque le lièvre la rejoint, il doit adapter sa vitesse et se préparer à doubler. À partir de cet instant, le lièvre a le droit de doubler dès qu'il repère une des bornes de changement de ligne.

Les élèves doivent donc améliorer le programme de leur Thymio pour refléter cette nouvelle situation.

Trouver l'algorithme final du lièvre (classe entière)

Le professeur trace au tableau l'algorithme du comportement du lièvre, sous la dictée des élèves. Il les guide et les relance pour que ce logigramme inclue en particulier :

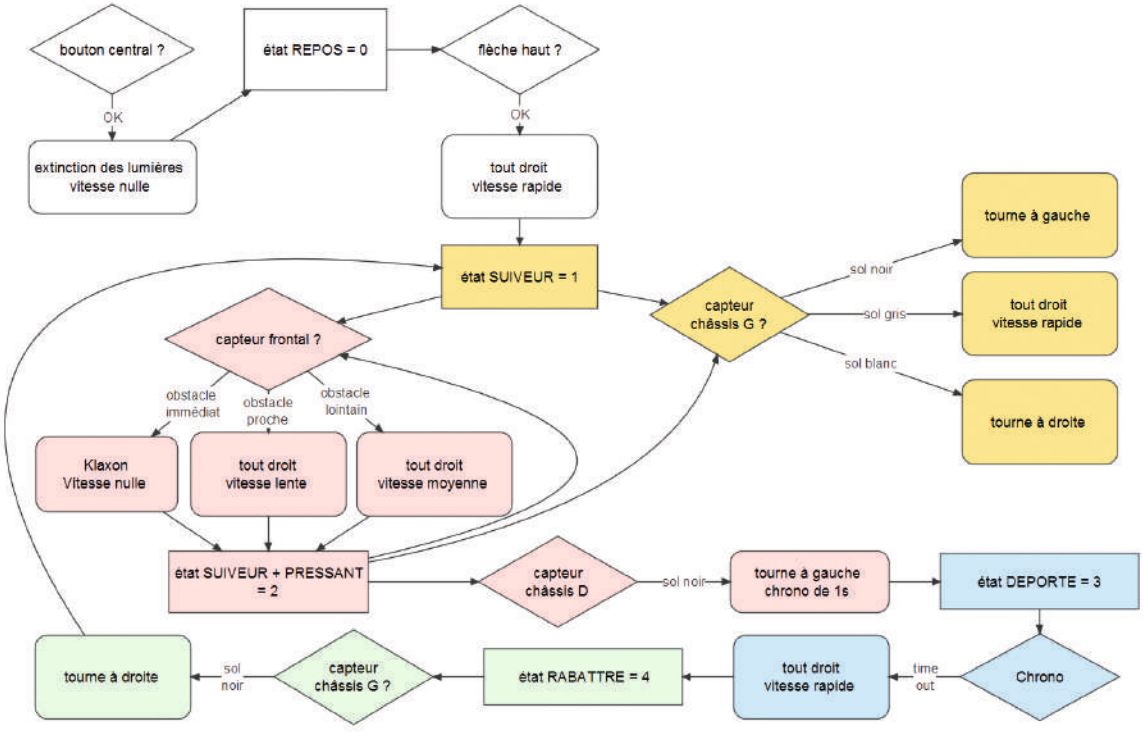
• Les différents états du lièvre
– Repos [code 0]
– Suiveur de ligne [code 1]
– Pressant : il suit la tortue et se prépare à doubler (il met son clignotant) [code 2]
– En train de se déporter : il change de voie [code 3]
– En train de se rabattre : il cherche à se rétablir sur la nouvelle voie [code 4]
• La direction du lièvre
• La vitesse du lièvre

Il est préférable de choisir un code commun pour les différents états du lièvre, afin de faciliter les mises en commun et le débogage des différents programmes.

Notes pédagogiques

• Il y a plusieurs façons de diriger cette étape. La première, proposée ici, est de réaliser en classe entière le logigramme au fur et à mesure. L'explicitation en français est difficile, car elle demande précision et rigueur : il est probable que cette séance prenne 2 heures au lieu d'une, et il faudra donc se concentrer sur le logigramme pendant la première heure et la programmation pendant la seconde. L'autre méthode, plus rapide, est d'afficher directement le logigramme proposé ci-dessous, et de demander aux élèves de vérifier s'il répond bien au problème.
• Ce logigramme ne prend pas en compte le retour du lièvre sur la piste de droite, car pour cela il faudrait s'assurer que la tortue est bien derrière lui, ce qui est particulièrement technique et difficile. Cela sort donc du cadre de cette initiation.

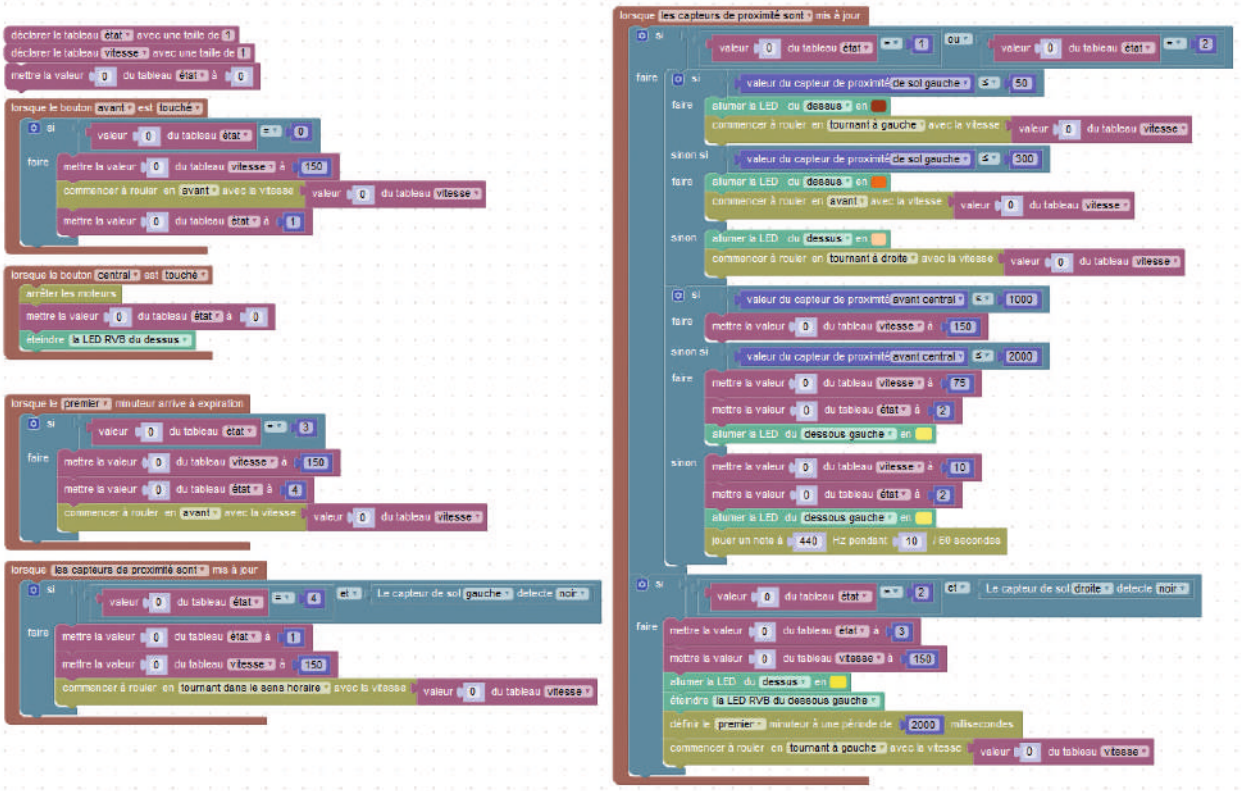
Les élèves aboutissent à un logigramme semblable à celui-ci :



Ce logigramme intègre déjà beaucoup d'algorithmes que les élèves ont implémentés : les blocs de couleur orange correspondent au suiveur de ligne amélioré de la Séance 5 ; les blocs de couleur rose correspondent quant à eux à la détection d'obstacles de la Séance 6.
La difficulté nouvelle consiste ici à bien découper les étapes du dépassement : en particulier, puisqu'il n'est pas possible de dire à Thymio « déporte-toi tant que tu ne croises pas de nouvelle ligne noire », il faut décomposer d'abord en « déporte-toi » puis en « va tout droit jusqu'à une nouvelle ligne noire », le tout temporisé grâce aux évènements « chronomètre ». En cela, les élèves redécouvrent que toute tâche complexe peut être décomposée en instructions élémentaires.

Implémenter l'algorithme du lièvre (par groupes)

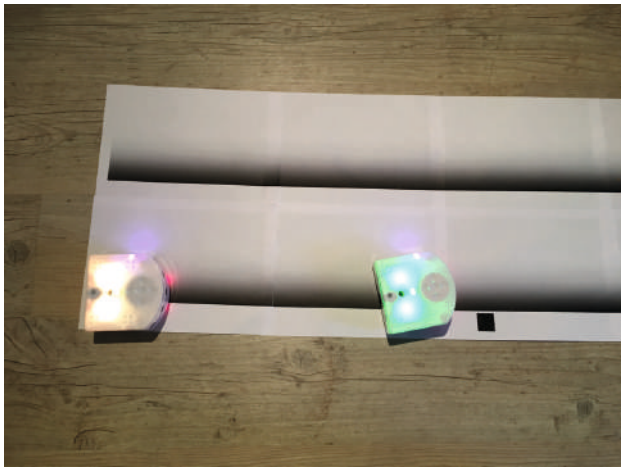
Aidés de ce logigramme, les élèves programment finalement le lièvre, et font régulièrement des tests sur une Fiche 4 ou une Fiche 7, en utilisant leur main comme obstacle mouvant.
Partant du programme précédent (le lièvre suit la ligne, rattrape la tortue et s'adapte à sa vitesse), il faut désormais :
– expliciter les états du lièvre à chaque étape ;
– ajouter les instructions du pseudo-lecteur de code-barres ;
– chronométrer le décrochage ;
– programmer le redressement sur la piste libre, à gauche, pour repartir en mode « suiveur » classique.
Le programme suivant est une solution possible.

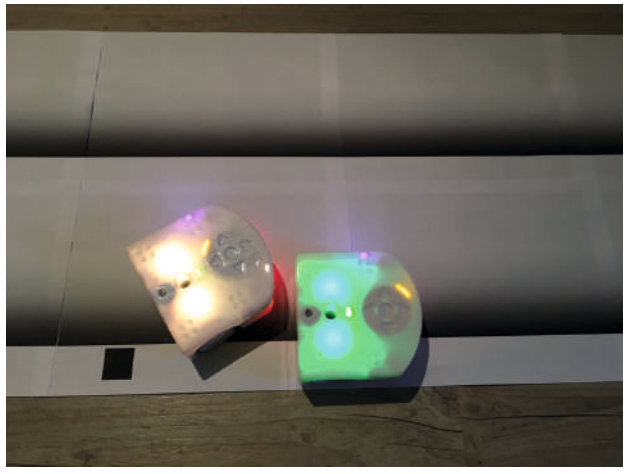


N.B. : ce programme corrigé, comme tous les autres, est disponible sur le site Web du projet (cf. page 404)

Conclusion

Dès que les groupes sont prêts, ils peuvent tester leurs Thymio sur cette nouvelle piste. Par l'essai-erreur, ils améliorent régulièrement leurs programmes. Au final, ils ont la satisfaction de voir le lièvre doubler la tortue, et filer sur la voie de gauche jusqu'à la ligne d'arrivée.





À gauche : situation initiale, la tortue a un peu d'avance sur la voie de droite, le lièvre la suit de peu sur la même voie. À droite : la tortue continue son chemin (état 1 ou mode SUIVEUR), tandis que le lièvre se déporte sur la gauche pour doubler (état 3 ou mode DEPORTE).

En fin de séance, les élèves reproduisent le logigramme final dans leur carnet de projet.

FICHE 7
Piste dégradée pour Thymio : dépassement autorisé

Séance 8 – Bilan : avantages et inconvénients de la voiture autonome

Discipline dominante	Technologie
Résumé	Les élèves organisent un débat pour échanger sur les avantages, les inconvénients, les risques et les possibilités des voitures autonomes.
Notions *(cf. scénario conceptuel, page 302)*	Machines : • Une voiture autonome est un robot.
Matériel	Pour la classe : • Un vidéoprojecteur • Un bâton de parole

Situation déclenchante

Les simulations automobiles que les élèves ont effleurées avec Thymio reposent sur des robots. Le professeur rappelle que pour améliorer le trafic routier, les élèves avaient mentionné les voitures autonomes. Quelle est la définition d'une voiture autonome ?

Les élèves énumèrent trois grandes qualités de tels véhicules : la présence de capteurs pour percevoir son environnement, le contrôle des actionneurs (moteur, vitesses, freins, feux) de la voiture, et l'omniprésence d'un ordinateur pour prendre les décisions. Capteurs, actionneurs, ordinateur, voilà encore une fois les trois fondements des robots.

Pourtant, loin de cette perspective révolutionnaire, Le professeur diffuse un reportage sur les accidents causés par ces robots. Par exemple, l'une de ces vidéos :

– http://mobile.francetvinfo.fr/sciences/high-tech/voiture-sans-pgroupee-tesla-une-enquete-ouverte-apres-le-premier-accident-mortel_1526815.html

– http://www.sciencesetavenir.fr/high-tech/drones/video-voici-le-film-de-l-accident-provoque-par-la-google-car_92765

La classe doit organiser un débat pour réfléchir aux règles à définir pour l'utilisation de ces nouveaux outils.

Joute orale : un débat sur les voitures autonomes (classe entière)

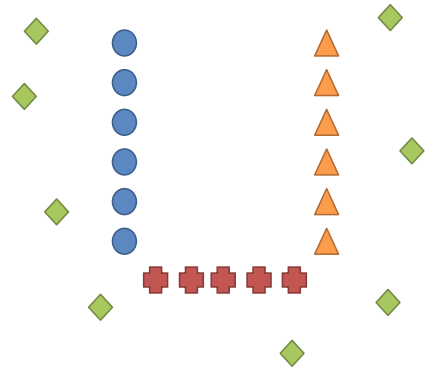
Le professeur énonce les règles du débat avant de le lancer.

La classe va se scinder en quatre groupes de tailles similaires, remplis sur la base du volontariat, mais pas forcément sur la base des convictions personnelles des élèves :

– Les « pour », qui veulent défendre les voitures autonomes et encourager leur démocratisation [ci-contre, les ronds bleus] ;

– Les « contre », qui veulent bannir cette nouvelle technologie [ci-contre, les triangles orange] ;

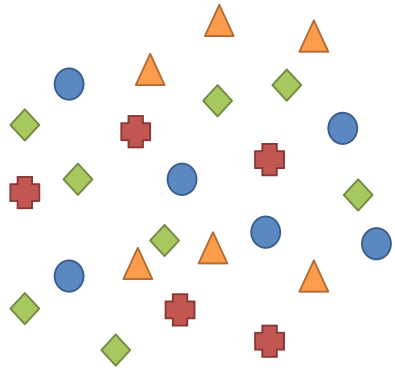
– Les « animateurs », qui vont distribuer la parole, relancer le débat et assurer la gestion du temps [ci-contre, les croix rouges] ;



– Les « observateurs », qui ne participeront pas du tout au débat mais en relèveront les arguments pour illustrer le débriefing [ci-contre, les losanges verts].
Idéalement, les « pour » et les « contre » seront assis face à face, de chaque côté de la pièce. Les « animateurs » seront placés entre les deux groupes, pour former un U. Les « observateurs » seront placés tout autour, en dehors du cercle de parole.

> **Note pédagogique**
> Selon le temps disponible ou l'autonomie des élèves, on peut au choix laisser quelques minutes de réflexion aux groupes pour préparer leurs arguments, ou alors leur demander de réaliser une recherche documentaire dans les jours qui précèdent.

Le « président » des « animateurs » dispose d'un « bâton de parole » qui donne le droit de parler exclusivement à celui qui le brandit. Les autres animateurs s'assureront du respect de cette règle, et de l'équité des temps de parole. Lorsque le possesseur du « bâton de parole » a terminé l'énoncé de son argument, il rend le bâton pour que les animateurs puissent le transmettre à un membre de l'autre groupe. Ainsi, chaque membre des groupes « pour » et « contre » aura eu l'occasion de donner au moins un argument en faveur de son camp. Un dernier animateur, le garant du temps, signale la fin du temps imparti (10, 15, 20 minutes maximum), que le débat ait abouti à un consensus, ou pas.
La dernière étape du débat, menée par les « observateurs », est la restitution : tous les participants sont réunis en un seul grand groupe, où l'on résume rapidement les arguments majeurs des deux camps. C'est l'occasion de relever les divergences d'opinion, et de tenter d'aboutir à une réponse commune à la question posée. Les élèves peuvent également témoigner de ce que le débat en lui-même leur a apporté.



Conclusion

Les élèves élaborent ainsi une conclusion commune, qu'ils reproduisent dans leur carnet de projet :
– Les voitures autonomes sont des robots, comme Thymio
– Pour garantir la sécurité des usagers, il faut imposer des règles afin d'encadrer légalement l'usage et la conception des voitures autonomes.
– L'écrivain Isaac Asimov est connu pour ses réflexions sur la robotique, qui l'ont conduit à énoncer trois lois fondamentales :
1. Les robots doivent protéger les humains.
2. Les robots doivent obéir aux humains, sauf si cela contredit la règle 1.
3. Les robots doivent protéger leur existence, sauf si cela contredit l'une des autres règles.

Variante : faire le projet avec *Aseba*/VPL

Discipline dominante	Technologie
Résumé	Cette variante reprend les principales étapes des séances de programmation de Thymio, adaptées pour le langage *Aseba*/VPL.

Avant-propos

Cette annexe reprend exactement le déroulé du projet « robotique avec Thymio » : tout le fil en est détaillé au cours des pages précédentes, et nous ne reprenons ici que les corrigés des étapes ou exercices, avec quelques commentaires pertinents le cas échéant.

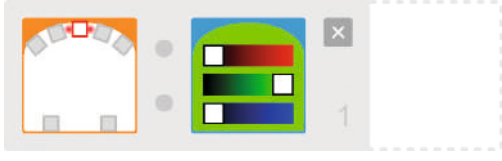
Préparation

L'utilisation de VPL requiert l'installation d'*Aseba Studio*, que nous avons déjà décrite (https://www.thymio.org/fr:start). Le lancement est cependant un peu différent.

Pour lancer VPL	
Méthode 1	Méthode 2
1. Brancher Thymio sur l'ordinateur avec le câble USB (il s'allume) **2.** Lancer Thymio-VPL	**1.** Lancer Thymio-VPL (une fenêtre « Choix d'une cible *Aseba* » s'ouvre). **2.** Brancher Thymio sur l'ordinateur avec le câble USB (il s'allume). **3.** Cocher la case « Port série », sélectionner « Thymio-II Robot », appuyer sur « Connecter ».
Pour programmer	
1. Écrire le programme **2.** Sauvegarder le programme **3.** Exécuter le programme	

Séance 2 : premier programme en VPL

Le professeur écrit à l'écran le programme suivant :



N.B. : VPL présente les instructions sous la forme de paires « capteur-actionneur » permettant d'expliquer quelle action effectuer dans quel cas. Les capteurs sont toujours à gauche et les actionneurs à droite. Il demande aux élèves d'essayer de prédire ce que fera le programme (et ce qu'il ne fera pas : revenir à l'état initial).

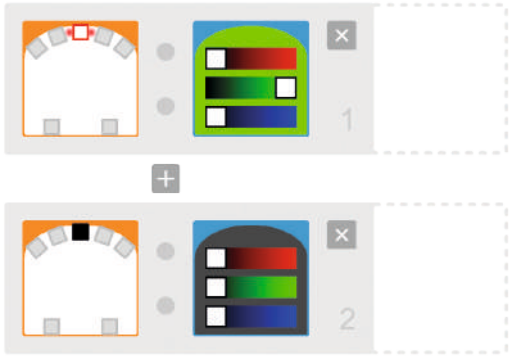
Dans VPL, les composants du robot s'affichent très distinctement :

• Capteurs (représentés dans le langage VPL par des cartes sur fond orange) : à noter qu'il n'est pas possible dans VPL d'en lire des valeurs numériques. Dans « VPL avancé », on pourra nuancer un peu plus ce constat, mais de manière moins fine qu'avec *Blockly*.

• Actionneurs (représentés dans le langage VPL par des cartes sur fond bleu).

• Ordinateur : apparie des évènements liés aux détections par les capteurs (colonne de gauche sur l'interface VPL) et des actions liées aux actionneurs (colonne de droite).

Après l'étude de ce cas, les élèves dicteront comment compléter ce programme par un contrordre :



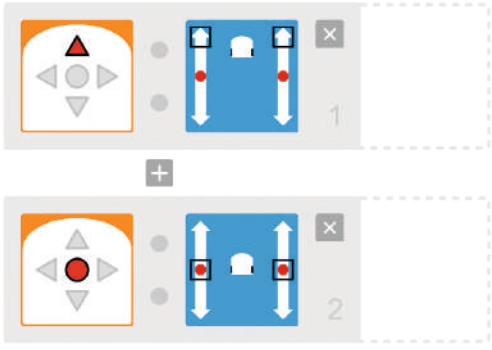
Le professeur démontre ainsi deux états fondamentaux des capteurs de Thymio vus par VPL : actif (icône blanche et rouge) et inactif (icône noire). Il passe sous silence pour l'instant l'état « indifférent » (icône grise) qui sera longuement approfondi à la séance suivante.

Note pédagogique :

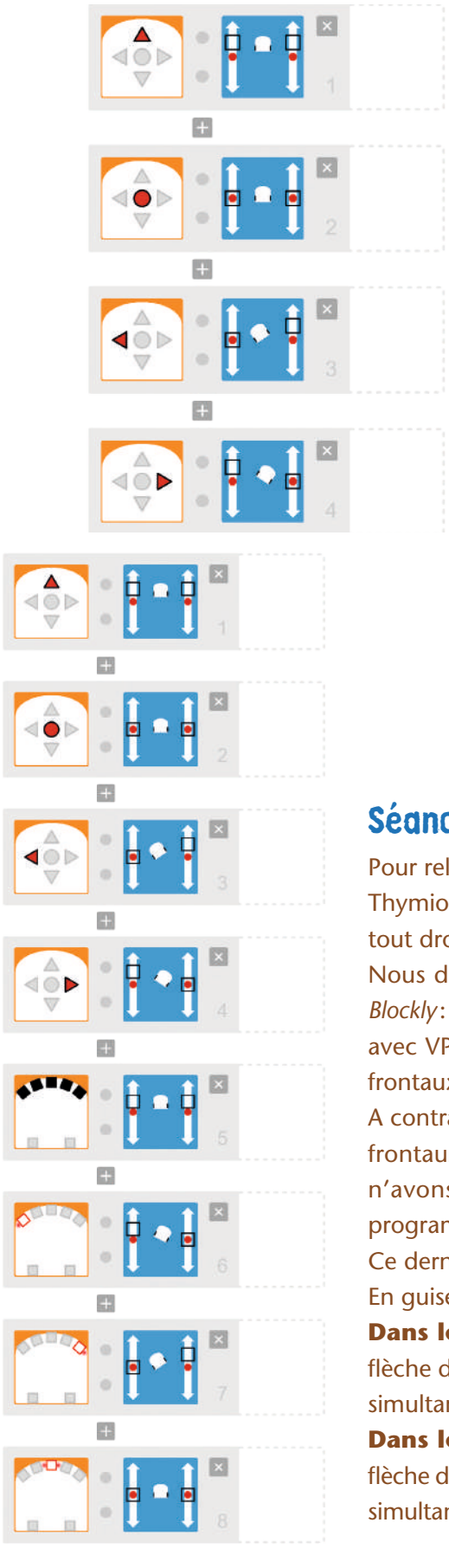
L'interface de VPL présente à droite l'équivalent textuel du programme graphique réalisé. Cela permet d'aborder le fait qu'il existe de nombreux langages de programmation, certains graphiques, certains textuels, mais que tous doivent être interprétés par la machine.

Séance 2 : défis pour prendre VPL en main

Défi 1 : Thymio avance grâce à la flèche du haut et s'arrête grâce au bouton central



Défi 2 : Thymio tourne grâce aux flèches



Défi 3 : Thymio s'arrête si un obstacle est devant lui

À ce stade, nous arrêtons l'imitation du mode mauve pour incorporer une détection d'obstacles.

Séance 3 - Défi 4 : Thymio contourne les obstacles

Pour relever le défi 4, il faut prévoir une porte de sortie (ligne 5) : si Thymio ne détecte explicitement rien devant lui, alors il peut aller tout droit (sinon il ne redémarrerait jamais).

Nous découvrons ici la première différence notable entre VPL et *Blockly* : il est facile d'ajouter des conditions additives (logique « ET ») avec VPL, ce qui nous permet de gérer facilement les 5 capteurs frontaux.

A contrario, avec *Blockly*, l'expression combinant tous ces capteurs frontaux aurait été bien plus longue. Par souci de lisibilité, nous n'avons donc traité que 3 capteurs sur 5 avec *Blockly*. Les 2 programmes sont donc similaires, mais pas identiques.

Ce dernier défi oblige les élèves à explorer des logiques ET ou OU. En guise de consolidation, le professeur distribue la Fiche 8.

Dans le cas A, Thymio avance tout droit lorsqu'on appuie sur la flèche du haut ; SI les trois capteurs centraux détectent un obstacle simultanément, ALORS Thymio s'arrête. C'est une logique « ET ».

Dans le cas B, Thymio avance tout droit lorsqu'on appuie sur la flèche du haut ; SI l'un des trois capteurs centraux détecte un obstacle simultanément, ALORS Thymio s'arrête. C'est une logique « OU ».

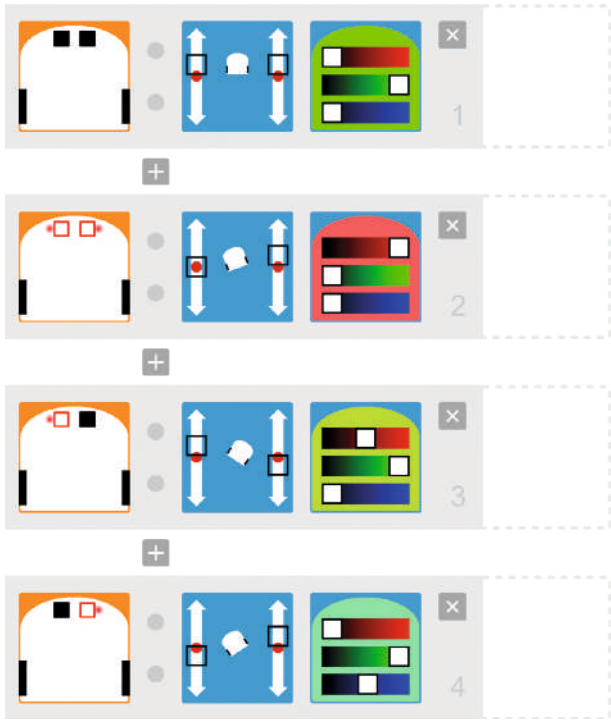
Séance 4 – Défi 5 : imiter le Thymio pisteur (cyan)

1. SI le capteur de châssis gauche détecte un sol noir ET le capteur de châssis droit détecte un sol noir, ALORS le capot s'allume en vert et Thymio avance tout droit.

2. SI le capteur de châssis gauche détecte un sol blanc ET le capteur de châssis droit détecte un sol blanc, ALORS le capot s'allume en rouge et Thymio tourne à gauche.

3. SI le capteur de châssis gauche détecte un sol blanc ET le capteur de châssis droit détecte un sol noir, ALORS le capot s'allume en vert anis et Thymio pivote sur place vers la droite.

4. SI le capteur de châssis gauche détecte un sol noir ET le capteur de châssis droit détecte un sol blanc, ALORS le capot s'allume en turquoise et Thymio pivote sur place vers la gauche.



Les élèves peuvent remarquer que l'absence d'événement est aussi, d'un point de vue programmatique, un événement à part entière. En effet, les capteurs de Thymio ne sont pas capables stricto sensu de détecter un sol noir : précisément, dans ce cas-là ils signalent qu'ils ne détectent pas de sol blanc. La programmation VPL permet donc de vérifier l'état actif des capteurs (carré blanc avec lumière rouge = « je détecte »), passif (carré noir = « je ne détecte rien ») ou indifférent (carré gris = « je ne regarde pas l'état de ce capteur »).

D'autre part, cet exemple confirme la logique exclusive de VPL : les capteurs impliqués dans une même paire action-évènement sont liés par un opérateur logique ET ! Si l'on voulait une logique additive (un opérateur logique OU), il faudrait créer deux lignes différentes, avec des états indifférents pour chaque capteur à tour de rôle.

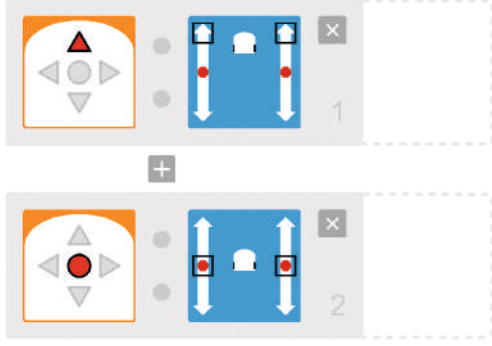
Comme illustré ici, on peut associer plusieurs actions à un seul et même évènement. C'est facultatif pour l'instant, mais cela deviendra vite obligatoire, dès la séance suivante. (Dans des versions précédentes de VPL 1.5, il était possible de créer plusieurs lignes concernant un seul et même évènement, désormais toutes les actions sont forcément rassemblées ; cela facilite non seulement la lecture mais également le débogage.)

Séance 5 : Thymio dans tous ses états

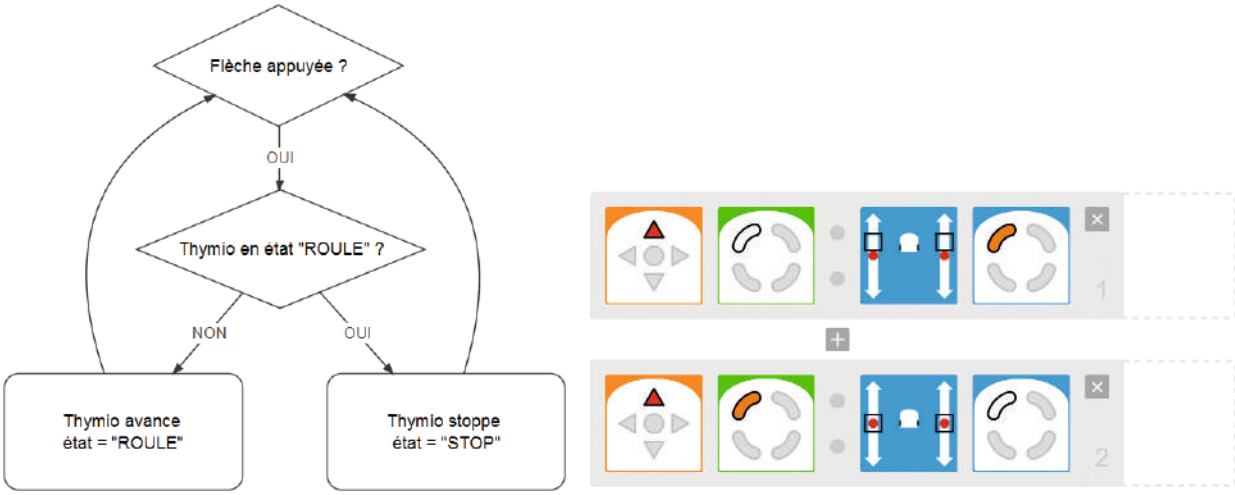
Note pédagogique

Cette étape est indispensable en programmation VPL, car ce langage de programmation, en version simplifiée, ne permet pas de créer des variables ou de manipuler de la mémoire. Cependant, le saut conceptuel qui permet de passer de VPL « simple » à VPL « avancé » risque de freiner la progression des élèves les plus en difficulté.

Les élèves arrivent à la conclusion qu'il faut dédier un seul capteur au suivi de la piste (disons le gauche) et libérer l'autre (ici le droit) pour la lecture du code-barres. Il faut donc apprendre à :
– Suivre une piste avec un seul capteur
– Lire un code-barres
– Retenir quel mode utiliser en fonction de l'information du code-barres
Le professeur annonce que Thymio a effectivement une (toute) petite mémoire interne qu'il est possible d'utiliser pour mémoriser des sous-programmes différents à utiliser dans des conditions différentes. (Dans la plupart des langages de programmation, *Blockly* en particulier, cette entité serait appelée « variable ».) Il introduit alors le mode « avancé » de la programmation Thymio : ce mode permet de réaliser des programmations plus fines, en jouant sur la sensibilité des capteurs et en ayant accès aux « états internes » de Thymio. Le professeur fait la démonstration sur son propre ordinateur projeté au tableau : ce mode s'active en appuyant sur l'icône « lauréat » de *VPL*.
Il y a plusieurs changements remarquables : les icônes des évènements sont plus complexes et il y a de nouvelles cartes avec des réveils ou un cercle de 4 segments. Les premières permettent de gérer des chronomètres, et les secondes donnent accès à l'interface des états de Thymio. On retrouve également ce cercle segmenté combiné systématiquement à chaque action sous forme d'une carte verte.
Le professeur reprend alors le programme du premier exercice effectué par les élèves :



Il leur demande de trouver l'algorithme qui permettrait d'avoir le même comportement, mais en appuyant exclusivement sur la flèche « haut ». Ils élaborent un algorithme comme celui-ci, que le professeur traduit aussitôt sous *VPL* :



Les élèves appliquent ce programme à leur propre Thymio, pour vérifier ce qu'il fait. C'est également la première fois qu'ils doivent appliquer plusieurs actions au même évènement. Ils observent en particulier que le capot du Thymio s'illumine partiellement (arc N-W) lorsqu'il roule : c'est la signature de l'état interne qui a été activé.

Les élèves peuvent éventuellement remarquer que les états se gèrent comme des capteurs : il est possible de combiner des états, ou d'en ignorer d'autres (arc blanc = état 0, arc orange = état 1, arc gris = état indifférent). Le professeur demande aux élèves de compter combien d'états internes sont disponibles : puisqu'il y a quatre « cadrans » (on pourrait parler de « bits »), chacun avec deux états explicites, il y a 2 (pour le premier bit) x 2 (pour le second bit) x 2 (pour le troisième bit) x 2 (pour le quatrième bit) = 16 états possibles… Thymio peut donc différencier au plus 16 situations différentes. Par défaut, à l'allumage, Thymio est dans l'état 0000.

Dans l'exemple ci-dessus, l'état « ROULE » est codé par 1XXX, et l'état « STOP » est codé par 0XXX. (On note par X les bits non pertinents. On peut bien évidemment les expliciter 1000 et 0000.)

Note pédagogique

Il n'est pas attendu ici que les élèves comprennent que 4 bits permettent de décrire 16 informations différentes (ici, des états internes de Thymio). L'approche combinatoire proposée suffit. Cela peut néanmoins servir de déclencheur pour une séance parallèle sur le binaire. Voir à ce sujet la séquence 1 du projet « 1, 2, 3… codez ! » pour le cycle 3.

Séance 5 : sensibilité de Thymio

Sous VPL, le programme ressemble à ceci :

1. SI Thymio est dans l'état 0XXX ET que la flèche « haut » est appuyée, ALORS Thymio avance et passe dans l'état 1XXX.

2. Quel que soit l'état de Thymio, SI le bouton central est appuyé, ALORS Thymio s'arrête et passe dans l'état 0XXX.

3. SI Thymio est dans l'état 1XXX ET que le capteur de châssis gauche détecte un sol blanc, ALORS le capot de Thymio se colore en rouge.

4. SI Thymio est dans l'état 1XXX ET que le capteur de châssis gauche détecte un sol gris, ALORS le capot de Thymio se colore en vert.

5. SI Thymio est dans l'état 1XXX ET que le capteur de châssis gauche détecte un sol noir, ALORS le capot de Thymio se colore en bleu.

Les élèves découvrent ainsi qu'il est possible d'obtenir des valeurs plus nuancées sur les capteurs, avec une quatrième option disponible : « mi-noir mi-blanc », définie par des seuils minimum et maximum.

Note pédagogique

Dans le cadre de ce calibrage, il n'est évidemment pas obligatoire de faire avancer le robot, et encore moins d'ajouter les états internes, mais c'est une bonne occasion de s'approprier l'outil vu à l'instant. De plus, cela permet d'écrire des programmes plus propres, avec des comportements distincts « repos » et « actif » : Thymio ne commence à agir que lorsqu'on le déclenche.

Séance 5 : suiveur de ligne amélioré

Le programme de la tortue est désormais :

1. SI Thymio est dans l'état 0XXX ET que la flèche « haut » est appuyée, ALORS Thymio avance et passe dans l'état 1XXX.

2. Quel que soit l'état de Thymio, SI le bouton central est appuyé, ALORS Thymio s'arrête et passe dans l'état 0XXX.

3. SI Thymio est dans l'état 1XXX ET que le capteur de châssis gauche détecte un sol blanc, ALORS le capot se colore en vert clair et Thymio pivote vers la droite.

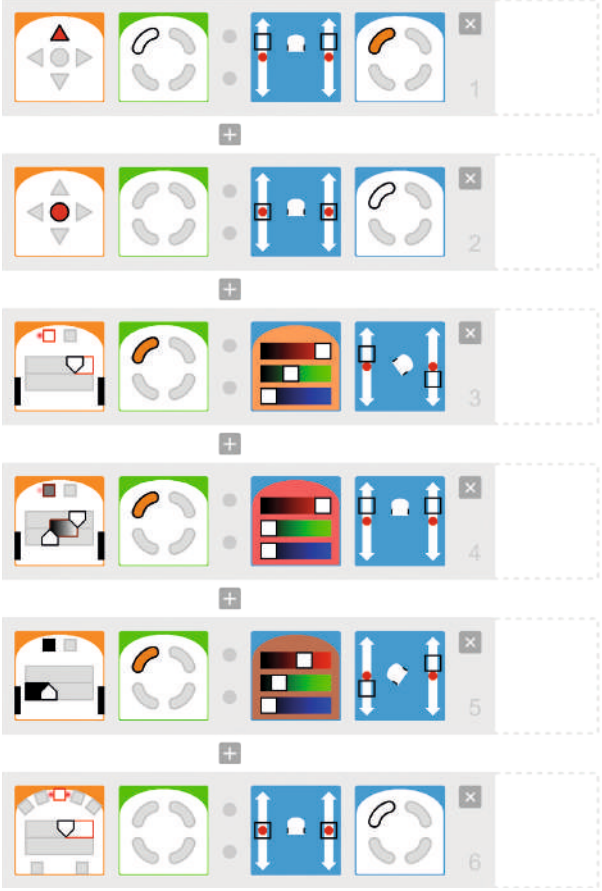
4. SI Thymio est dans l'état 1XXX ET que le capteur de châssis gauche détecte un sol gris, ALORS le capot se colore en vert vif et Thymio avance tout droit.

5. SI Thymio est dans l'état 1XXX ET que le capteur de châssis gauche détecte un sol noir, ALORS le capot se colore en vert foncé et Thymio pivote vers la gauche.

Les différents états de Thymio sont définis ici de la façon suivante :
0XXX = « repos »
1XXX = « suiveur ».

Séance 6 : le programme du lièvre

Solution 1 : freinage d'urgence



Solution 2 : suivre la tortue

Séance 7 : programme final du lièvre

Le programme suit le logigramme dressé par la classe, avec une dénomination différente pour les différents états :

- Repos [code 0000]
- Suiveur de ligne [code 1000]
- Pressant : il suit la tortue et se prépare à doubler (il met son clignotant) [code 1100]
- En train de se déporter : il change de voie [code 0010]
- En train de se rabattre : il cherche à se rétablir sur la nouvelle voie [code 0001]

Cette proposition combine astucieusement les codes des états « suiveur de ligne » et « pressant », car même si le lièvre cherche désormais à doubler dans l'état « pressant », il est tout de même obligé de suivre la ligne tant qu'il n'a pas eu l'autorisation de changer de file. Par contre, il faut cesser immédiatement de suivre la ligne dès que le signal du dépassement est donné.

En ce sens, l'état 1X00 répond parfaitement à ce critère, car c'est le point commun entre les états « suiveur 1000 » et « pressant 1100 ». Le programme suivant est une solution possible. En particulier, il est optimisé au sens où il limite les cas redondants. Par exemple, il est possible de programmer deux lignes distinctes « si mon état est 1000 [SUIVEUR] et que mon capteur de châssis gauche voit un sol gris, alors je vais tout droit » et « si mon état est 1100 [PRESSANT] et que mon capteur de châssis gauche voit un sol gris, alors je vais tout droit », ou alors de factoriser tout ceci en « si mon état est 1X00 et que mon capteur de châssis gauche voit un sol gris, alors je vais tout droit ». Il est envisageable de simplifier beaucoup le programme proposé, mais au détriment de la lisibilité, ce qui à ce stade d'apprentissage ne serait pas constructif.

1. SI Thymio est dans l'état 0000 ET que la flèche « haut » est appuyée, ALORS Thymio avance tout droit à grande vitesse et passe dans l'état 1000.

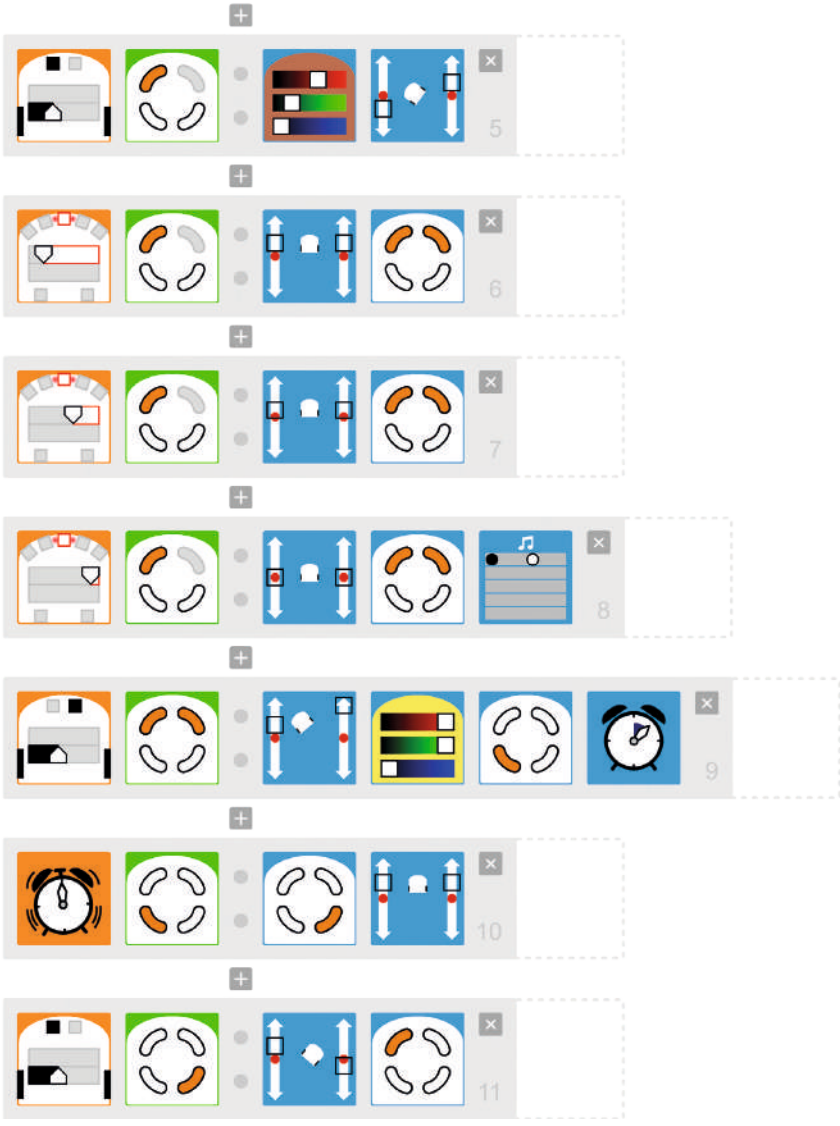
2. Quel que soit l'état de Thymio, SI le bouton central est appuyé, ALORS Thymio s'arrête, passe dans l'état 0XXX, et son capot s'éteint.

3. SI Thymio est dans l'état 1X00 ET que le capteur de châssis gauche détecte un sol blanc, ALORS le capot se colore en orange et Thymio pivote vers la droite.

4. SI Thymio est dans l'état 1X00 ET que le capteur de châssis gauche détecte un sol gris, ALORS le capot se colore en rouge et Thymio avance tout droit à grande vitesse.

.../...

.../...



5. SI Thymio est dans l'état 1X00 ET que le capteur de châssis gauche détecte un sol noir, ALORS le capot se colore en brun et Thymio pivote vers la gauche.

6. SI Thymio est dans l'état 1X00 ET que le capteur frontal détecte un obstacle lointain, ALORS Thymio avance tout droit à vitesse modérée et passe dans l'état 1100.

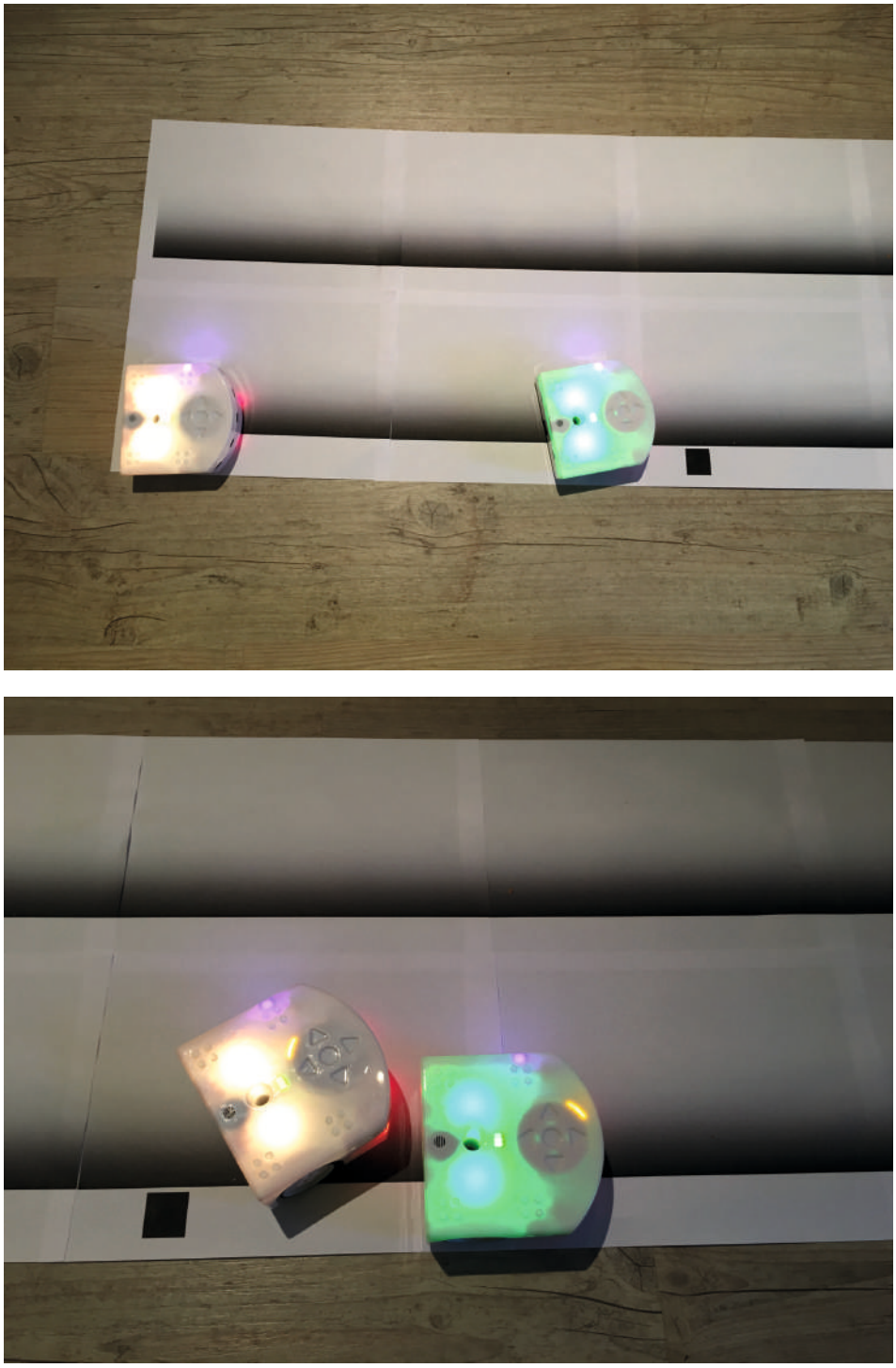
7. SI Thymio est dans l'état 1X00 ET que le capteur frontal détecte un obstacle proche, ALORS Thymio avance tout droit à vitesse lente et passe dans l'état 1100.

8. SI Thymio est dans l'état 1X00 ET que le capteur frontal détecte un obstacle immédiat, ALORS Thymio freine, émet un son, et passe dans l'état 1100.

9. SI Thymio est dans l'état 1100 ET que le capteur de châssis droit détecte un sol noir, ALORS Thymio se déporte à gauche, passe dans l'état 0010, déclenche un chrono de 2 s et le capot se colore en jaune.

10. SI Thymio est dans l'état 0010 et que le chrono s'arrête, alors Thymio avance tout droit à grande vitesse et passe dans l'état 0001.

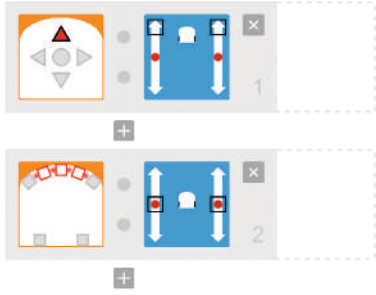
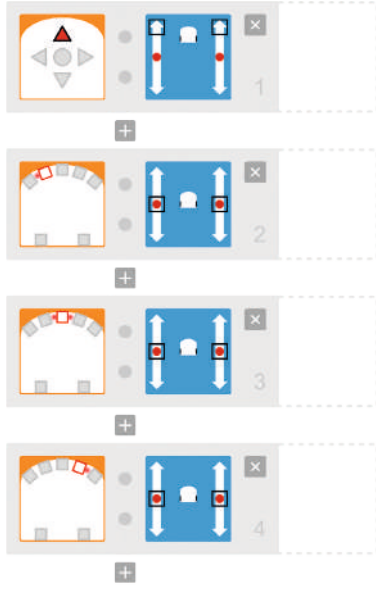
11. SI Thymio est dans l'état 0010 et que le capteur de châssis gauche détecte un sol noir, ALORS Thymio pivote à droite et passe dans l'état 1000.



Au-dessus : situation initiale, la tortue a un peu d'avance sur la voie de droite, le lièvre la suit de peu sur la même voie. Au-dessous : la tortue continue son chemin (on repère bien ici l'état 1000 du mode SUIVEUR), tandis que le lièvre se déporte sur la gauche pour doubler (on repère également l'état 0010 du mode DEPORTE).

FICHE 8
Thymio et les opérateurs logiques de VPL

Consigne : Dans les deux cas suivants, écris d'abord tes hypothèses, en fonction de ce que tu devines du comportement probable de Thymio. Ensuite, reproduis ces programmes sur ton propre Thymio et écris alors tes observations. Qu'en penses-tu ?

	Cas A	Cas B
Programme		
Hypothèses		
Observations		

Cette ressource est issue du projet thématique *1,2,3... CODEZ !*, paru aux Éditions Le Pommier.

Retrouvez l'intégralité de ce projet sur : https://www.fondation-lamap.org/projets-thematiques.

Fondation *La main à la pâte*
43 rue de Rennes
75006 Paris
01 85 08 71 79
contact@fondation-lamap.org

Site : www.fondation-lamap.org

FONDATION La main à la pâte
POUR L'ÉDUCATION À LA SCIENCE