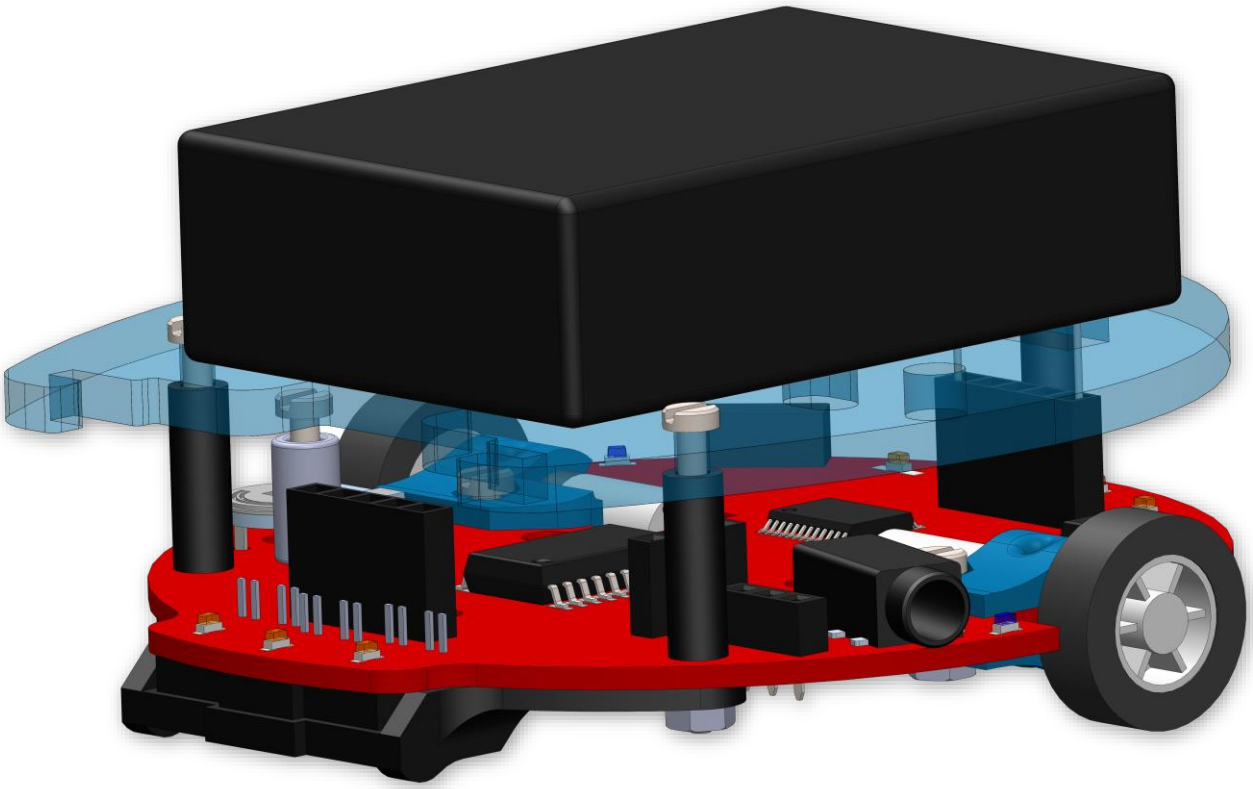
Robot LouPiot V1
Programmation avec Blockly



TECHNOLOGIE

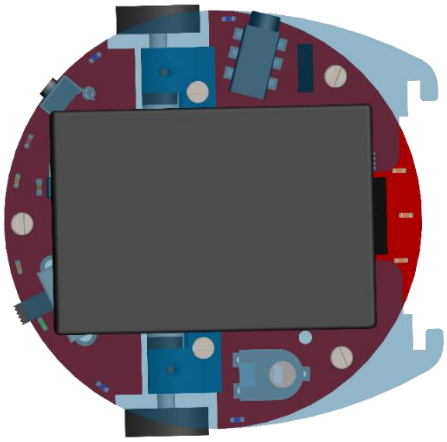
Ressources disponibles pour le projet K-LP

Autour du projet Loupiot, nous vous proposons un ensemble de **ressources téléchargeables gratuitement sur le wiki :**

Lien wiki: http://a4.fr/wiki/index.php/Loupiot

SOMMAIRE

Introduction



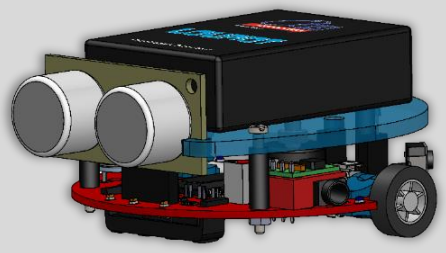
Loupiot est un robot de table conçu pour la découverte et l'apprentissage de la programmation.
Son faible encombrement permet de mener facilement des activités robotiques dans un espace restreint se limitant à une feuille format A4 ou A3 à proximité de l'ordinateur utilisé pour le programmer.
Fonctionnant à partir d'un microcontrôleur PICAXE 20M2, Loupiot possède 5 capteurs et 5 actionneurs (listés ci-dessous).

Ce document propose un parcours progressif pour découvrir et se perfectionner avec la programmation en se basant sur une série d'exemples ludiques autour de Loupiot grâce à ses capteurs et actionneurs. Tous les exemples sont réalisés à partir de PICAXE Blockly pour Windows et les premiers exemples de programmes sont également fournis avec la version logigramme correspondante.

Version de base – Capteurs et actionneurs

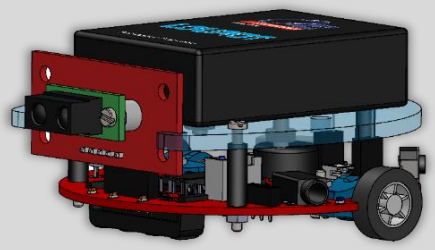
- 3 capteurs de ligne permettant de suivre une ligne noire opaque.
- Une entrée de lecture permettant de lire la tension des piles du robot pour connaitre le restant d'autonomie.
- 3 LED programmables (deux témoins lumineux oranges droit et gauche et un témoin lumineux rouge pour signaler une alerte).
- Une horloge permettant de savoir depuis combien de secondes le robot est allumé.
- Deux moteurs avec vitesse et sens de rotation réglables.

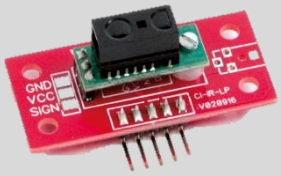
Options disponibles



Option capteur de distance à ultrason :
Renvoie une valeur de distance entre 0 et 2,5 mètres (valeur du capteur dans le vide, environ 0 à 0,75 cm quand placé sur le robot).

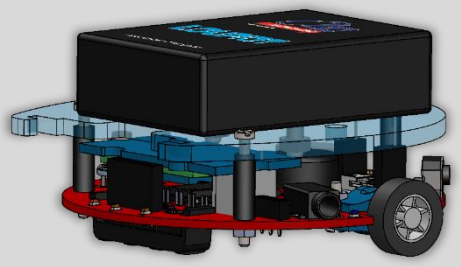
Réf : K-LP-US





Option détection d'obstacle :
Un capteur de proximité infrarouge prévient quand un obstacle est détecté à moins de 5 cm.

Réf : K-LP-IR



Option Bluetooth :
Module Grove permettant la communication avec un smartphone Android.

Réf : S-113020008

Option porte stylo :
Permet de tracer des lignes de couleurs.
Comprend des markers et des pistes effaçables.

Réf : K-LP-PS

Prérequis

Pour la version de base

- Câble de programmation USB : Câble de programmation PICAXE (Réf : CABLE-USBPICAXE)
- 3 piles AAA de 1,5 Volts ou accu AAA 1,2 Volts (Réf : voir-ci-dessous)

Nous proposons 3 références de piles compatibles avec le robot Loupiot :

Piles alcalines (piles conseillées)	Réf : PILE-LR03-10	Test d'endurance : 10 h*
Piles salines	Réf : PILE-R03-4	Test d'endurance : 3h30*
Accus 1,2V 800 mAh (rechargeable)	Réf : ACCU-NIMH-2R03-0A8	Test d'endurance : 6h30*

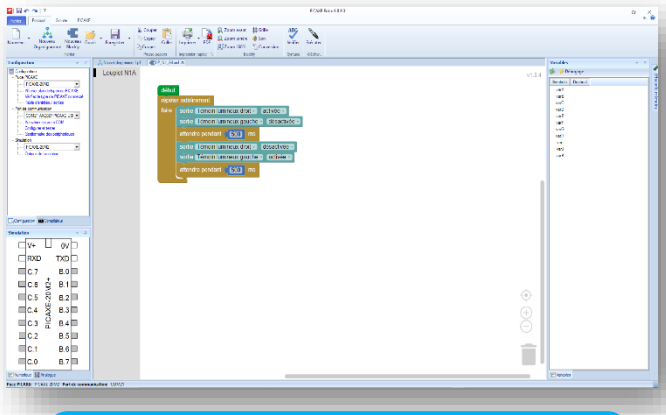
* Le test d'endurance a été réalisé sur la piste de démonstration du Loupiot avec le sous-programme suivi de ligne. Le temps relevé correspond à la perte de la ligne à partir d'un réglage des capteurs optimal en début de programme.

Pour les options disponibles

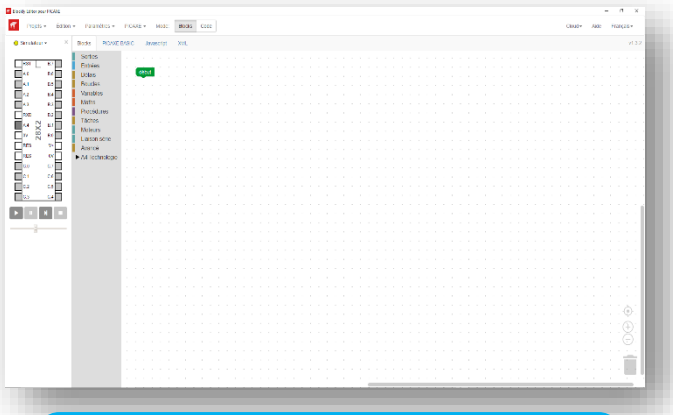
- Pour l'option module Bluetooth, être munis d'un smartphone Android (dans les paramètres de sécurité, la case « Sources inconnues » doit être au préalable cochée pour pouvoir installer les applications d'A4).
- Pour l'option capteur de distance à ultrasons, ne pas oublier de brancher le module stepUp voltage (fourni dans l'option) sur le robot à l'emplacement prévu.

Logiciels

Avoir téléchargé la dernière version de BlocklyForPicaxe ou Editor 6 disponible sur www.a4.fr
Lien : https://www.a4.fr/logiciels-et-livres/logiciels/programmation/graphique-par-blocs/blockly.html



Picaxe Editor 6

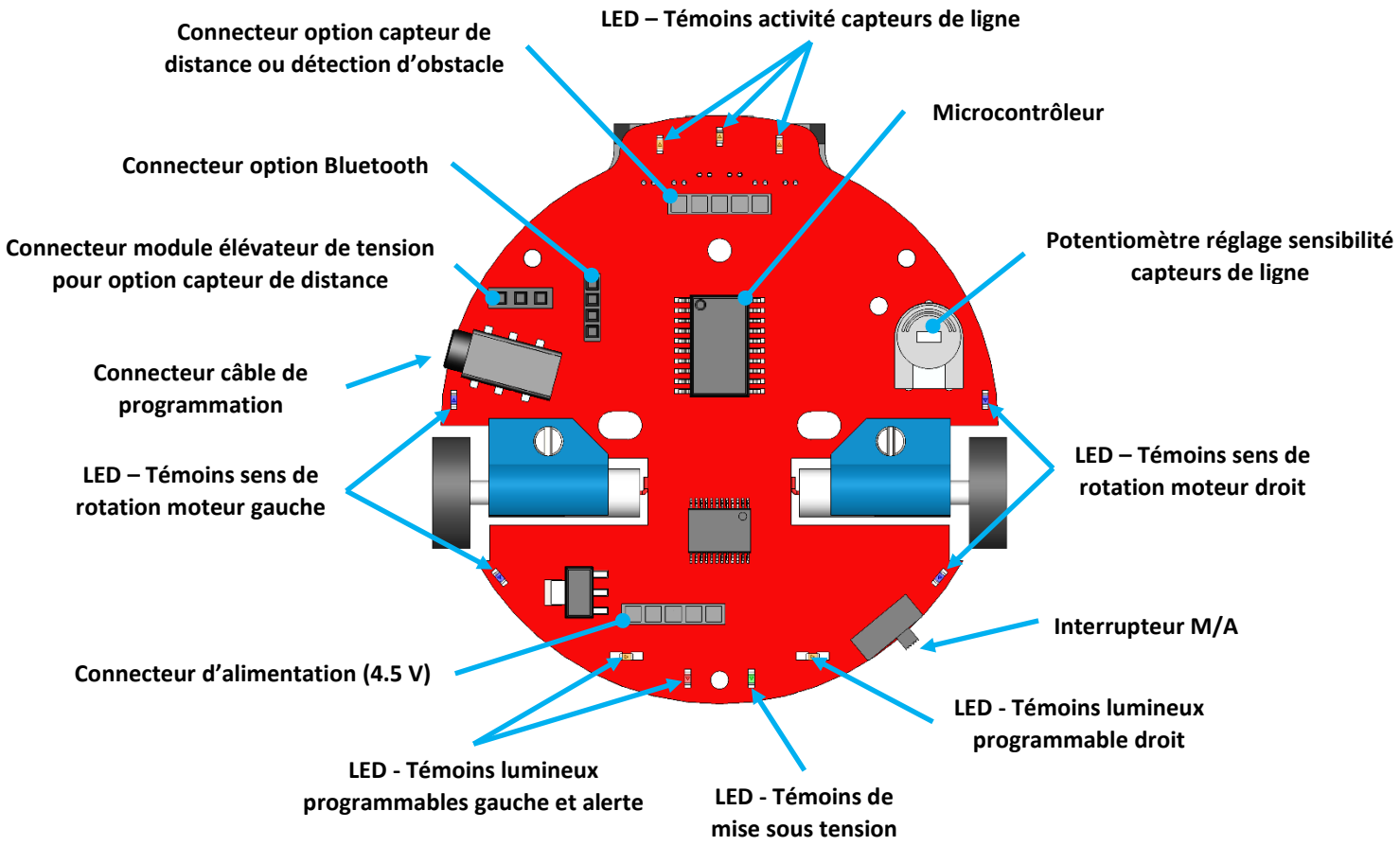


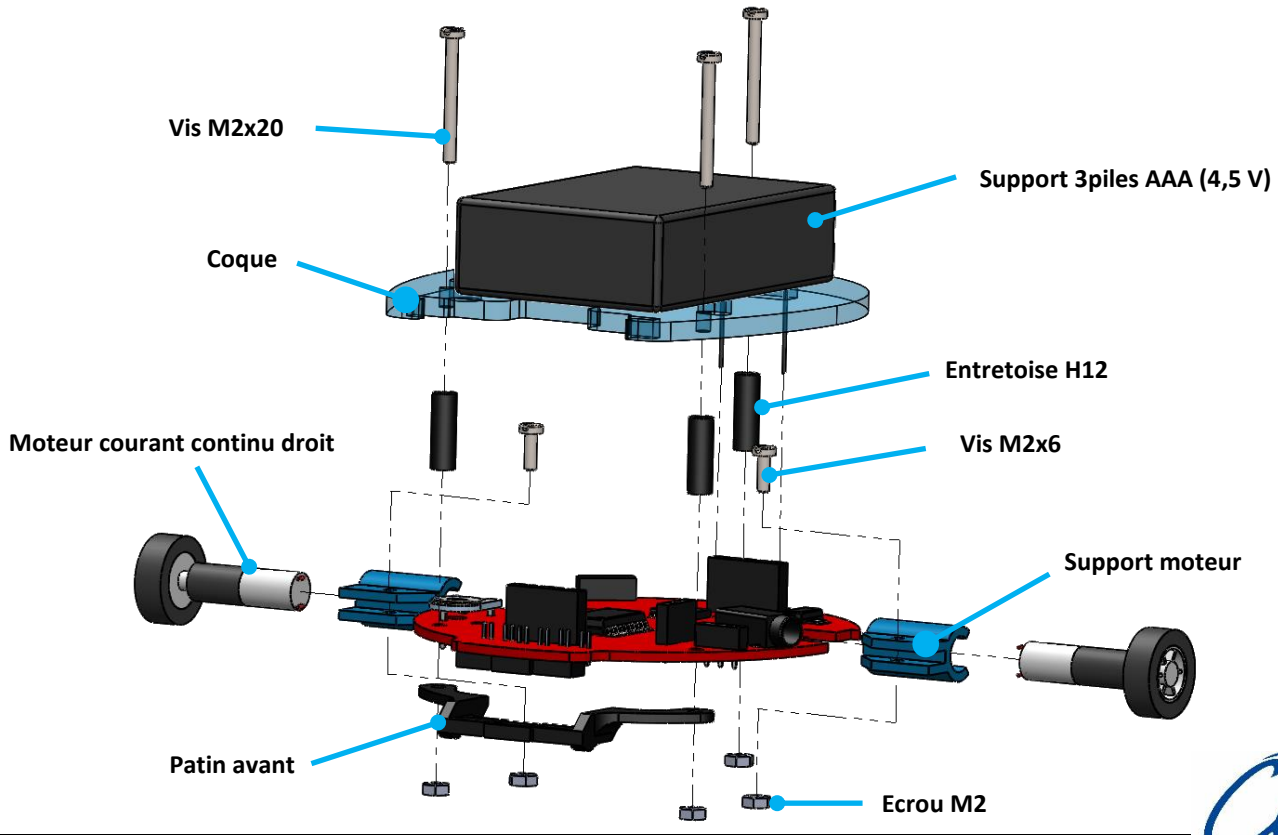
Blockly For Picaxe

Aide complémentaire

Prise en main de BlocklyForPicaxe : http://a4.fr/wiki/index.php/PICAXE_Blockly
Prise en main de Picaxe Editor : http://a4.fr/wiki/index.php/PICAXE_Editor
Installation des drivers Picaxe : http://a4.fr/wiki/index.php/Installation_des_pilotes_du_câble_AXE027
Erreur PICAXE: « Hardware not found on COM » :
http://a4.fr/wiki/index.php/Erreur_PICAXE:_Hardware_not_found_on_COM!

Composants principaux du circuit et nomenclature





Caractéristiques avancées

Moteur à courant continu :
Rapport de réduction : 136 :1.
Dimensions : Ø 6 x L 19 mm.
500 tr/min à vide à 6 V.
Couple à l'arrêt du moteur : 550 g.cm à 6 V.
Consommation 45 mA à 6 V à vide.
Lien : https://www.pololu.com/product/2358/resources
Plan du moteur en annexe.

Roues du Loupiot : Ø 14 x 4,5 mm.
Lien : https://www.pololu.com/product/2356
Plan des roues en annexe.

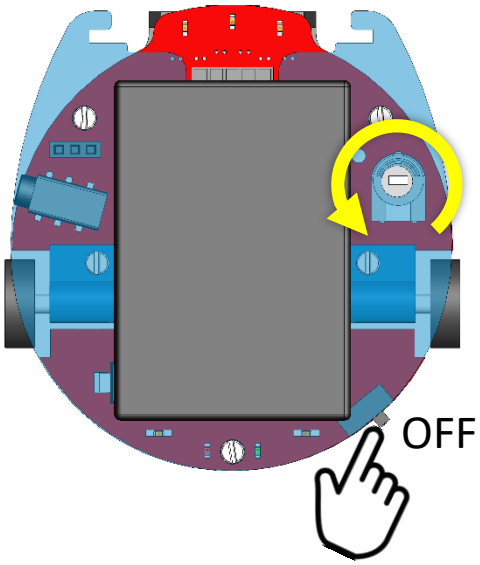
Picaxe 20M2 :
Tension d'alimentation : 3,3 V (provenant du régulateur)
Notice pour les microcontrôleurs Picaxe de type M2 : http://www.picaxe.com/docs/picaxem2.pdf
Schéma des entrées / sorties Picaxe 20M2 en annexe.

Pont en H moteur courant continu :
Lien (datasheet du composant) : https://www.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf

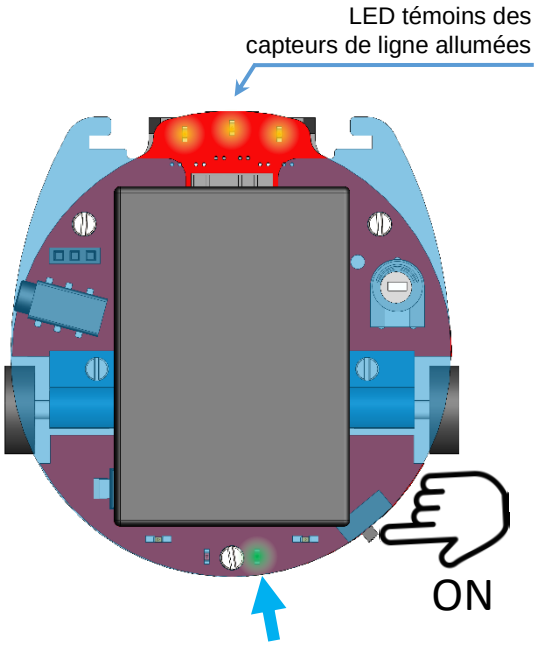
Capteurs de ligne :
Lien (datasheet du composant) : http://www.vishay.com/docs/83752/tcrt1000.pdf
Le robot peut détecter une ligne noire totalement opaque allant jusqu'à 2 mm de largeur si les capteurs sont bien réglés.

Réglage des capteurs infrarouges :

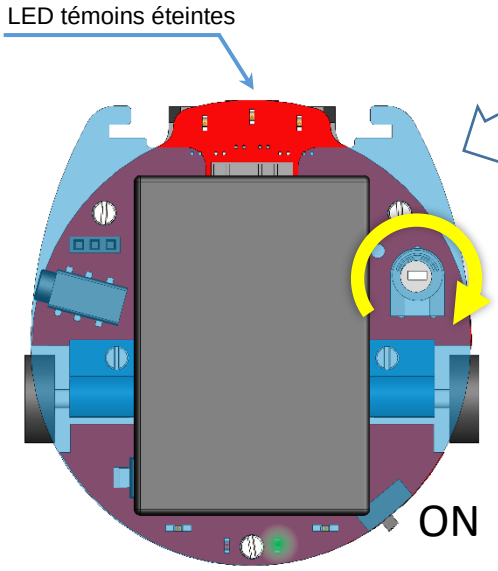
Le robot est livré préréglé. Cependant, les capteurs de ligne (situés à l'avant du robot sous leur LED témoin) sont sensibles au transport et à l'environnement lumineux. Il arrivera donc que vous ayez à les re-régler pour pouvoir tester vos programmes. Pour cela, il suffit de suivre les étapes suivantes :



A - Positionner le robot sur la surface blanche où il va évoluer. Eteindre le robot. A l'aide d'un tournevis plat, tourner le potentiomètre totalement à gauche jusqu'à rencontrer la butée.



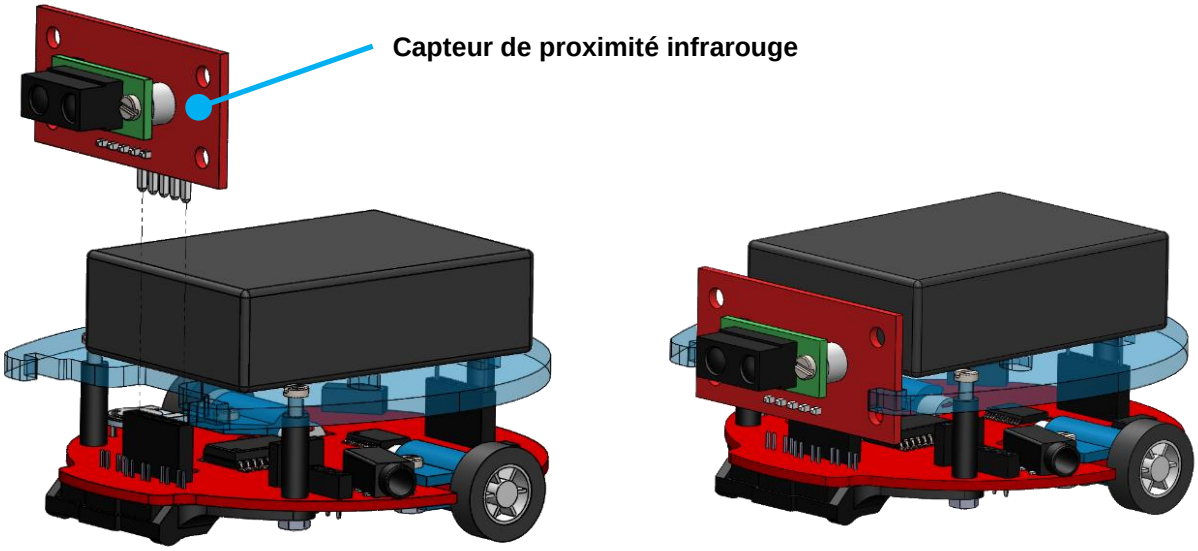
B - Allumer le robot : les 3 LED témoins des capteurs de ligne doivent être allumées.



C - Tourner doucement le potentiomètre dans le sens inverse jusqu'à ce que les 3 LED s'éteignent.

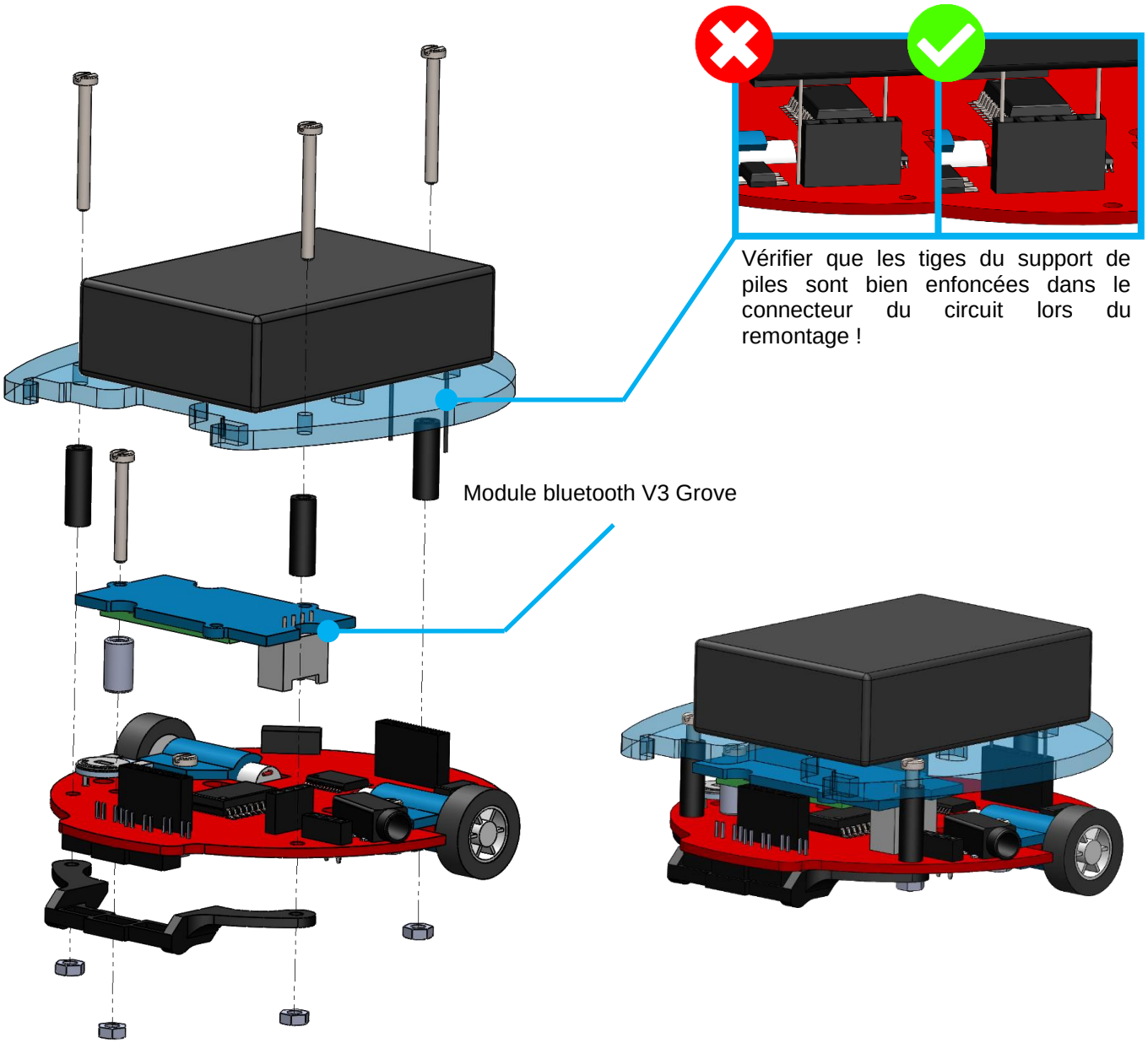
D - Ajuster la sensibilité de détection en tournant le potentiomètre un peu plus à droite afin que les LED ne s'activent pas lorsque l'avant du robot est soulevé de 1 ou 2 mm (cela peut arriver quand il accélère de façon brutale).
Pour tester le bon fonctionnement du réglage, placer les capteurs sur une ligne noire : les LED témoins doivent s'activer.

Montage option détection d’obstacle



Montage option bluetooth

1) Démonter la coque du robot afin de pouvoir positionner le module à son emplacement sur le circuit.
2) Brancher le module Bluetooth au connecteur prévu sur le circuit (il est aussi possible de monter une entretoise à l'autre extrémité du capteur pour améliorer son branchement (voir ci-dessous)).
3) Remonter la coque en prenant garde à ne pas oublier le patin avant venant sous le robot.



Montage option capteur de distance à ultrasons

Le kit contient deux composants : Un télémètre à ultrason mesurant la distance et un élévateur de tension 5V qui l'alimente.

1) Démonter la coque du robot afin de pouvoir positionner le module « stepUp voltage » à l'emplacement prévu sur le circuit.
2) Brancher le module « StepUp voltage » au connecteur prévu sur le circuit.
3) Remonter la coque en prenant garde à ne pas oublier le patin avant venant sous le robot.

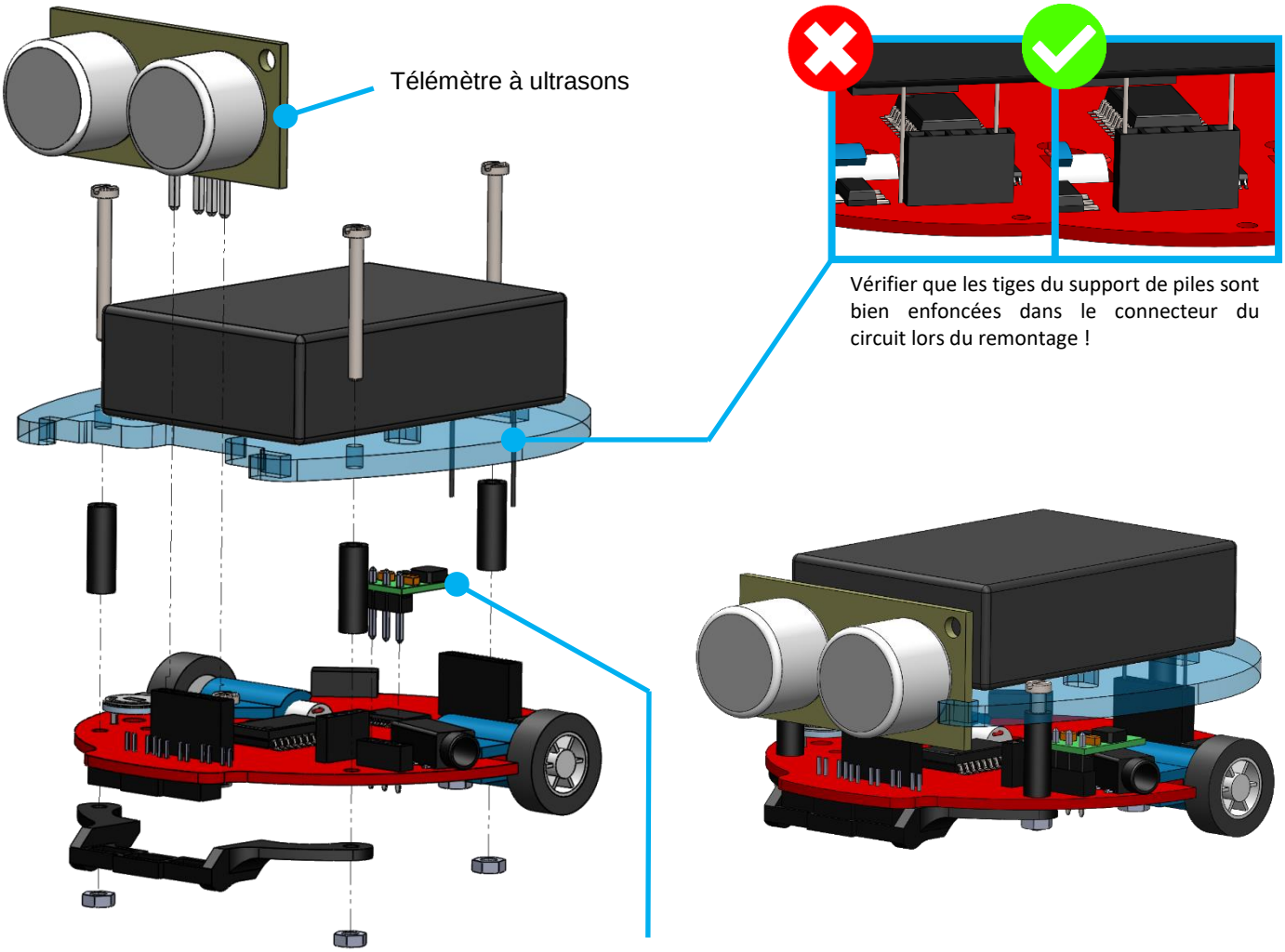


Tableau d'affectation des entrées et sorties

Désignation Blockly	Désignation Picaxe 20M2	Description
Entrées / Capteurs		
Capteur ligne droit	C.4	Capteur infrarouge pour suivi de ligne droit
Capteur ligne centre	C.5	Capteur infrarouge pour suivi de ligne centre
Capteur ligne gauche	C.6	Capteur infrarouge pour suivi de ligne gauche
Lecture batterie	B.5	Lecture de la tension de la batterie
Capteur proximité*	B.7	Selon l'option : Lecture de la distance mesurée par le télémètre à ultrason / Détection d'obstacles à moins de 5 cm (OPTION)
Sorties / Actionneurs		
Témoin gauche / BP	C.7	Témoin programmable orange gauche / Bouton-poussoir
Témoin droit / buzzer	B.0	Témoin programmable orange droit / Buzzer
Témoin alerte	B.4	Témoin programmable rouge
Communication		
BLTH TX*	C.0	Envoi de données Bluetooth (OPTION)
BLTH RX*	B.6	Lecture de données Bluetooth (OPTION)
Contrôle moteur		
PWM moteur droit	C.3	Réglage de la vitesse du moteur droit
Moteur avant droit	C.1	Réglage de la direction du moteur droit
Moteur arrière droit	C.2	Réglage de la direction du moteur droit
PWM moteur gauche	B.1	Réglage de la vitesse du moteur gauche
Moteur avant gauche	B.2	Réglage de la direction du moteur gauche
Moteur arrière gauche	B.3	Réglage de la direction du moteur gauche

Il existe dans le logiciel Blockly un bloc moteur permettant de contrôler directement les deux moteurs du robot Loupiot. Le tableau ci-dessous est donné pour contrôler les moteurs par vous-même avec les blocs de base de Blockly. Attention : dans le premier cas, si on utilise le bloc moteur, les entrées/sorties données ci-dessous ne doivent pas être utilisées car le bloc moteur les contrôle déjà.

Table de vérité - Contrôle des moteurs						
	Moteur Gauche			Moteur Droit		
Broches Utilisées	B.2	B.3	B.1	C.1	C.2	C.3
Marche Avant	Activé	Désactivé	Vitesse PWM	Activé	Désactivé	Vitesse PWM
Marche Arrière	Désactivé	Activé		Désactivé	Activé	
Arrêt Moteur	Activé	Activé		Activé	Activé	

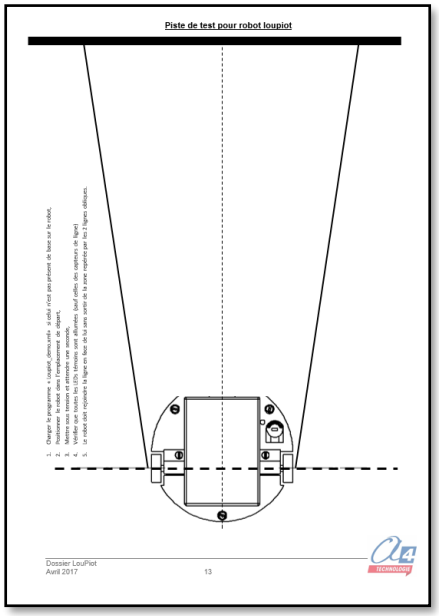
Programme de test pour vérification

Le robot Loupiot est livré préprogrammé (programme « Loupiot_demo_V1.xml »). Pour vérifier qu'il est opérationnel, imprimer la piste de test fournie en annexe et suivre les instructions présentes sur celle-ci.

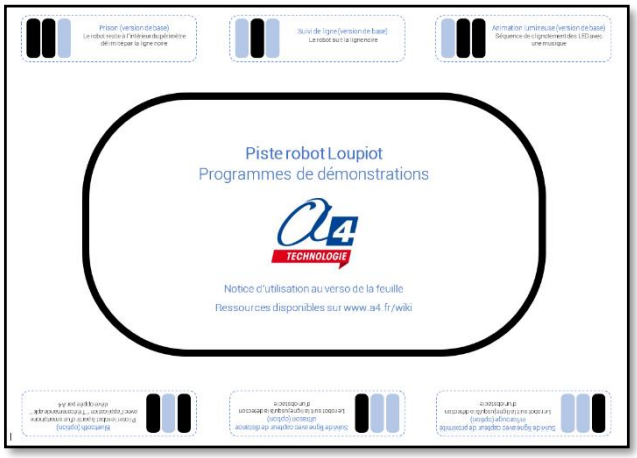
ATTENTION : Veillez à avoir réglé vos capteurs de lignes avant de commencer à utiliser le robot (voir procédure de réglage page 10).

En plus de la piste de test, vous trouverez en annexe une piste pour lancer les autres sous-programmes contenus dans le programme « Loupiot_demo_V1.xml ».
Il s'agit de 6 programmes de démonstration illustrant les capacités du Loupiot (3 pour le Loupiot de base et un pour chaque option). Voir procédure sur la piste.

Si le programme de base a été écrasé, il est possible de le recharger dans le Loupiot. Ce programme est disponible en téléchargement libre sur www.a4.fr.



Piste de test Loupiot



Piste pour programmes de démonstrations
« K-LP-NOTICE-V1 »

Programmation version de base niveau 1

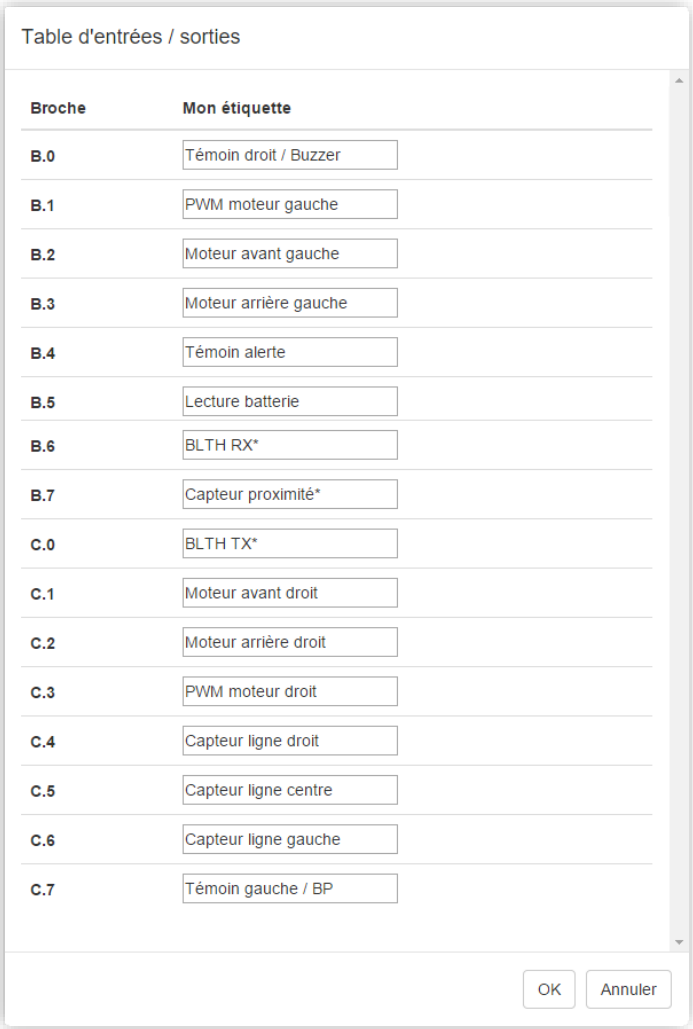
- Découvrir et maîtriser le matériel avec des exemples très simples.
- Appréhender les différentes fonctionnalités du matériel.
- Débuter en programmation.

Introduction

Ce premier niveau permet de découvrir toutes les fonctionnalités de base du robot, au fur et à mesure d'exercices simples. Ce niveau vous permettra également d'introduire les différentes structures de base de la programmation, notamment, celles demandées dans les nouveaux programmes : séquences, boucles, structures conditionnelles (test) et variables.

Nous vous conseillons pour chaque exercice d'essayer d'écrire le programme vous-même avant de regarder la correction et les remarques associées.
Il est nécessaire de partir du modèle de base fourni avec les exercices corrigés du niveau correspondant.
Exemples : pour le programme « LP_N1_A1.xml » (LP pour loupiot, N1 pour niveau 1 et A1 pour sous-niveau 1 programme numéro 1), charger le programme modèle « LP_N1_A.xml ».
Pour le programme « LP_N1_C3.xml », charger le programme modèle « LP_N1_C.xml ».

Au fur et à mesure que vous allez avancer dans les sous-niveaux, la liste de blocs contenus dans les programmes modèles va s'agrandir jusqu'à ce que vous ayez tous les blocs nécessaires pour commencer à programmer le robot par vous-même. (Programme modèle le plus complet : « LP_N1_C.xml ». Note : il ne comprend pas les blocs pour les options !)



Remarque : Les noms contenus dans les listes déroulantes des différents blocs utilisés ont été modifiés pour les programmes du robot Loupiot. Voici ci-contre la liste de nom paramétrée pour tous nos programmes. Voir la documentation de Picaxe Editor pour savoir comment modifier cette liste.

Cette liste est la même pour le robot Loupiot V1 et V2 mais le bouton poussoir (BP) et le buzzer ne sont que pour le Loupiot V2.

Certains blocs proposés dans nos exemples de programmes blockly (blocs pour jouer des sons par ex.) ne sont pas utilisables avec le loupiot V1.

Liste des programmes du niveau 1

Nom du fichier	Description	Objectif
Niveau 1 A (Modèle : LP_N1_A.xml)		
LP_N1_A1.xml	Activer un témoin lumineux	Fonctionnalité matérielle abordée : - Allumage / extinction des témoins lumineux Notions de programmation abordées : - Séquence d'instructions - Temps d'attente - Boucle infinie et Boucle « For » - Activer / désactiver une sortie
LP_N1_A2.xml	Activer / désactiver un témoin lumineux	
LP_N1_A3.xml	Faire clignoter un témoin lumineux	
LP_N1_A4.xml	Faire clignoter deux témoins lumineux en alternance	
LP_N1_A5.xml	Faire clignoter un témoin lumineux 5 fois (première méthode)	
LP_N1_A6.xml	Faire clignoter un témoin lumineux 5 fois (deuxième méthode)	
Niveau 1 B (Modèle : LP_N1_B.xml)		
LP_N1_B1.xml	Avancer	Fonctionnalité matérielle abordée : - Gestion des moteurs
LP_N1_B2.xml	Avancer puis s'arrêter	
LP_N1_B3.xml	Tourner à droite puis à gauche	
LP_N1_B4.xml	Tourner en rond	
LP_N1_B5.xml	Mouvement répété	
LP_N1_B6.xml	Accélération brutale	
Niveau 1 C (Modèle : LP_N1_C.xml)		
LP_N1_C1.xml	Recopier l'état d'une entrée	Fonctionnalité matérielle abordée : - Lecture des entrées du robot (capteur de ligne / lecture batterie) Notions de programmation abordées : - Debug - Structures conditionnelles
LP_N1_C2.xml	Recopier l'état de plusieurs entrées	
LP_N1_C3.xml	Avancer jusqu'à la ligne 1 (une condition)	
LP_N1_C4.xml	Avancer jusqu'à la ligne 2 (conditions imbriquées)	
LP_N1_C5.xml	Lecture batterie / Debug	

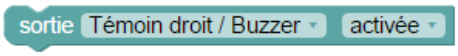
Exercice niveau 1 – A1 : Activer un témoin lumineux

Fichier modèle : LP_N1_A.xml

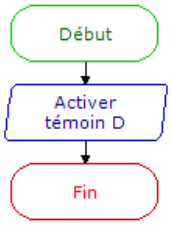
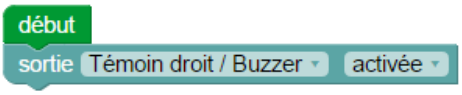
Objectif : allumer le témoin droit du robot.

Notion(s) abordée(s) : activation d'une sortie.

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A1_Organigramme.plf	Fichier Blockly : LP_N1_A1.xml

Remarque(s) :

- Un programme téléchargé écrase le précédent.
- Le programme démarre à partir de l'instruction « Début » dès la fin du téléchargement ou dès la mise sous tension du robot.
- Par défaut, toutes les sorties de la carte de pilotage du robot sont désactivées. L'activation d'une sortie reste valide tant qu'une instruction de désactivation n'est pas exécutée, même quand le programme est terminé.

Exercice niveau 1 – A2 : Activer / désactiver un témoin lumineux

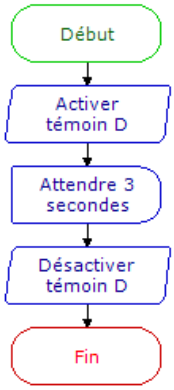
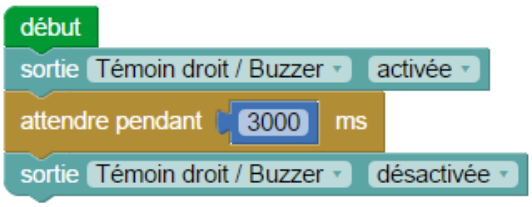
Fichier modèle : LP_N1_A.xml

Objectif : allumer le témoin droit pendant 3 secondes puis l'éteindre.

Notion(s) abordée(s) : temps d'attente.

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A2_Organigramme.plf	Fichier Blockly : LP_N1_A2.xml

Remarque(s) :

- L'instruction « Attendre » permet d'introduire un temps d'attente avant l'exécution de l'instruction qui la suit.
- En programmation par bloc, la dernière instruction exécutée marque la fin du programme.

Exercice niveau 1 – A3 : Faire clignoter un témoin lumineux

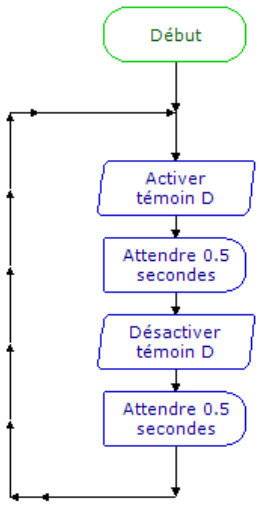
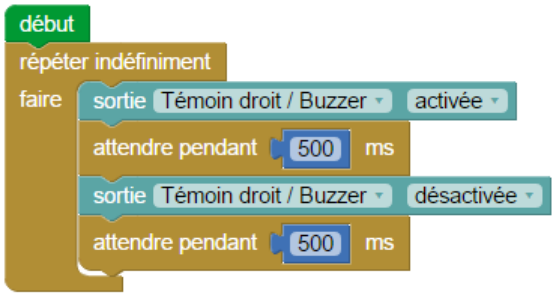
Fichier modèle : LP_N1_A.xml

Objectif : faire clignoter le témoin droit toutes les secondes.

Notion(s) abordée(s) : boucle infinie.

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A3_Organigramme.plf	Fichier Blockly : LP_N1_A3.xml

Remarque(s) :

- Les instructions sont exécutées de manière séquentielle : les unes à la suite des autres.
  Le microcontrôleur exécute plusieurs milliers d'instructions par seconde. Il est nécessaire dans cet exemple de placer des temps d'attente pour voir les changements d'état de la LED. Un seul temps d'attente ne suffit pas pour rendre le clignotement perceptible.
- La séquence d'instructions englobée dans la boucle « répéter indéfiniment » est exécutée en permanence. Quand on arrive à la dernière instruction (dernier bloc) englobée dans la boucle « répéter indéfiniment », le programme reprend à partir de la première instruction contenue dans la boucle.

Exercice niveau 1 – A4 : Faire clignoter deux témoins lumineux en alternance

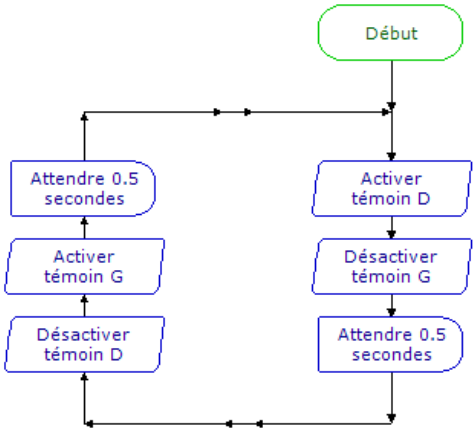
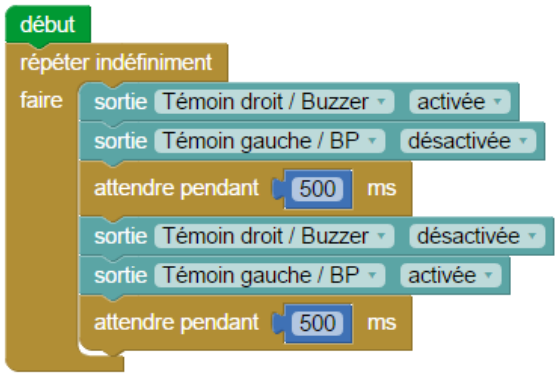
Fichier modèle : LP_N1_A.xml

Objectif : faire clignoter de manière alternée les témoins droit et gauche toutes les secondes.

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A4_Organigramme.plf	Fichier Blockly : LP_N1_A4.xml

Remarque(s) :

Exercice niveau 1 – A5 : Faire clignoter un témoin lumineux 5 fois - Première méthode

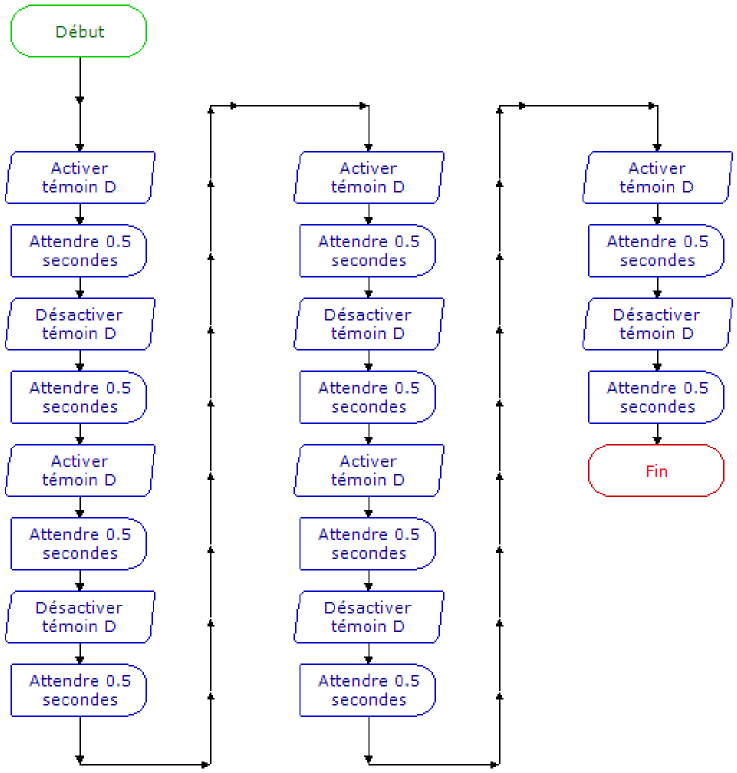
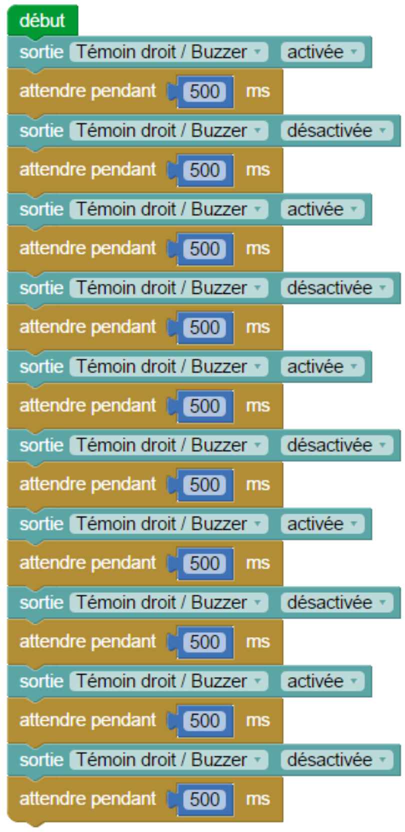
Fichier modèle : LP_N1_A.xml

Objectif : faire clignoter le témoin droit 5 fois avec une période de 1 seconde.

Notion(s) abordée(s) : répéter n fois une séquence.

Instruction(s) utilisée(s) :

Correction :

<table>
<tr><td>Organigramme</td><td>Blocs</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Fichier organigramme PE6 : LP_N1_A5_Organigramme.plf</td><td>Fichier Blockly : LP_N1_A5.xml</td></tr>
</table>

Remarque(s) :

- La répétition multiple d'une même séquence d'instructions peut être fastidieuse à rédiger. Une instruction plus appropriée permet de simplifier l'écriture du programme (voir exercice suivant).

Exercice niveau 1 – A6 : Faire clignoter un témoin lumineux 5 fois – Deuxième méthode

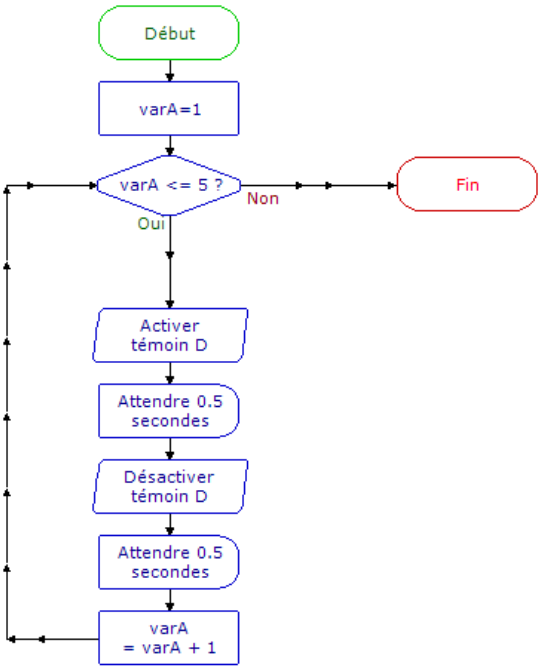
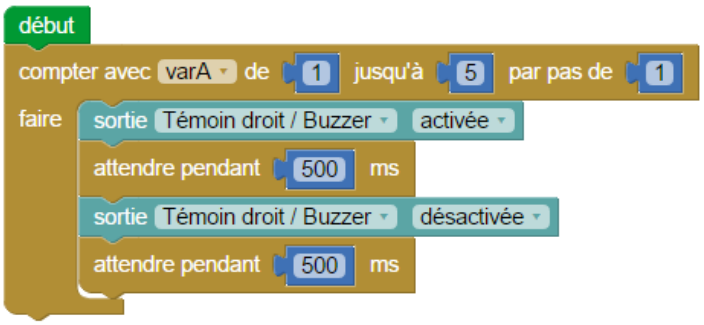
Fichier modèle : LP_N1_A.xml

Objectif : faire clignoter le témoin droit 5 fois avec une période de 1 seconde.

Notion(s) abordée(s) : boucle de type répéter n fois (boucle FOR).

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A6_Organigramme.plf	Fichier Blockly : LP_N1_A6.xml

Remarque(s) :

- Le bloc « compter avec varA de 1 jusqu'à 5 par pas de 1 » englobe une séquence qui est répétée 5 fois.
- « VarA » est une variable qui vaut zéro au lancement du programme puis qui prend successivement les valeurs 1, 2, …, 5 avant que le programme ne s'arrête. Elle peut être utilisée en lecture pour les blocs contenus dans la boucle mais ne doit pas être modifiée durant celle-ci.

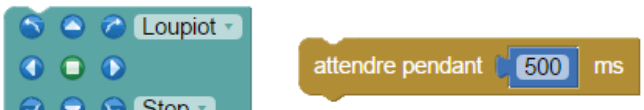
Exercice niveau 1 – B1 : Avancer

Fichier modèle : LP_N1_B.xml

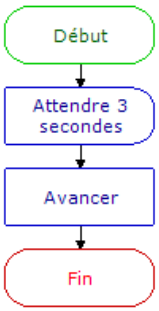
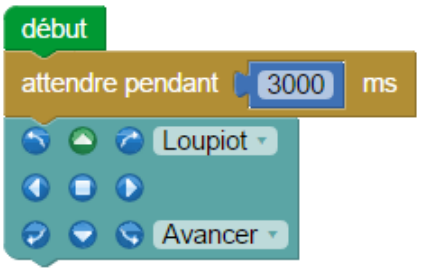
Objectif : faire avancer le robot au bout de trois secondes après la mise sous tension.

Notion(s) abordée(s) : utiliser l'instruction spéciale pour animer le robot (bloc moteur Loupiot)

Instruction(s) utilisée(s) :



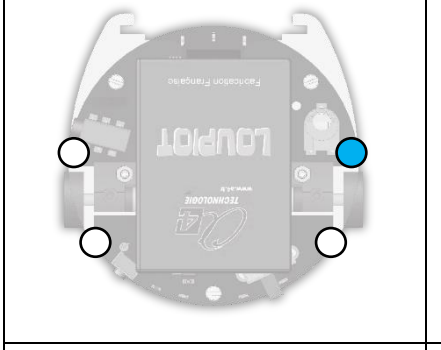
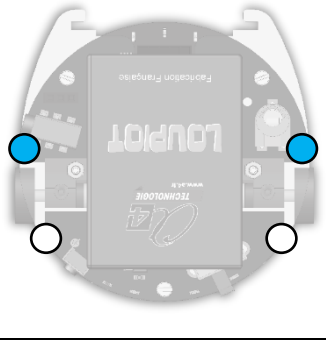
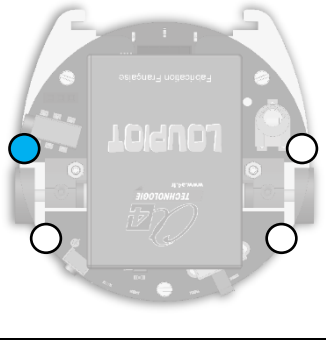
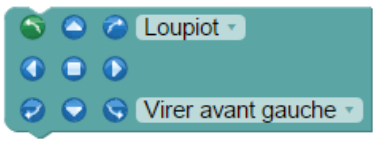
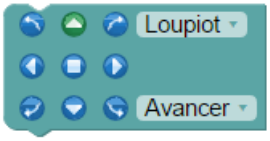
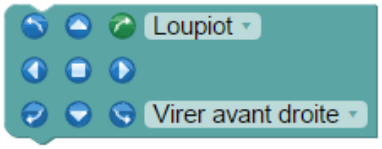
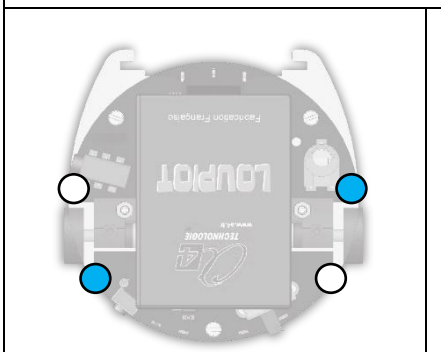
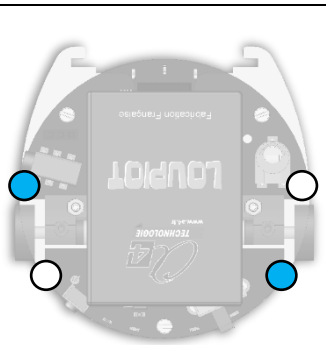
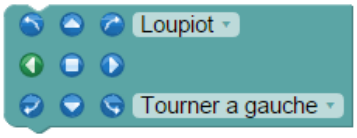
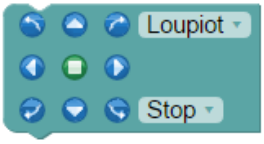
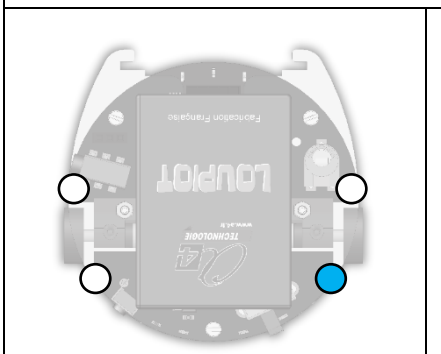
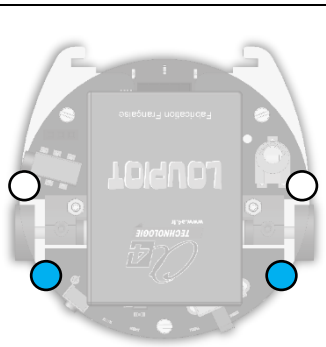
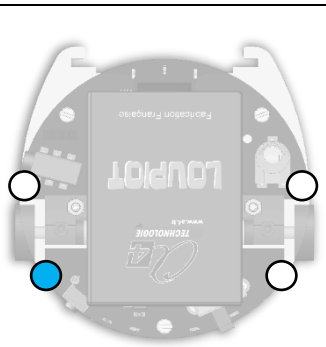
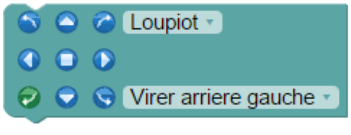
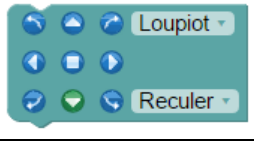
Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_B1_Organigramme.plf	Fichier Blockly : LP_N1_B1.xml

Remarque(s) :

- L'instruction « Avancer » reste active jusqu'à la mise hors tension du Loupiot car comme nous l'avons vu précédemment, les entrées/sorties gardent leur état même à la fin du programme.
- Des témoins lumineux bleus s'activent pour indiquer le sens de rotation des moteurs (voir tableau page suivante).
- L'attente de 3 secondes permet de débrancher le robot avant que celui-ci ne démarre lorsque le programme a fini de se télécharger. Cette attente sera placée dans tous les programmes N1-B utilisant les moteurs.

Ce tableau présente l'état des LED témoins du sens de rotation des moteurs droit et gauche en fonction de la direction paramétrée sur le bloc moteur Loupiot. Note : il est possible que les LED soient activées mais que le robot n'avance pas : les LED représentent le sens de rotation du moteur et non la vitesse.

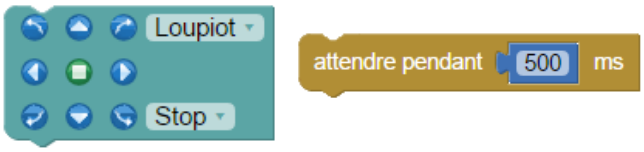
Exercice niveau 1 – B2 : Avancer puis s'arrêter

Fichier modèle : LP_N1_B.xml

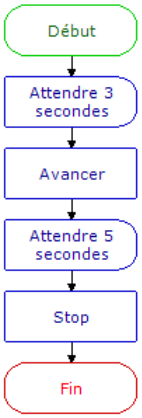
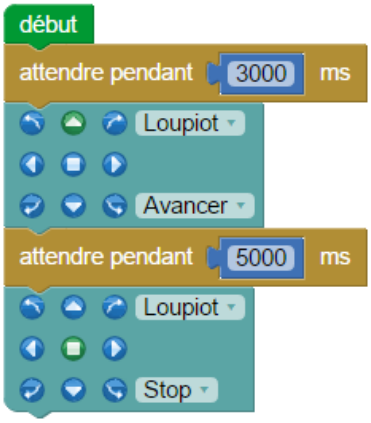
Objectif : Au bout de 3 secondes, faire avancer le robot pendant 5 secondes puis l'arrêter.

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_B2_Organigramme.plf	Fichier Blockly : LP_N1_B2.xml

Remarque(s) :

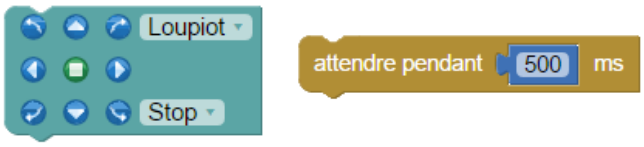
Exercice niveau 1 – B3 : Tourner à droite puis à gauche

Fichier modèle : LP_N1_B.xml

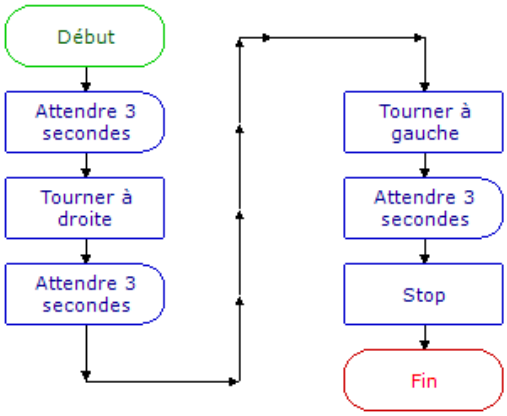
Objectif : Au bout de 3 secondes, faire tourner le robot sur lui-même à droite pendant 3 secondes, puis à gauche pendant 3 secondes, puis l'arrêter.

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_B3_Organigramme.plf	Fichier Blockly : LP_N1_B3.xml

Remarque(s) :

- L'instruction « Tourner » inverse le sens des moteurs ce qui a pour conséquence de faire tourner le robot sur lui-même.
- L'instruction « Virer » anime une roue à la fois. Le robot décrit un arc de cercle autour de la roue opposée (voir exemple suivant).

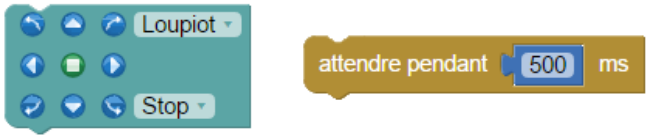
Exercice niveau 1 – B4 : Tourner en rond

Fichier modèle : LP_N1_B.xml

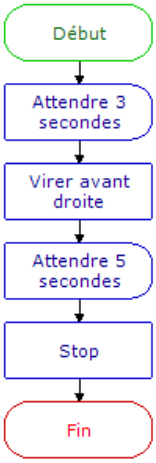
Objectif : faire tourner le robot en rond pendant 5 secondes puis l'arrêter.

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_B4_Organigramme.plf	Fichier Blockly : LP_N1_B4.xml

Remarque(s) :

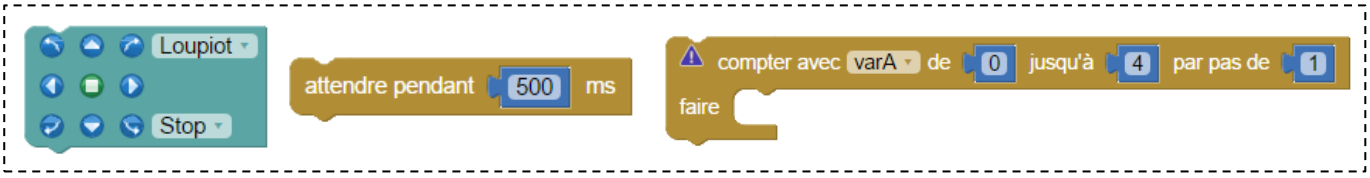
Exercice niveau 1 – B5 : Mouvement répété

Fichier modèle : LP_N1_B.xml

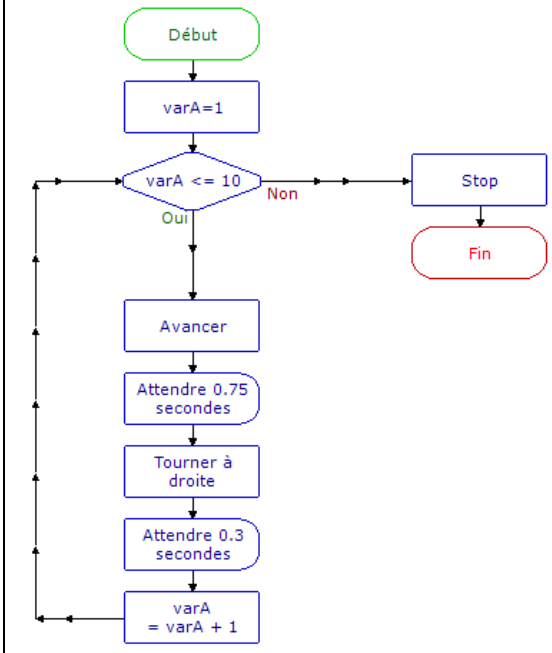
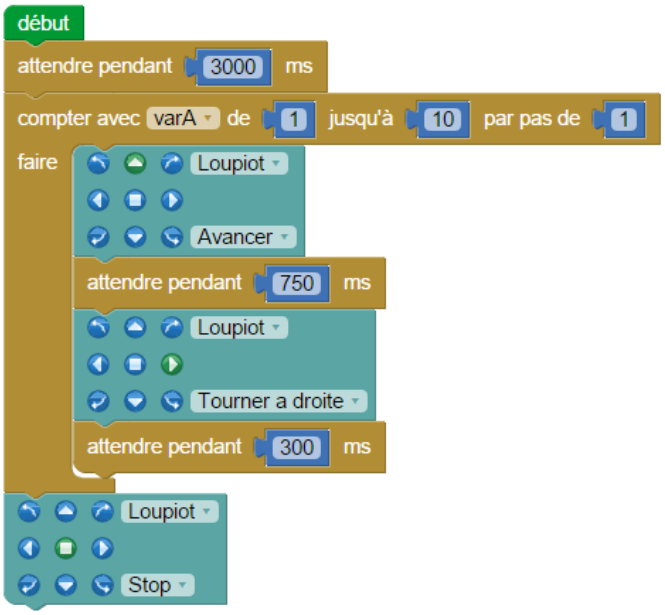
Objectif : répéter 10 fois l'action suivante : avancer puis tourner à droite afin d'effectuer un polygone.

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_B5_Organigramme.plf	Fichier Blockly : LP_N1_B5.xml

Remarque(s) :

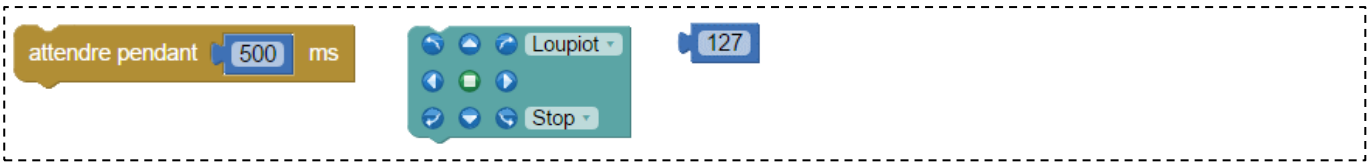
Exercice niveau 1 – B6 : Accélération brutale

Fichier modèle : LP_N1_B.xml

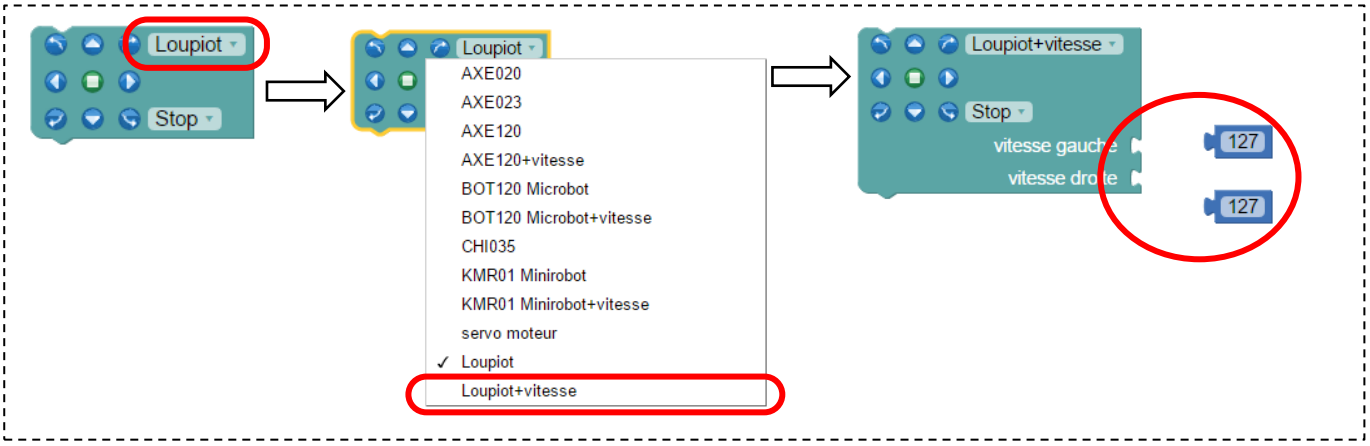
Objectif : Au bout de 3 secondes, faire avancer le robot à 50% de sa vitesse pendant 3 secondes, puis à 100% pendant 3 secondes avant de l'arrêter.

Notion(s) abordée(s) : ajouter le contrôle de la vitesse au bloc de contrôle des moteurs.

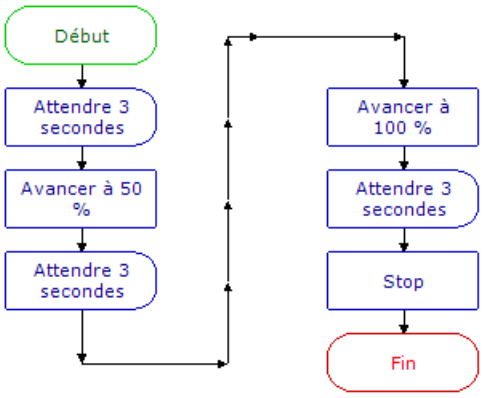
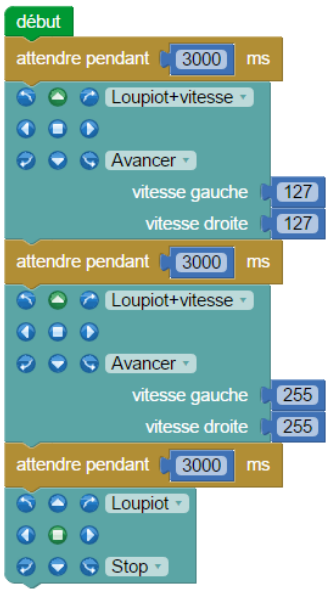
Instruction(s) utilisée(s) :



Ajouter l'option vitesse sur le bloc de contrôle des moteurs :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_B6_Organigramme.plf	Fichier Blockly : LP_N1_B6.xml

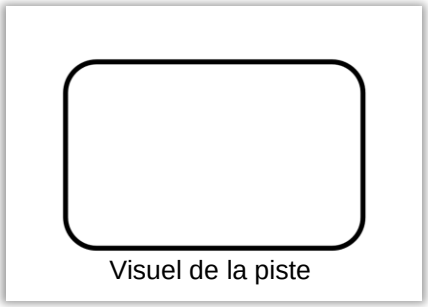
Remarque(s) :

- Par défaut, la vitesse du Loupiot est paramétrée à 127 (50% de la vitesse max.), 255 correspond à la vitesse maximum du robot.
- Le robot ne commence à avancer qu'à partir d'une consigne de vitesse d'environ 30-40 (en fonction de l'état des batteries).

Exercice niveau 1 – C1 : Recopier l'état d'une entrée

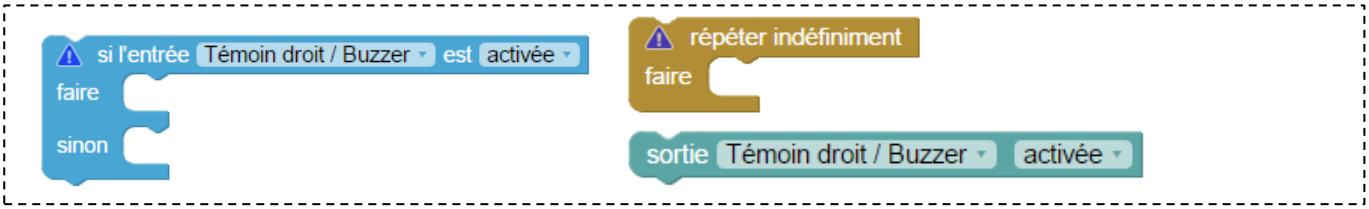
Fichier modèle : LP_N1_C.xml

Objectif : recopier l'état du capteur de ligne centre sur le témoin alerte. Note : Imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes du niveau 2B pour tester ce code.

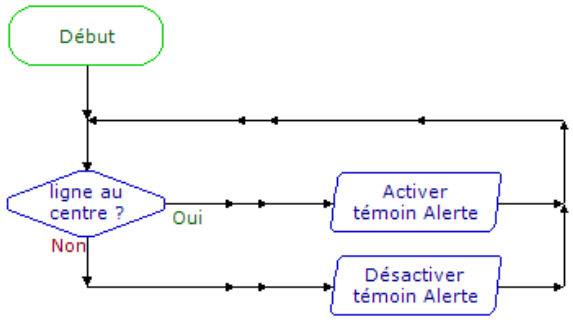
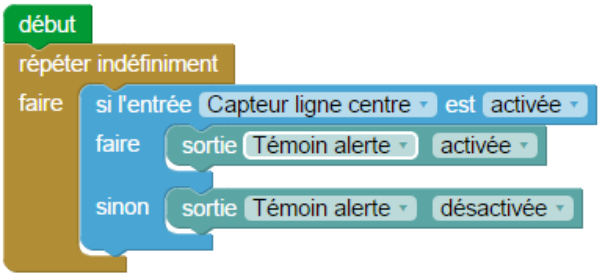

Visuel de la piste

Notion(s) abordée(s) : structure conditionnelle / Lecture de l'état d'une entrée du robot.

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C1_Organigramme.plf	Fichier Blockly : LP_N1_C1.xml

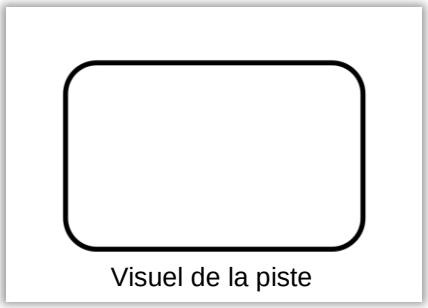
Remarque(s) :

- La boucle infinie est nécessaire pour interroger en permanence l'état du capteur de ligne centre.
- Si une ligne se trouve sous un capteur de ligne, l'entrée est activée. Quand on interroge ce capteur avec le bloc « si l'entrée … est activée » la condition est donc validée.
- Attention : régler correctement les capteurs de ligne avant de commencer les programmes de niveau 1C.

Exercice niveau 1 – C2 : Recopier l'état de plusieurs entrées

Fichier modèle : LP_N1_C.xml

Objectif : recopier l'état des capteurs de ligne droit, centre et gauche respectivement sur les témoins lumineux droit, alerte et gauche. Note : imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes du niveau 2B pour tester ce code.

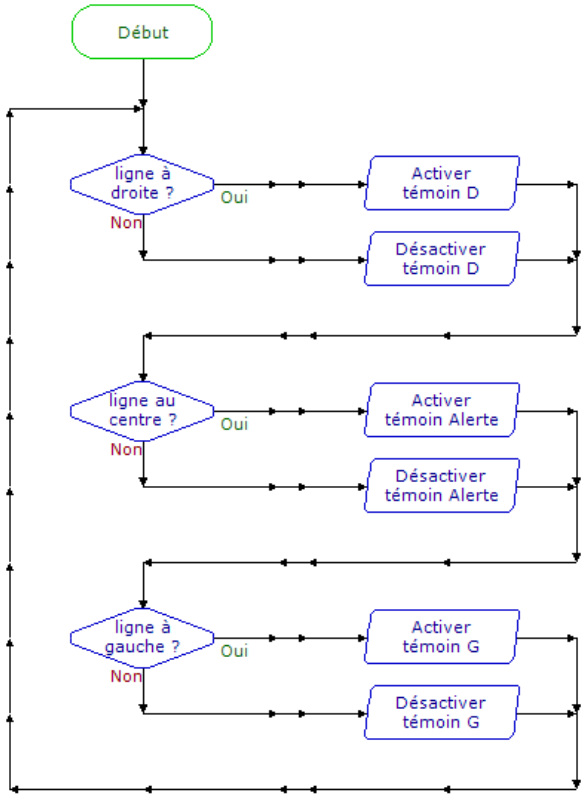
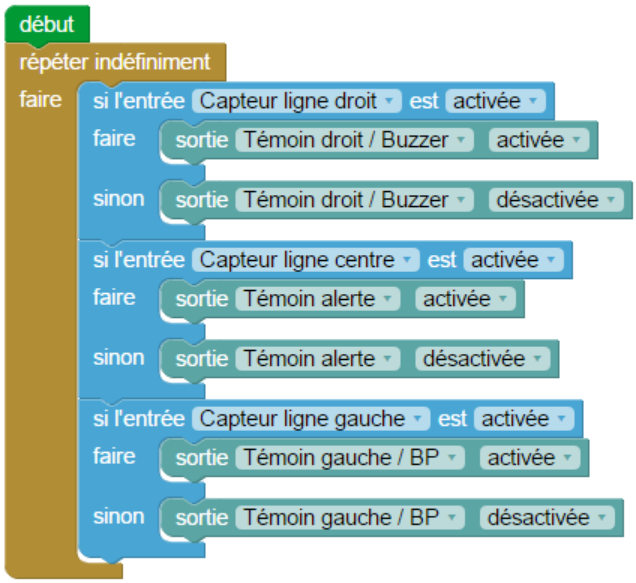


Visuel de la piste

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C2_Organigramme.plf	Fichier Blockly : LP_N1_C2.xml

Remarque(s) :

Exercice niveau 1 – C3 : Avancer jusqu'à la ligne 1

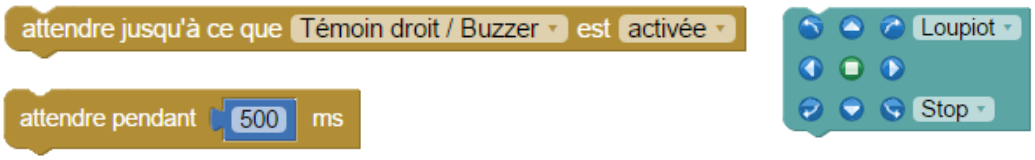
Fichier modèle : LP_N1_C.xml

Objectif : Au bout de 3 secondes, faire avancer le robot jusqu'à détecter une ligne noire sous le capteur de ligne centre.
Note : imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes du niveau 2B pour tester ce code.

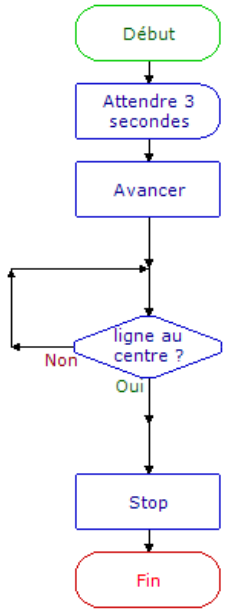
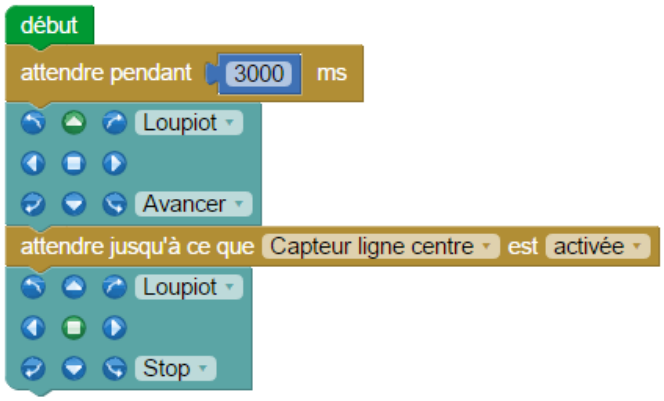
Visuel de la piste

Notion(s) abordée(s) : attendre qu'une condition soit validée.

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C3_Organigramme.plf	Fichier Blockly : LP_N1_C3.xml

Remarque(s) :

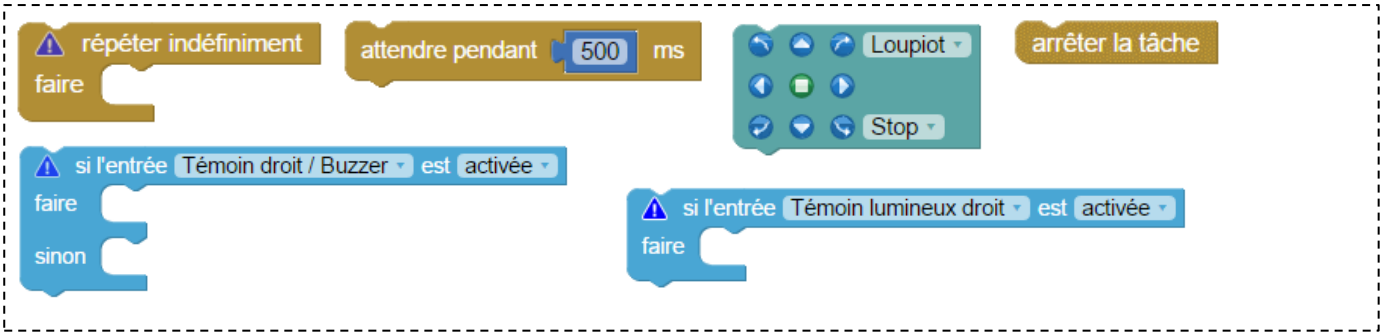
Exercice niveau 1 – C4 : Avancer jusqu'à la ligne 2

Fichier modèle : LP_N1_C.xml

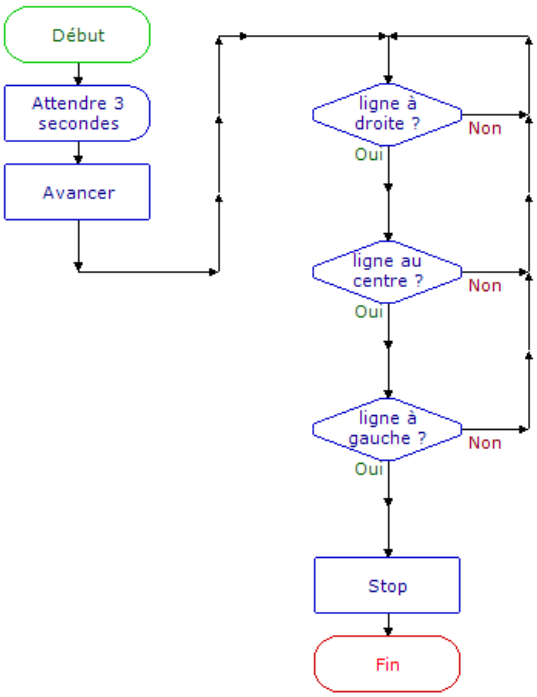
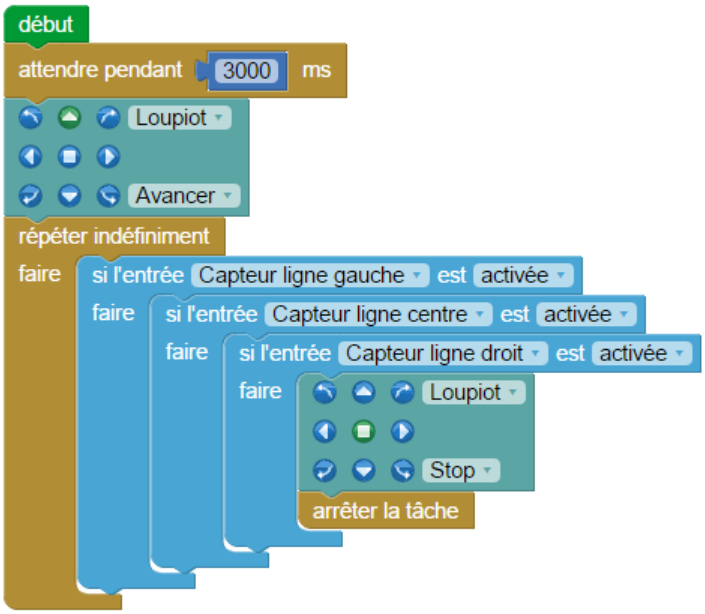
Objectif : Au bout de 3 secondes, faire avancer le robot jusqu'à détecter une ligne noire sous les trois capteurs de ligne.

Notion(s) abordée(s) : conditions imbriquées.

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C4_Organigramme.plf	Fichier Blockly : LP_N1_C4.xml

Remarque(s) :

- Si une seule condition vient à échouer, on reprend au début de la boucle infinie.
- Le bloc « arrêter la tâche » sert à forcer la fin du programme. En effet, si les 3 conditions sont remplies, le robot s'arrête mais le programme reprend au début de la boucle infinie après l'instruction « Stop » ce qui ne sert plus à rien dans notre cas.

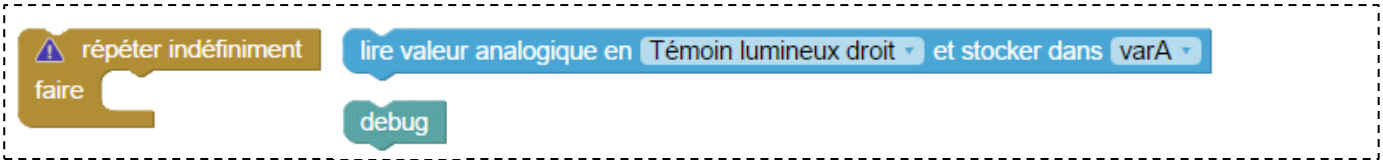
Exercice niveau 1 – C5 : Lecture batterie / debug

Fichier modèle : LP_N1_C.xml

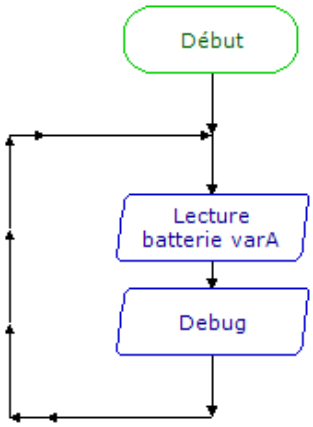
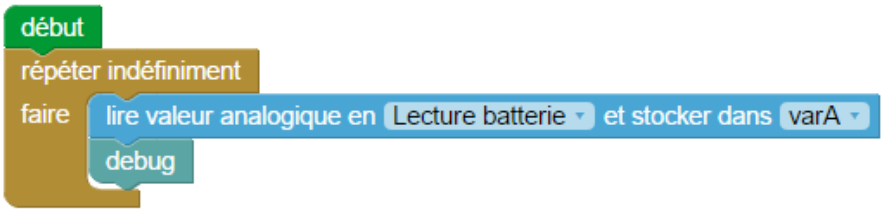
Objectif : lire le niveau de tension de la batterie du Loupiot et l'afficher.

Notion(s) abordée(s) : lire une valeur analogique / afficher une variable sur l'ordinateur / stocker une valeur dans une variable.

Instruction(s) utilisée(s) :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C5_Organigramme.plf	Fichier Blockly : LP_N1_C5.xml

Remarque(s) :

- La valeur de la batterie est stockée dans une variable allant de 0 à 255.
  255 correspond à 6,6 V et 0 à 0 V aux bornes de la batterie. On peut donc retrouver la tension actuelle de la batterie à partir d'un produit en croix.
- Pour savoir comment fonctionne le Debug, veuillez consulter la documentation du logiciel (page 24).
  Lien : http://www.a4telechargement.fr/Logiciels_Programmation/Blockly_Manuel-utilisateur-FR-10.2016.pdf
- Attention : le câble doit rester branché au robot pour qu'il puisse communiquer avec l'ordinateur lors du debug.
- Attention : le debug est un des rares blocs dont le temps d'exécution est non négligeable.
  Prenez garde à prendre ce temps en considération si vous utilisez cette instruction dans vos programmes.

- Cas concrets d'utilisation des différentes fonctionnalités du matériel.
- Exemples d'application à difficulté croissante.

Introduction

Le niveau 2 permet d'utiliser de manière plus concrète toutes les notions de programmation abordées dans le niveau 1. De nouveaux blocs et de nouvelles notions de programmation (pour optimiser le code) sont introduits.

Liste des programmes du niveau 2

Nom du fichier	Description	Objectif
Niveau 2 A (Modèle : LP_N2_A.xml)		
LP_N2_A1.xml	Chenillard	Fonctionnalité matérielle abordée : - Utilisation concrète des témoins lumineux Notions de programmation abordées : - Utilisation approfondie des variables
LP_N2_A2.xml	Clignotement en fonction de la position d'une ligne	
LP_N2_A3.xml	Accélération / Décélération du clignotement d'une LED	
Niveau 2 B (Modèle : LP_N2_B.xml)		
LP_N2_B1.xml	Suivi d'une ligne fine	Fonctionnalité matérielle abordée : - Utilisation concrète de la gestion des moteurs Notions de programmation abordées : - Procédures
LP_N2_B2.xml	Suivi d'une ligne large	
LP_N2_B3.xml	Accélération / décélération	
LP_N2_B4.xml	Accélération / décélération avec procédure	
Niveau 2 C (Modèle : LP_N2_C.xml)		
LP_N2_C1.xml	Détecter 3 fois un code	Fonctionnalité matérielle abordée : - Utilisation concrète des capteurs du robot Notions de programmation abordées : - Opérations booléennes
LP_N2_C2.xml	Aller – retour sur une ligne	
LP_N2_C3.xml	Prison	

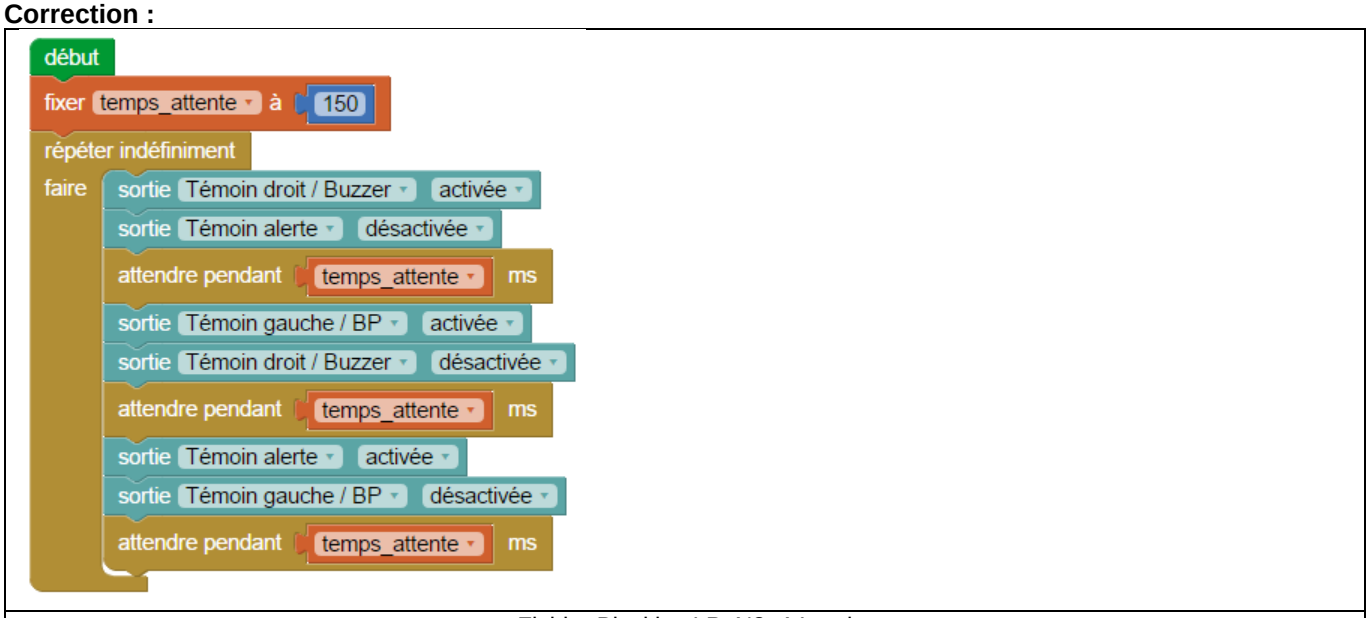
Exercice niveau 2 – A1 : Chenillard

Fichier modèle : LP_N2_A.xml

Objectif : Faire clignoter 3 LED les unes à la suite des autres.

Instruction(s) utilisée(s) :

Correction :



Fichier Blockly : LP_N2_A1.xml

Remarque(s) :

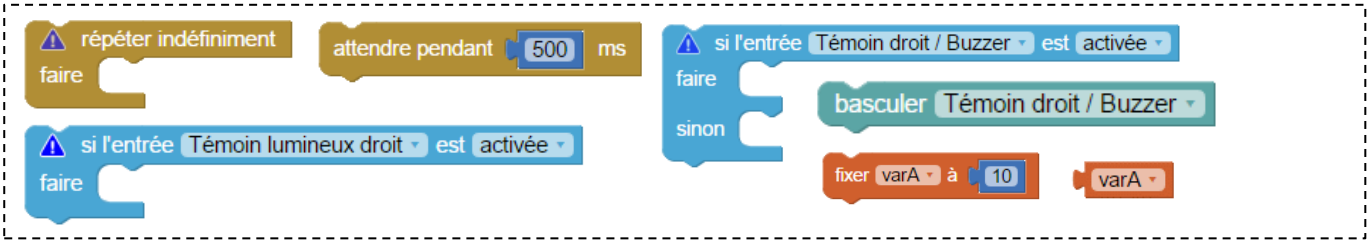
- On utilise ici une variable pour fixer le même temps à toutes les attentes. Dans ce cas, il n'y a qu'un seul champ de texte à modifier et dans le cas contraire 3.
- Modifier la valeur de la variable du temps d'attente pour voir comment le programme réagit.
- Pour modifier le nom d'une variable ou en créer une nouvelle, veuillez lire la documentation du logiciel (page 30).
  Lien : http://www.a4telechargement.fr/Logiciels_Programmation/Blockly_Manuel-utilisateur-FR-10.2016.pdf

Exercice niveau 2 – A2 : Clignotement en fonction de la position d'une ligne

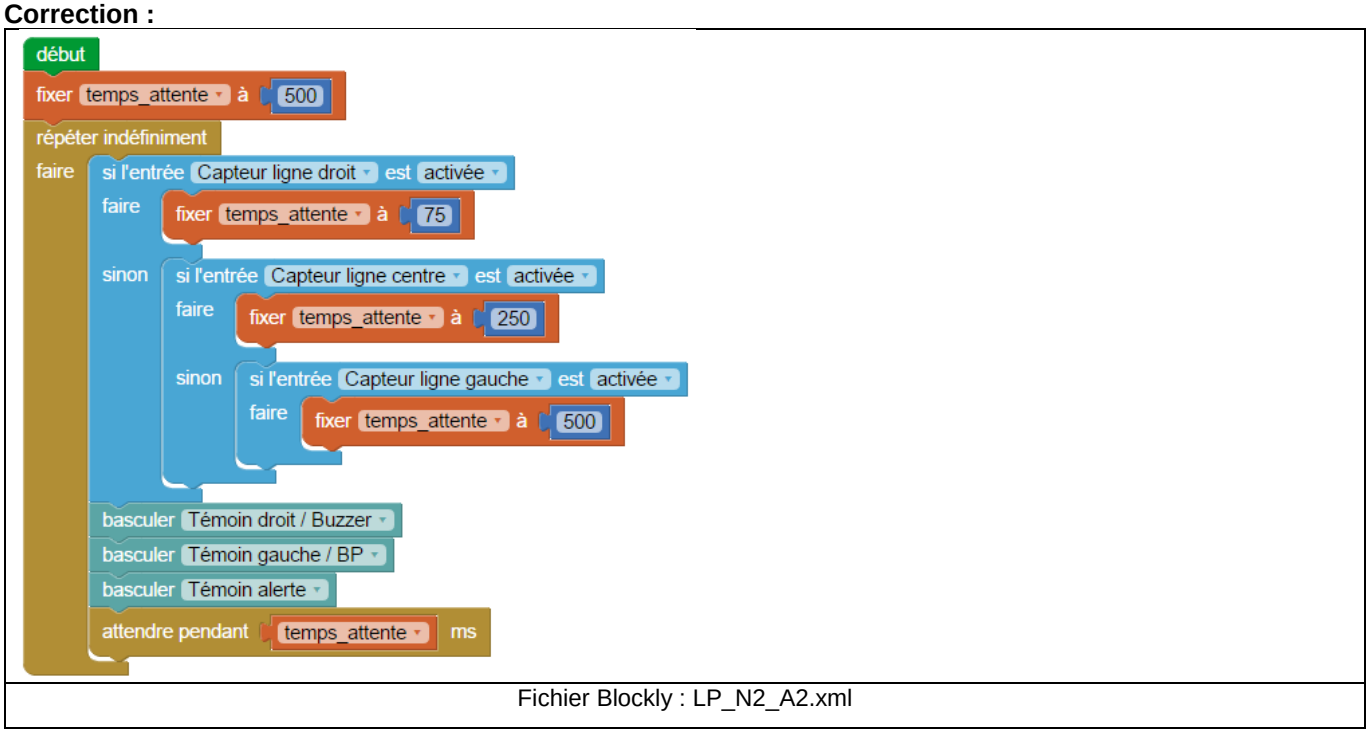
Fichier modèle : LP_N2_A.xml

Objectif : Faire clignoter les 3 témoins lumineux à une vitesse variant en fonction de la position d'une ligne noire par rapport aux capteurs de ligne (rapide à droite et lent à gauche).

Instruction(s) utilisée(s) :



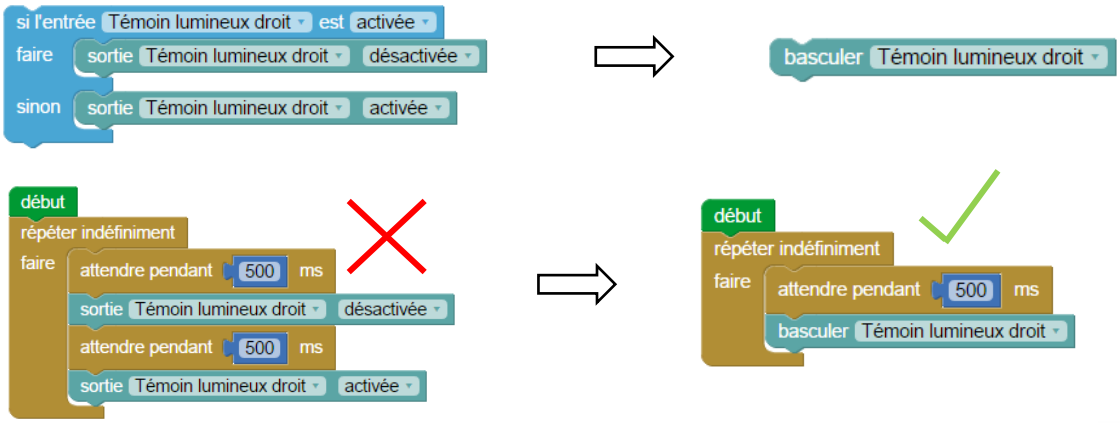
Correction :



Fichier Blockly : LP_N2_A2.xml

Remarque(s) :

- L'instruction « basculer » inverse l'état de la sortie : si elle est activée, on la désactive et inversement. Cela permet de rendre le code plus simple comme indiqué ci-dessous :

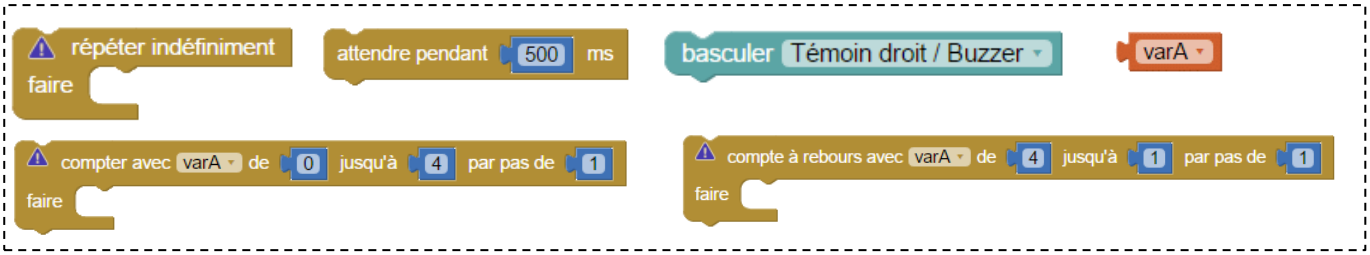


Exercice niveau 2 – A3 : Accélération / décélération du clignotement d'une LED

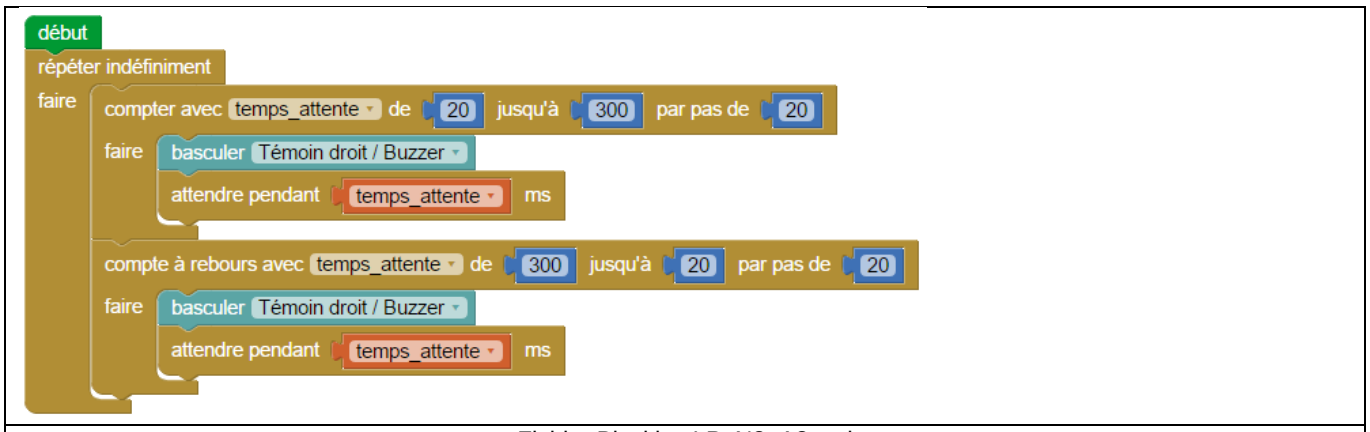
Fichier modèle : LP_N2_A.xml

Objectif : Accélérer puis décélérer le clignotement d'une LED en jouant sur le temps d'attente.

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_A3.xml

Remarque :

- La variable utilisée dans la boucle « compter … » prendra différentes valeurs correspondant au compte à rebours (300, 280, …. 40, 20, 0).

Exercice niveau 2 – B1 : Suivi d'une ligne fine

Fichier modèle : LP_N2_B.xml

Objectif : Suivre une ligne faisant la largeur d'un capteur de ligne
(Note : Imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes pour tester ce code).

Instruction(s) utilisée(s) :

Visuel de la piste

répéter indéfiniment
faire

si l'entrée Témoin droit / Buzzer est activée
faire
sinon

attendre pendant 500 ms

Loupiot
Stop

si l'entrée Témoin lumineux droit est activée
faire

Correction :

début
attendre pendant 3000 ms
Loupiot
Avancer
répéter indéfiniment
faire
si l'entrée Capteur ligne droit est activée
faire
Loupiot
Virer avant droite
sinon
si l'entrée Capteur ligne centre est activée
faire
Loupiot
Avancer
sinon
si l'entrée Capteur ligne gauche est activée
faire
Loupiot
Virer avant gauche

Fichier Blockly : LP_N2_B1.xml

Remarque(s) :

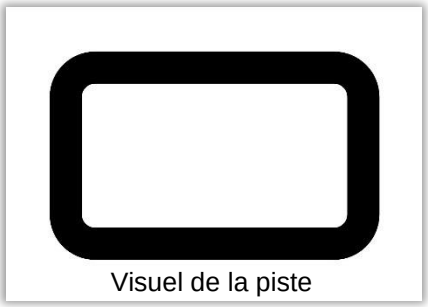
- Pensez à bien régler vos capteurs de ligne lors du test de ces programmes (voir la procédure ci-dessus).
- Pour tous les programmes utilisant les moteurs du robot, nous placerons une attente de 3 secondes avant de lancer les moteurs. Cela évite que le robot démarre directement après le transfert du programme et arrache le câble de programmation.

Exercice niveau 2 – B2 : Suivi d'une ligne large

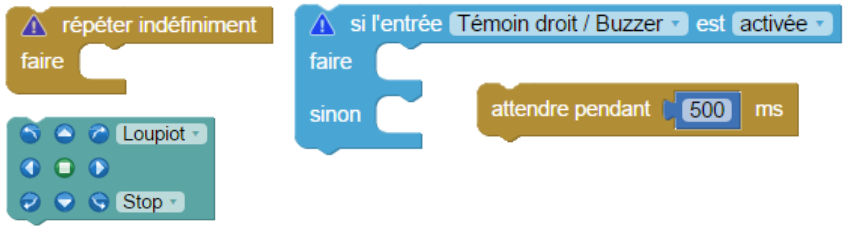
Fichier modèle : LP_N2_B.xml

Objectif : Suivre une ligne faisant la largeur des 3 capteurs de ligne (Note : Imprimer la piste "Piste_suivi_ligne_type2" donnée avec les programmes pour tester ce code).

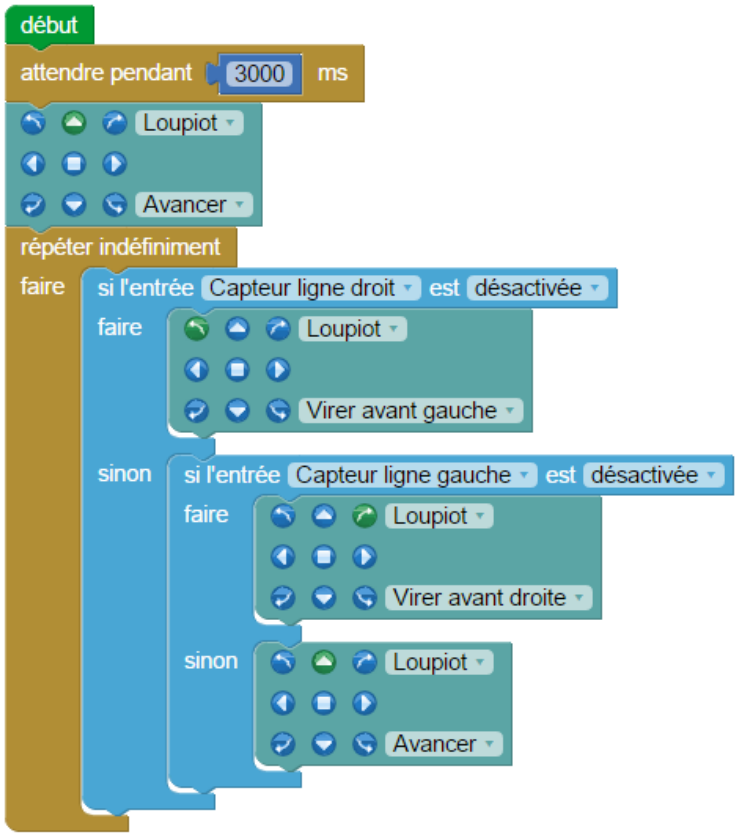
Instruction(s) utilisée(s) :



Visuel de la piste



Correction :



Fichier Blockly : LP_N2_B2.xml

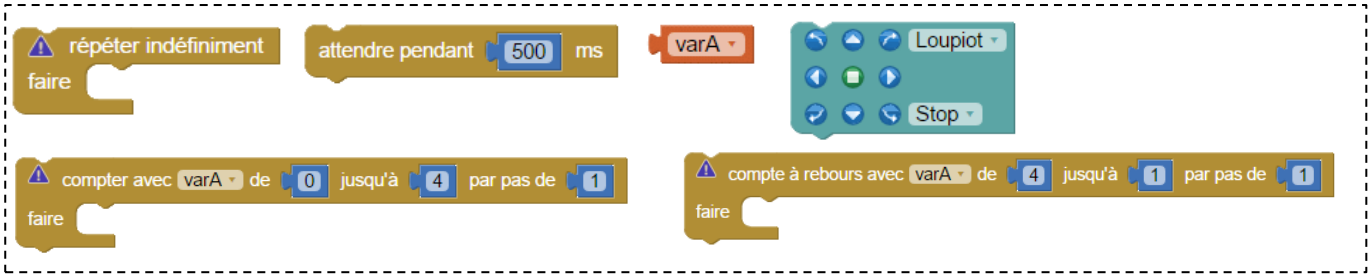
Remarque(s) :

Exercice niveau 2 – B3 : Accélération / décélération

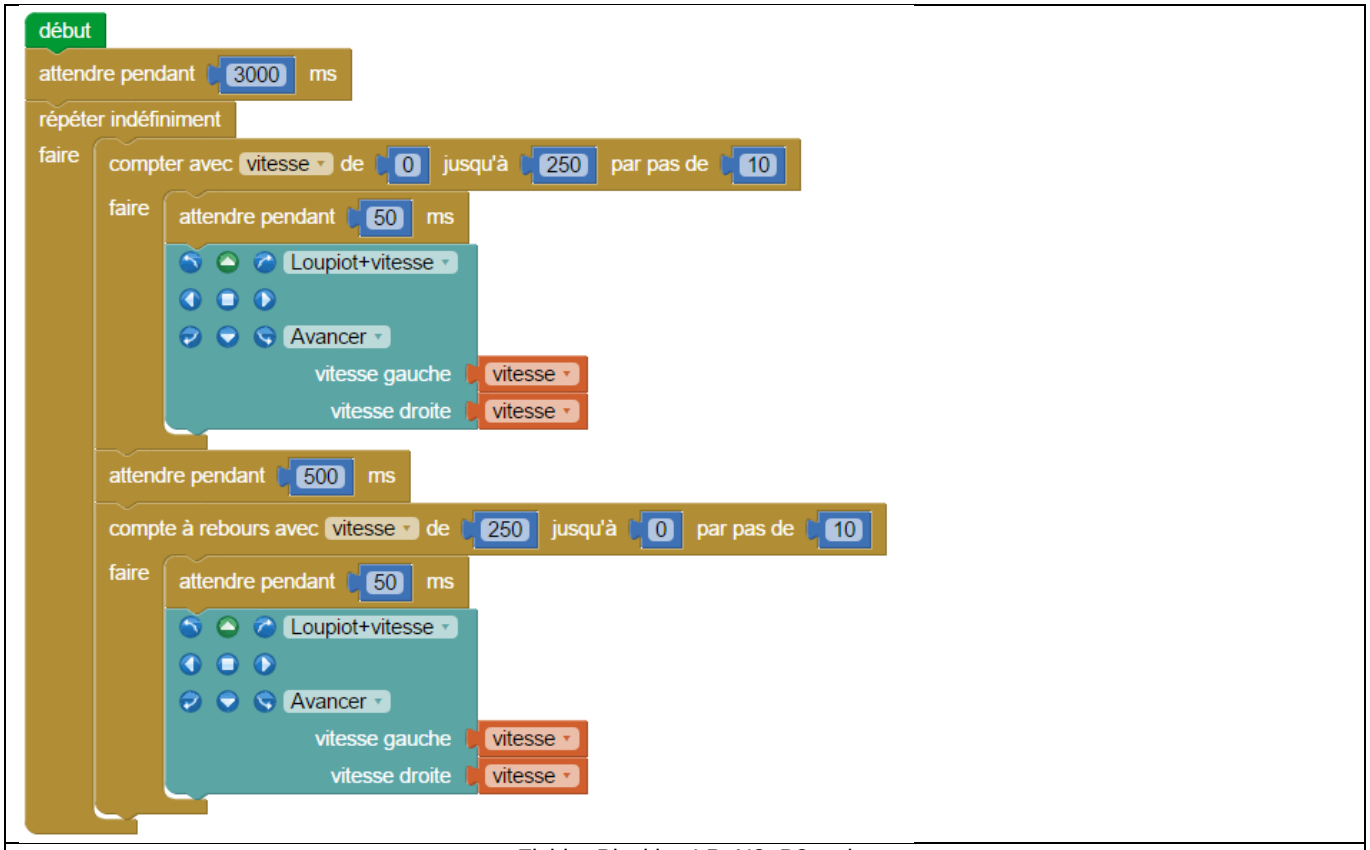
Fichier modèle : LP_N2_B.xml

Objectif : Faire accélérer puis décélérer le robot tout en avançant.

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_B3.xml

Remarque(s) :

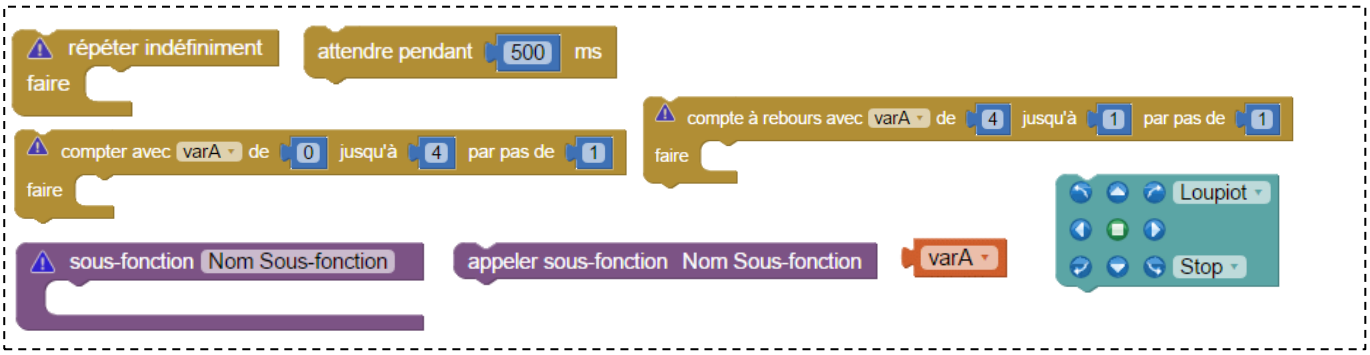
- Une boucle « répété n fois » allant de 0 à 250 par pas de 10 est exécutée 26 fois (0, 10, …, 240, 250). Une accélération ou une décélération dure 1,3 secondes = 26 x 50 ms (le temps d'attente placée dans la boucle « répété n fois »).

Exercice niveau 2 – B4 : Accélération / décélération avec procédure

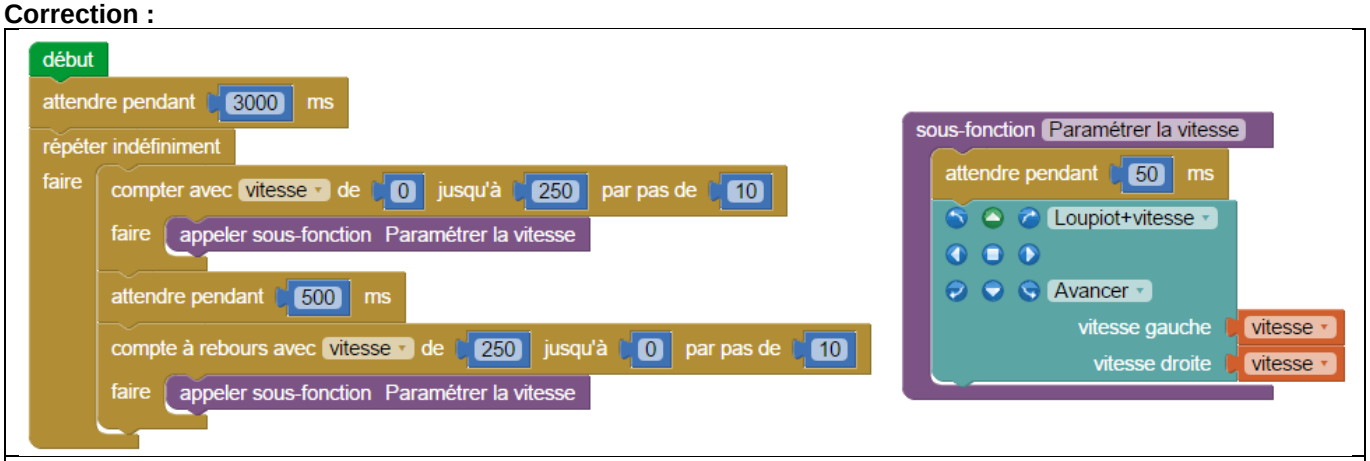
Fichier modèle : LP_N2_B.xml

Objectif : Faire accélérer puis décélérer le robot tout en avançant en utilisant des procédures pour rendre le programme plus lisible.

Instruction(s) utilisée(s) :



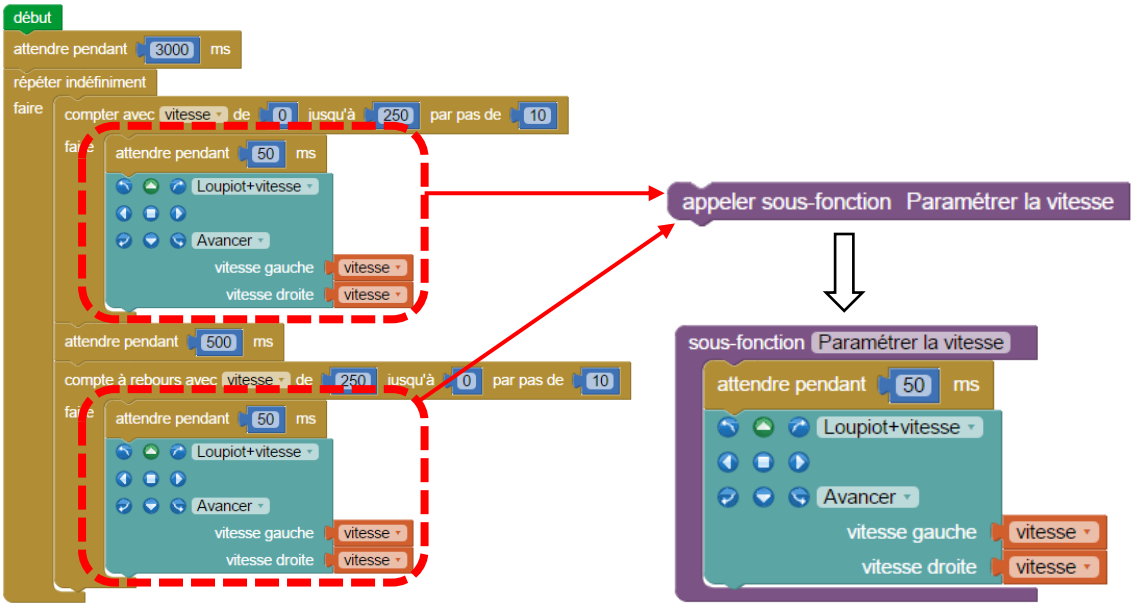
Correction :



Fichier Blockly : LP_N2_B4.xml

Remarque(s) :

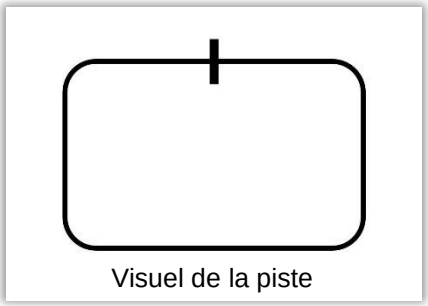
- La procédure sert à ne pas réécrire deux fois la même suite d'instructions et donc à rendre le code principal plus lisible.



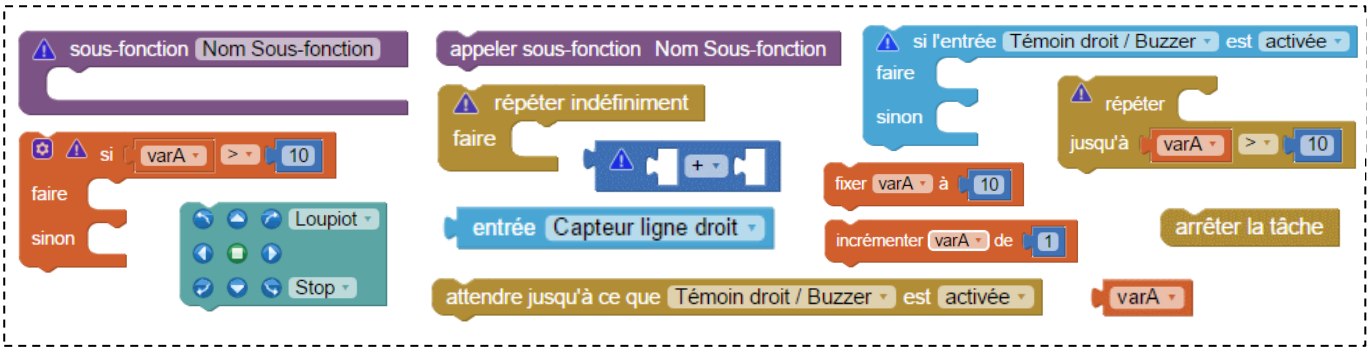
Exercice niveau 2 – C1 : Détecter 3 fois un code

Fichier modèle : LP_N2_C.xml

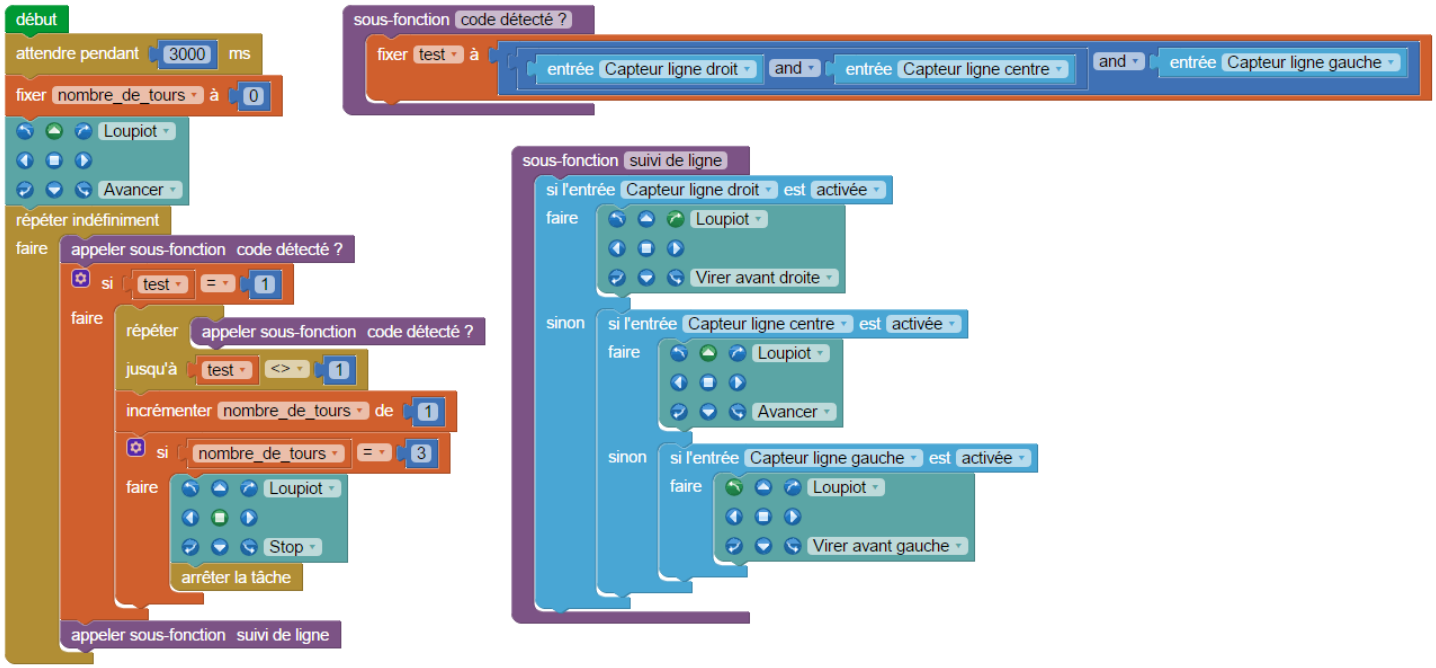
Objectif : Faire suivre une ligne fine au robot.
Un code (3 capteurs de ligne activés) est placé sur la ligne et doit être détecté trois fois par le robot pour le stopper
(Note : Imprimer la piste "Piste_suivi_ligne_avec_code" donnée avec les programmes pour tester ce code).


Visuel de la piste

Instruction(s) utilisée(s) :



Correction :



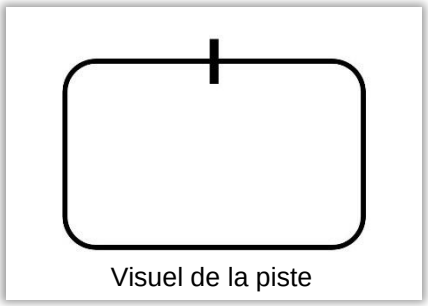
Fichier Blockly : LP_N2_C1.xml

Remarque(s) :

- La procédure « code détecté ? » comporte une opération booléenne : elle renvoie dans la variable « test » 0 si au moins un des 3 capteurs de ligne est désactivé, et 1 s'ils sont tous activés (ce dernier cas correspond à un code détecté).
- Lors de la détection d'un code, pour ne pas recompter la même détection plusieurs fois, on place la boucle « jusqu'à test <> 1 » pour attendre que le robot soit bien sorti du code avant de poursuivre le programme.
- Lorsque le code a été détecté trois fois et que le robot a été arrêté, on force l'arrêt du programme pour que celui-ci ne recommence pas au début de la boucle infinie.

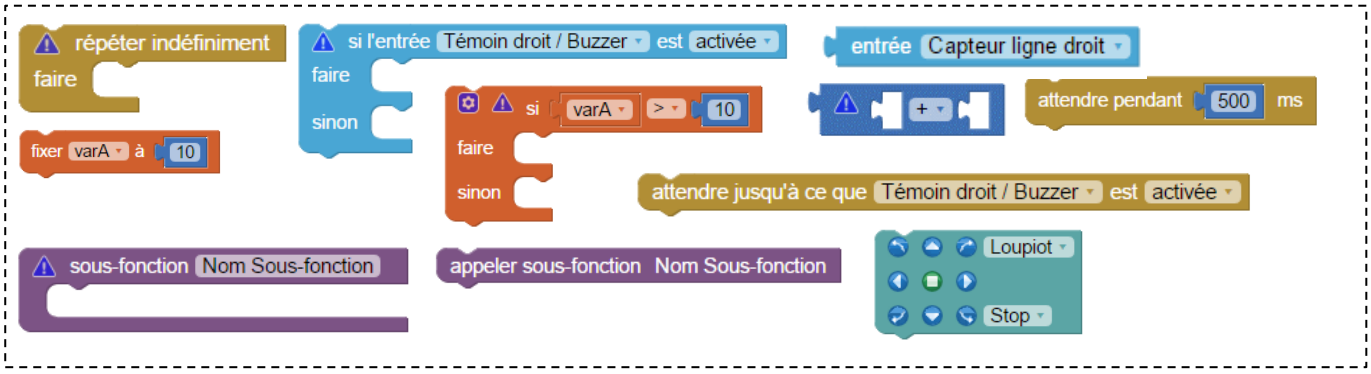
Exercice niveau 2 – C2 : Aller – retour sur une ligne

Fichier modèle : LP_N2_C.xml

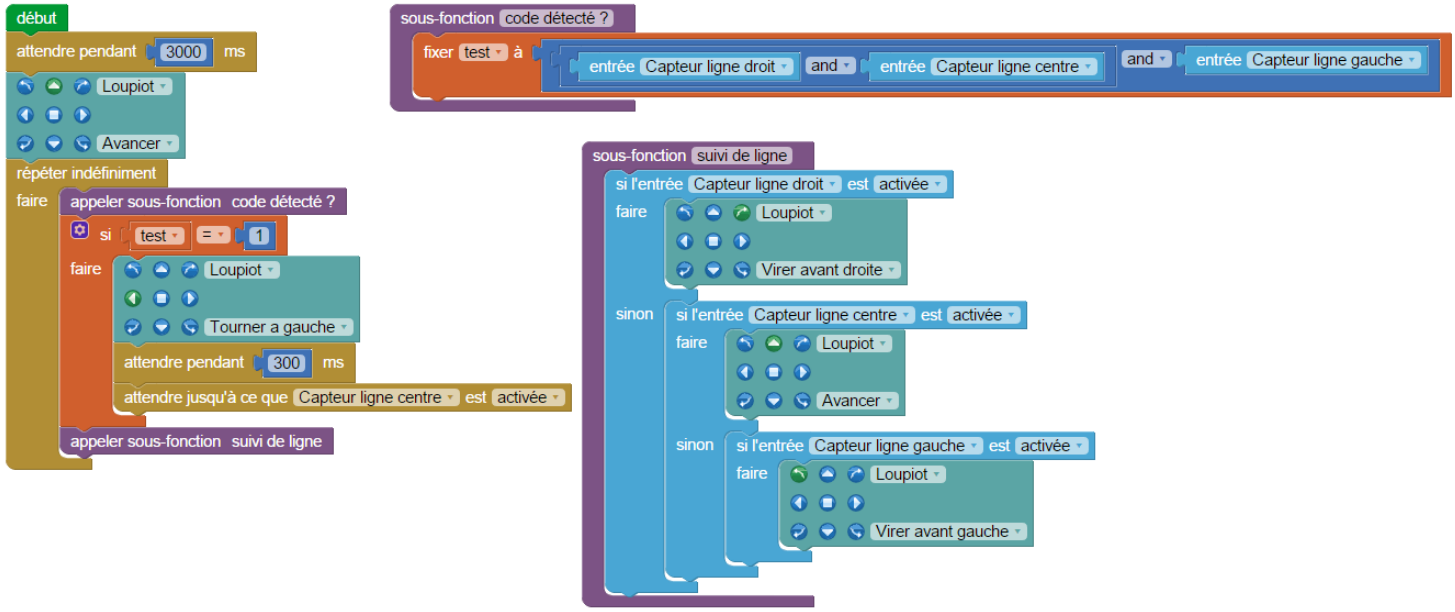

Visuel de la piste

Objectif : Faire suivre une ligne fine au robot.
Lorsqu'un code (3 capteurs de ligne activés) est détecté, le robot fait demi-tour et repart sur la ligne en sens inverse
(Note : Imprimer la piste "Piste_suivi_ligne_avec_code" donnée avec les programmes pour tester ce code).

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_C2.xml

Remarque(s) :

- Quand un code est détecté, le robot tourne sur lui-même pendant 0,3 seconde (le temps de sortir ses 3 capteurs de la ligne noire). Il continue ensuite jusqu'à retrouver la ligne sous son capteur de ligne centre grâce à l'instruction « attendre jusqu'à ce que capteur ligne centre est activé ».

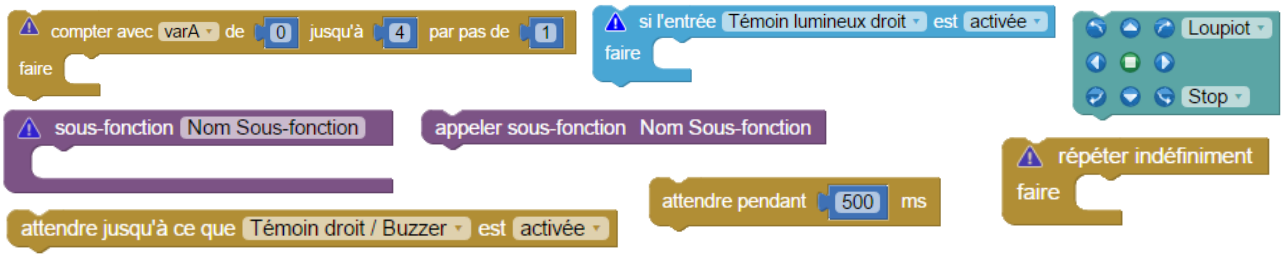
Exercice niveau 2 – C3 : Prison

Fichier modèle : LP_N2_C.xml

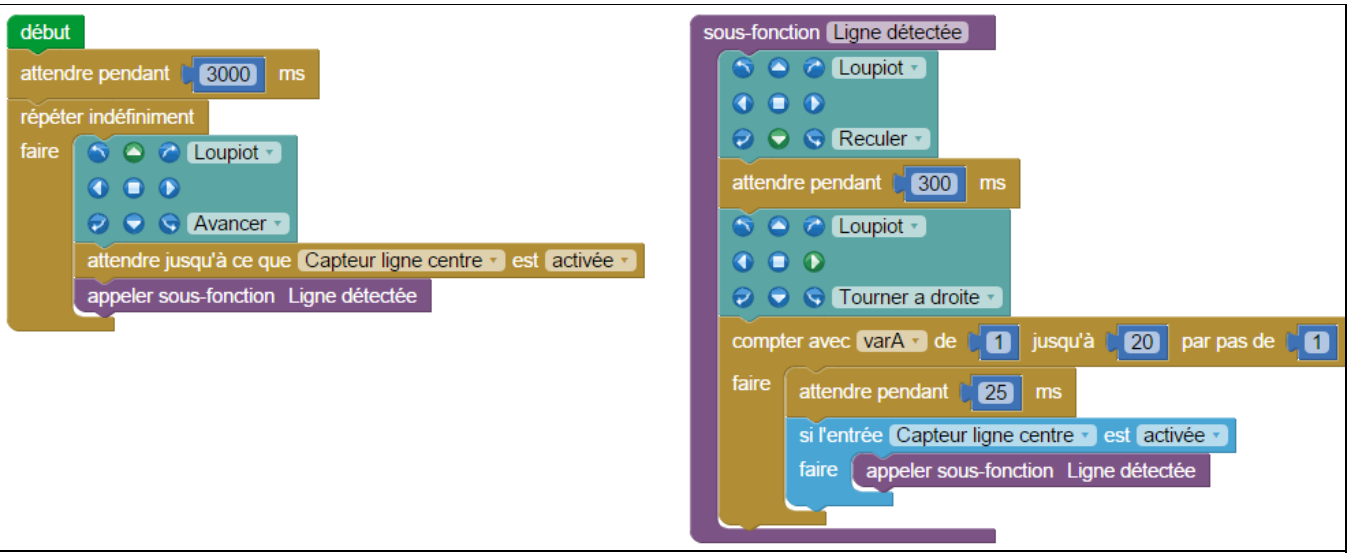
Objectif : Empêcher le robot de sortir d'un périmètre délimité par une ligne noire.
(Note : Imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes du niveau 2B pour tester ce code.)

Visuel de la piste

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_C3.xml

Remarque(s) :

- Lorsqu'une ligne noire est détectée par le capteur de ligne centre, la procédure « Ligne détectée » est déclenchée : le robot va reculer pour s'éloigner de la ligne pendant 0,3 seconde. Il va ensuite changer de trajectoire pour éviter de retoucher la ligne en tournant à droite.
  Pendant que le robot tourne, on vérifie qu'il ne rencontre pas une autre ligne qui pourrait se trouver à sa droite en vérifiant l'état du capteur de ligne centre toute les 25 ms 20 fois.
  Si pendant le virage, il retrouve une ligne sous son capteur de ligne centre, on recommence toute la procédure « Ligne détectée ».
- Le robot tourne pendant 0,5 seconde dans le cas où aucune ligne n'est détectée.
  En effet, comme expliqué précédemment dans les remarques, la boucle « compter avec varA… » vérifiant qu'il n'y ait pas d'autres lignes pendant le virage s'exécute 20 fois et contient une attente de 25 ms.
  Le temps de virage total vaut donc 20 x 25 ms = 0,5 seconde.

Programmation version de base + options niveau 3

Introduction

Le niveau 3 vous fait découvrir des exemples d'utilisation concrets autour des différentes options du robot Loupiot. Attention : ce niveau nécessite de bien connaitre les fonctionnalités de base du robot Loupiot et les bases de la programmation par blocs apprises dans les niveaux 1 et 2.

Liste des programmes du niveau 3

Nom du fichier	Description	Objectif
Niveau 3 A (Modèle : LP_N3_A.xml)		
LP_N3_A1.xml	Recevoir une donnée	Fonctionnalité matérielle abordée : - Option Bluetooth Notions de programmation abordées : - Communication sans fil
LP_N3_A2.xml	Envoyer une donnée	
LP_N3_A3.xml	Contrôler l'utilisation du robot à distance	
Niveau 3 B (Modèle : LP_N3_B.xml)		
LP_N3_B1.xml	Lire une distance avec le debug	Fonctionnalité matérielle abordée : - Option capteur de distance à ultrasons
LP_N3_B2.xml	Radar de proximité	
LP_N3_B3.xml	Suivi de ligne avec évitement d'obstacle	
Niveau 3 C (Modèle : LP_N3_C.xml)		
LP_N3_C1.xml	Prévenir la présence d'un obstacle	Fonctionnalité matérielle abordée : - Option détection d'obstacle
LP_N3_C2.xml	Suivi de ligne avec évitement d'obstacle	
Niveau 3 C (Modèle : LP_N3_C.xml)		
LP_N3_D1.xml	Dessiner une forme géométrique	Fonctionnalité matérielle abordée : - Option porte-stylo
LP_N3_D2.xml	Remplir au mieux une zone délimitée	

Niveau 3 A – Introduction à l'option Bluetooth

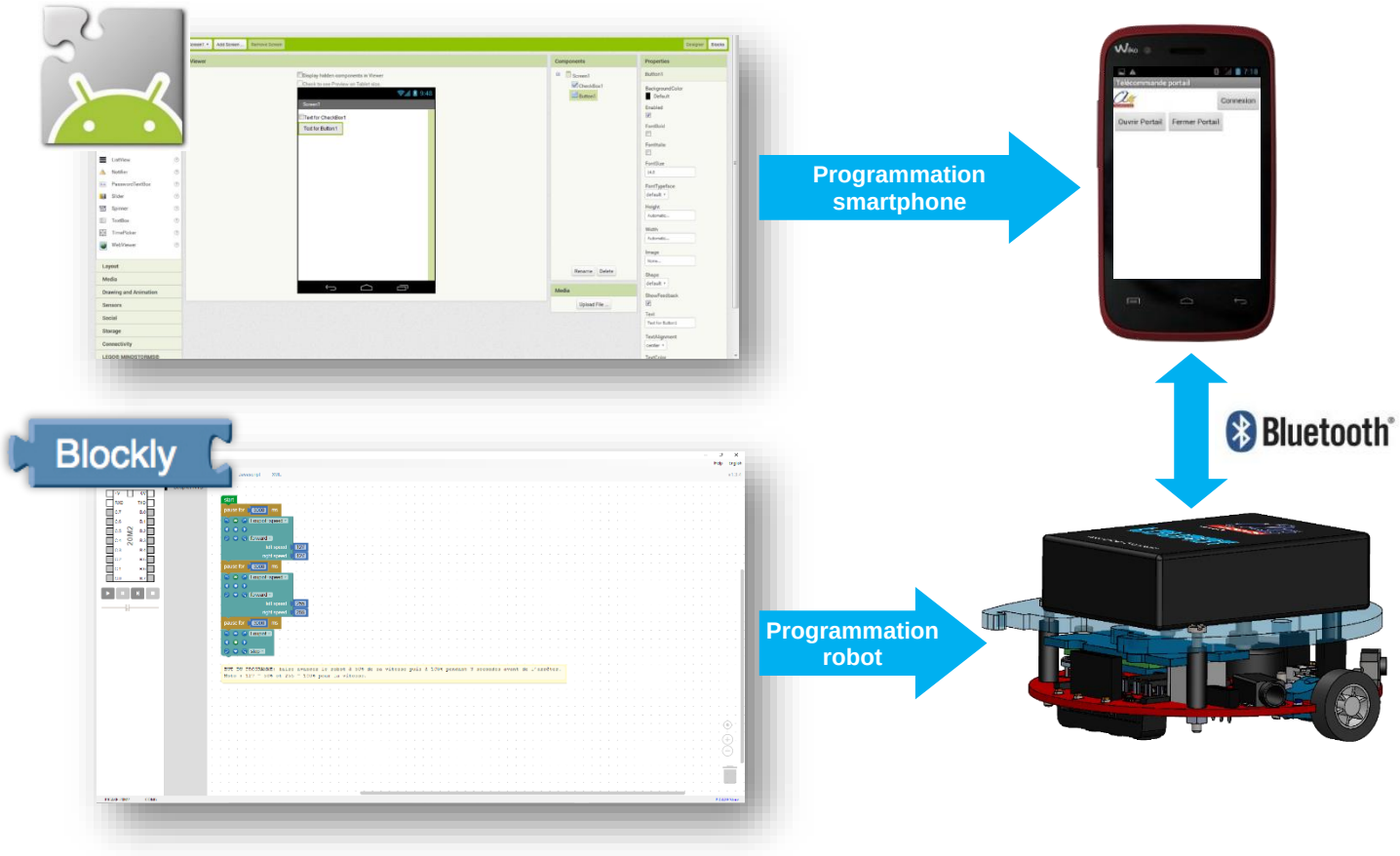
L'option Bluetooth permet d'illustrer la communication sans fil bidirectionnelle entre un smartphone Android et le robot Loupiot via un module Bluetooth Grove. Les applications Android proposées dans les exemples ci-dessous sont réalisées sur AppInventor (nécessite un compte Gmail pour la création d'une session).

AppInventor est un environnement de développement d'application Android sur une plateforme internet fonctionnant avec Blockly. Elle a été développée par le MIT pour l'éducation. Pour chaque exemple, nous proposons 3 fichiers d'application Android sous AppInventor :

- Un fichier « nom de l'application. Apk », version finie de l'application à installer sur votre smartphone.
  Utiliser ce fichier si vous ne souhaitez pas utiliser AppInventor et seulement coder la partie robot.
- Un fichier « nom de l'application. Aia », fichier projet du programme fini.
  Ce fichier s'ouvre sur AppInventor afin de regarder, modifier ou améliorer le code de l'application.
- Un fichier « Bluetooth_base.aia », fichier projet de base proposé par A4 pour la communication Bluetooth.
  Il peut être apparenté au programme modèle pour Blockly. Il sert de base pour coder une application sur AppInventor. Le projet de base contient toute la partie connexion/déconnexion au robot par Bluetooth.

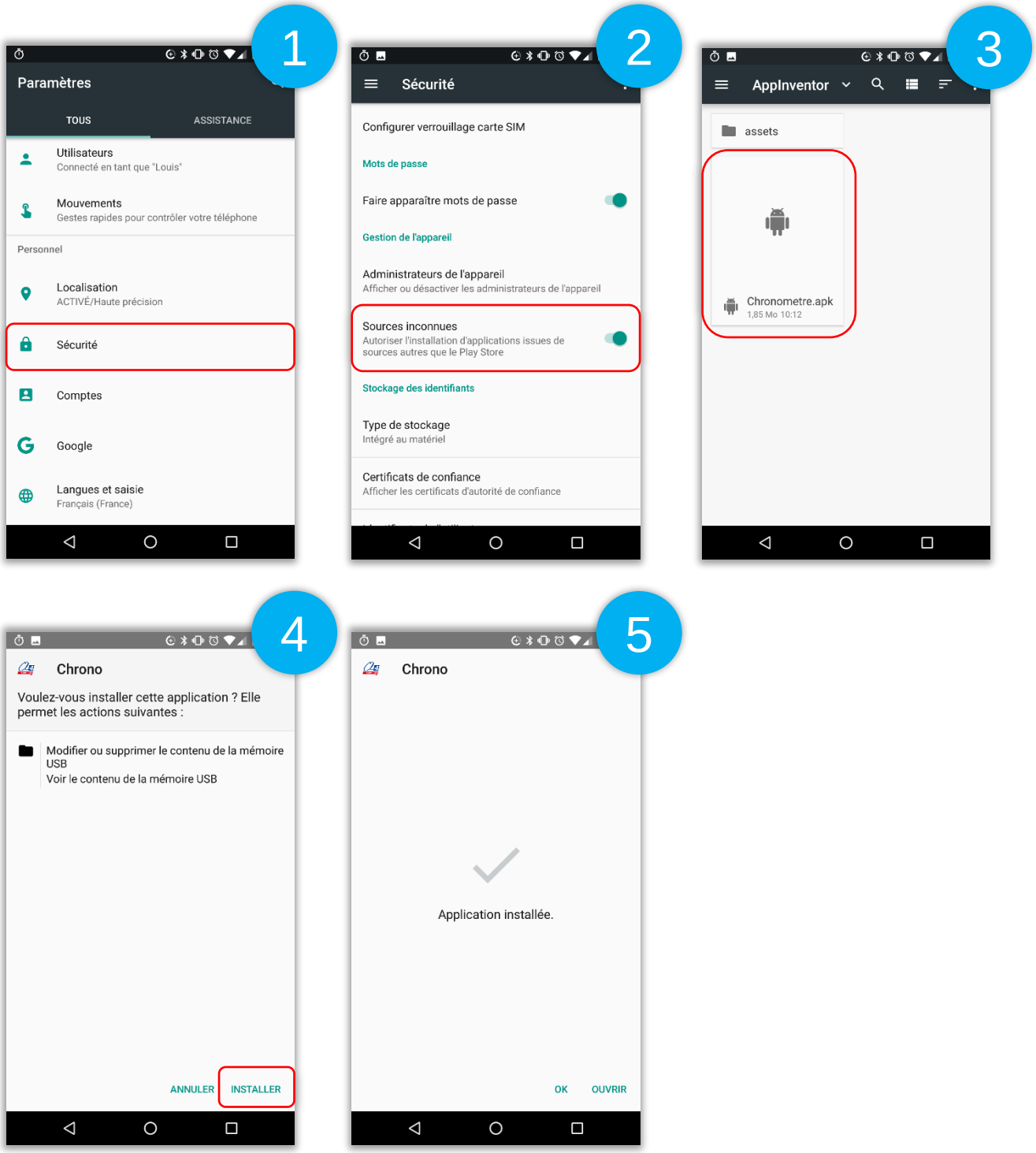
Attention : cette documentation n'indique pas comment utiliser AppInventor.
Si vous souhaitez utiliser les fichiers .Aia, une connaissance de l'environnement de développement est requise.
Nous expliquons dans les exemples la logique de la partie « blocks » de l'application AppInventor.
Pour apprendre à utiliser AppInventor, nous proposons des vidéos formations gratuites (voir site www.a4.fr)
Nous vous conseillons également de suivre les différents tutoriels présents sur internet ou la documentation proposée par le MIT en anglais.

Avant de commencer à programmer veuillez monter le module Bluetooth comme indiqué dans la partie montage de la documentation.



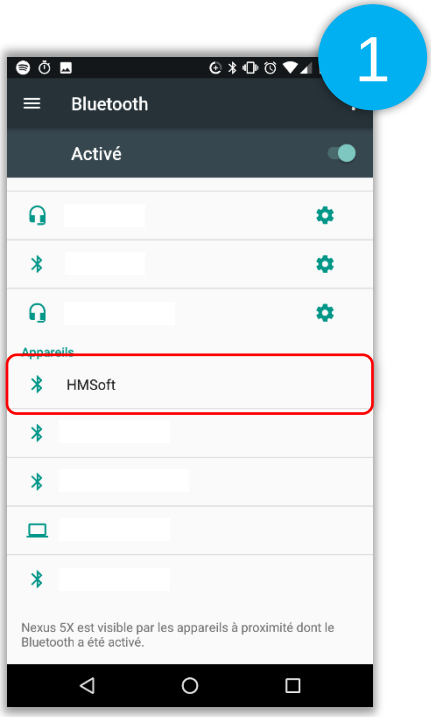
Procédure de connexion au robot à partir d'une application Android A4

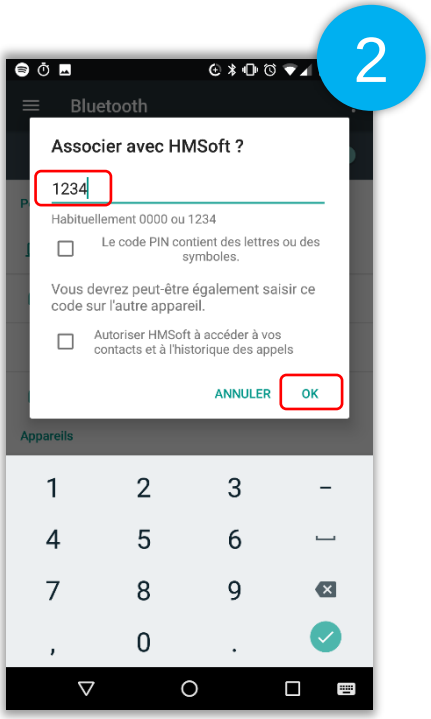
Pour tester une application Android réalisée avec AppInventor, installez-la sur votre smartphone à partir du fichier .Apk fourni ou créé par vous-même et préalablement enregistré dans la mémoire de votre téléphone.
Attention à bien cocher dans Paramètres >> sécurité, la case : « Sources inconnues ».
Pour installer l'application, cliquer sur le fichier .Apk dans les fichiers de votre smartphone.

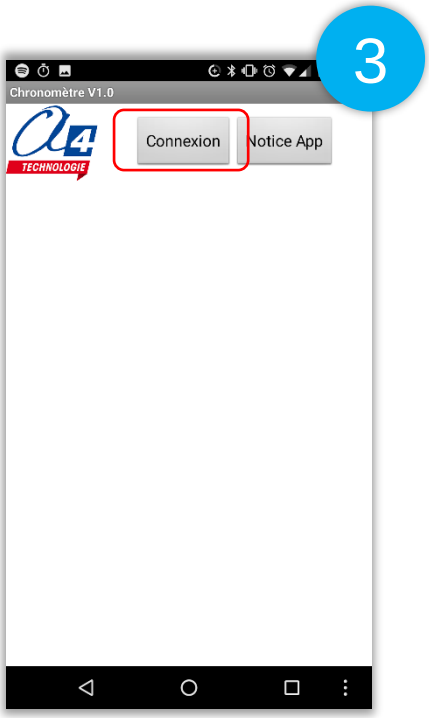


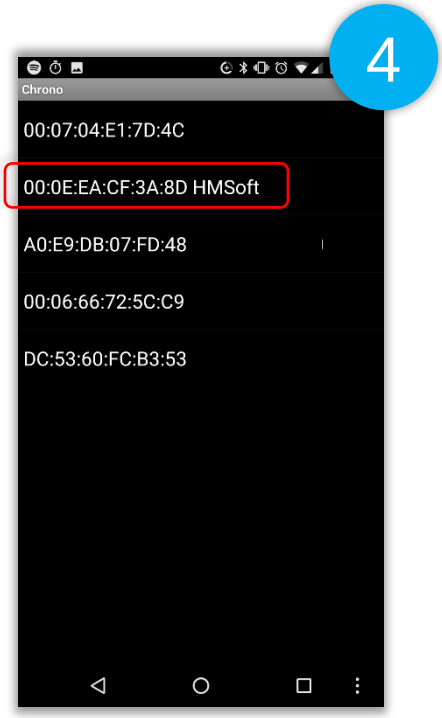
Une fois l'application installée, mettre le robot sous tension. Le module Bluetooth possède une LED témoin sur sa face inférieure près de la prise de connexion.
Quand le robot n'est pas connecté à un smartphone, cette LED clignote indiquant qu'elle recherche un appareil auquel se connecter. Une fois la connexion établie, la LED éclaire en permanence indiquant que la connexion est réussie.

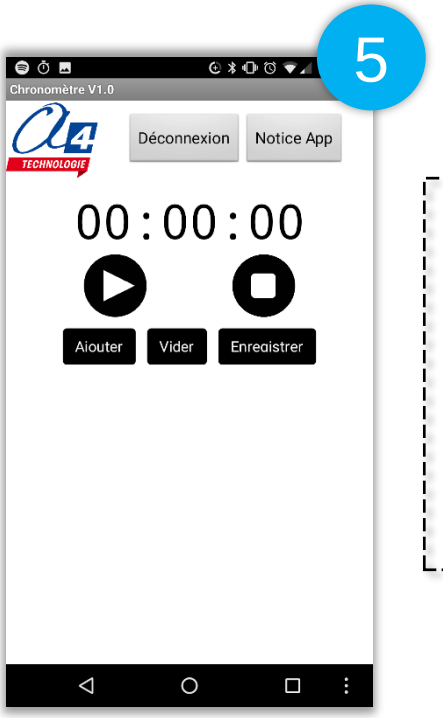
Une fois le robot mis sous tension, aller dans les paramètres Bluetooth du smartphone pour appairer le module Grove. Lancer une recherche d'appareil.
Le module se nomme « HM Soft ».
Cliquer dessus et rentrer le code pin « 0000 » ou « 1234 ».
Une fois l'appairage terminé, lancer l'application préalablement installée et cliquer sur « Connexion ».
Le module doit maintenant se trouver dans la liste qui vient d'apparaitre. Cliquer dessus.
Si la connexion est réussie, la LED du module bluetooth doit se figer et l'application doit afficher son contenu.
Si la connexion échoue, retenter l'opération ou recommencer un appairage.

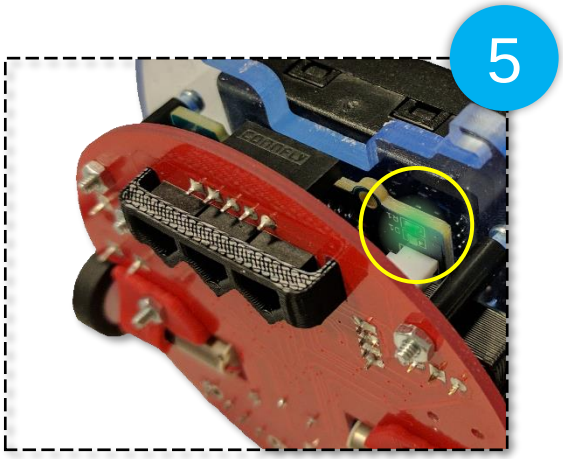












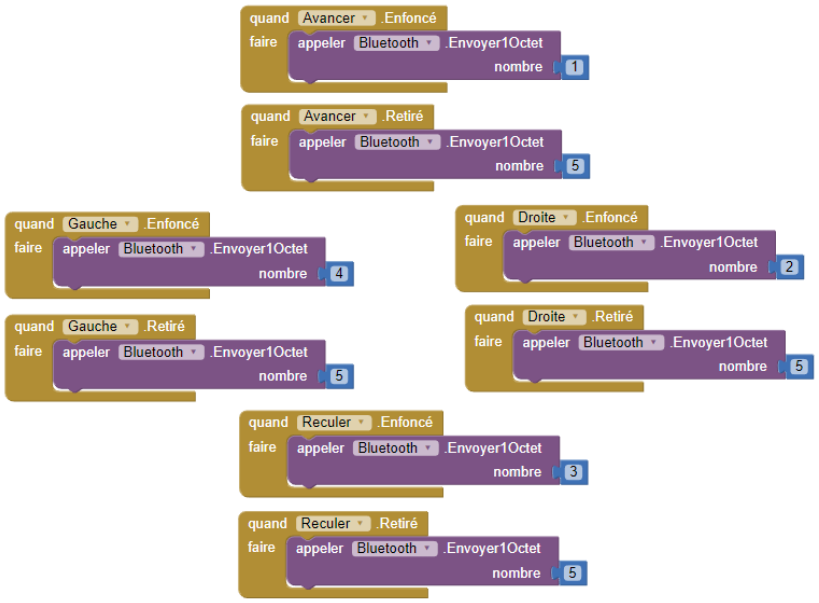
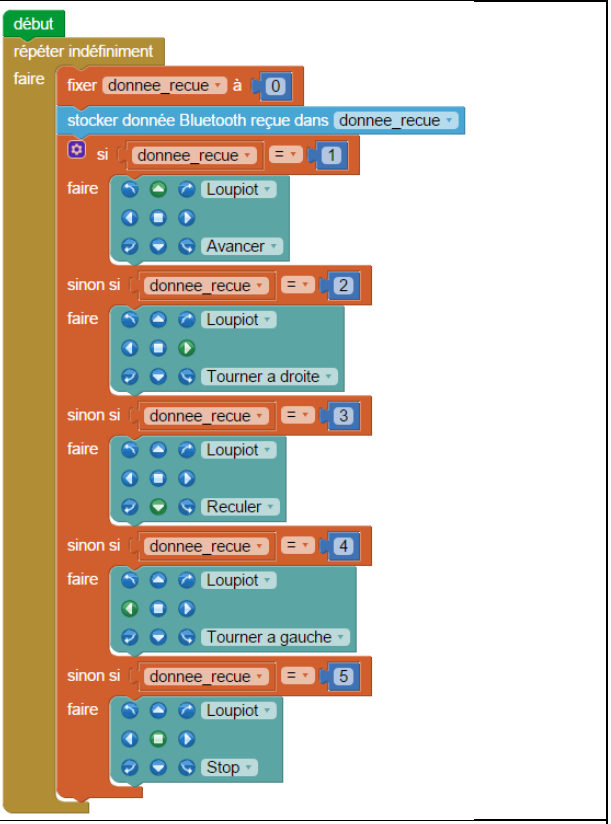
Exercice niveau 3 – A1 : Recevoir des données

Fichier modèle : LP_N3_A.xml / Bluetooth_base.aia

Objectif : Contrôler les déplacements du robot à partir d'un smartphone envoyant des consignes via le Bluetooth.

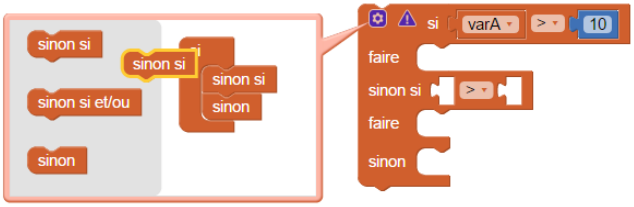
Notion(s) abordée(s) : Réception de données par le robot.

Correction :

AppInventor	Blockly
	
Fichier aia AppInventor : Loupiot_Rx.aia / Loupiot_Rx.apk	Fichier Blockly : LP_N3_A1.xml

Remarque(s) :

- Pour ajouter ou retirer une nouvelle condition dans le bloc « Si », cliquer sur l'engrenage et glisser ou retirer des blocs dans le bloc principal comme illustré ci-contre.



- Le bloc « stocker donnée bluetooth reçue dans donnee_recue » ne modifie pas la valeur actuelle de la variable « donnee_recue » si aucunes données ont été envoyées par le smartphone. On initialise donc cette variable à zéro manuellement à chaque tour de boucle avant d'appeler des données reçues par bluetooth.

Exercice niveau 3 – A2 : Envoyer des données

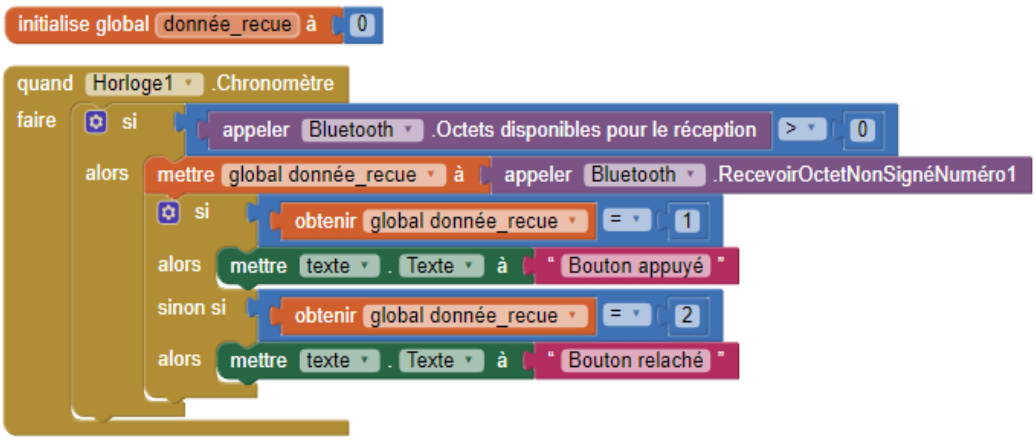
Fichier modèle : LP_N3_A.xml / bluetooth_base.aia

Objectif : Faire connaitre au smartphone l'état du capteur de ligne centre du robot en envoyant des données quand il est activé ou désactivé.

Notion(s) abordée(s) : Envoyer des données par le robot.

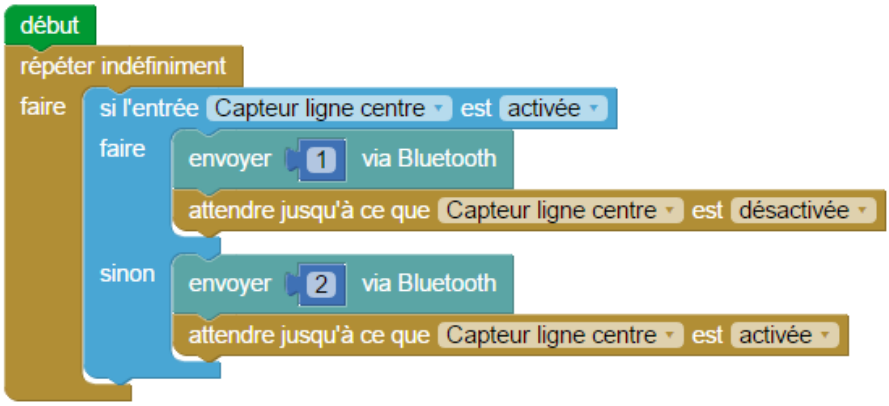
Correction :

AppInventor



Fichier aia AppInventor :
Loupiot_Tx.aia / Loupiot_Tx.apk

Blockly



Fichier Blockly : LP_N3_A2.xml

Remarque(s) :

- Le bloc « appeler Bluetooth RecevoirOctetNonSignéNuméro1 » ne doit pas être utilisé si on n'a pas vérifié que des données étaient disponibles sous peine de faire crasher l'application.
- Le bloc « appeler Bluetooth Octets disponibles pour la réception » renvoie le nombre de données qui ont été reçues par le smartphone et qui n'ont encore jamais été lues par le bloc vu précédemment dans la remarque précédente.
- Un composant horloge a été placé dans la face « designer » d'AppInventor. Il exécute les blocs qu'il contient toutes les 50 ms (temps de déclenchement paramétrable dans AppInventor).
- Dans le programme picaxe les attentes placées après les envois de données servent à n'envoyer qu'une seule fois l'information lors d'un changement d'état du capteur. Si elles n'étaient pas présentes, la boucle infinie s'exécutant rapidement, le smartphone se retrouverait submergé de données.

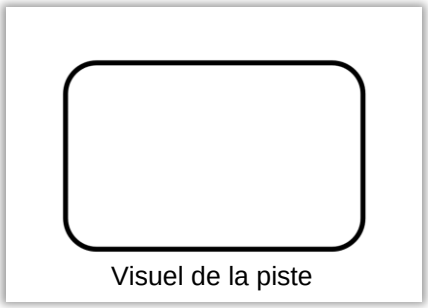
Exercice niveau 3 – A3 : Contrôler l'utilisation du robot à distance

Fichier modèle : LP_N3_A.xml / bluetooth_base.aia

Objectif : Quand un code est détecté par les capteurs de ligne du robot, celui-ci demande au smartphone l'autorisation de suivre une ligne et attend la réponse. Note : Imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes du niveau 2 B pour tester ce code.

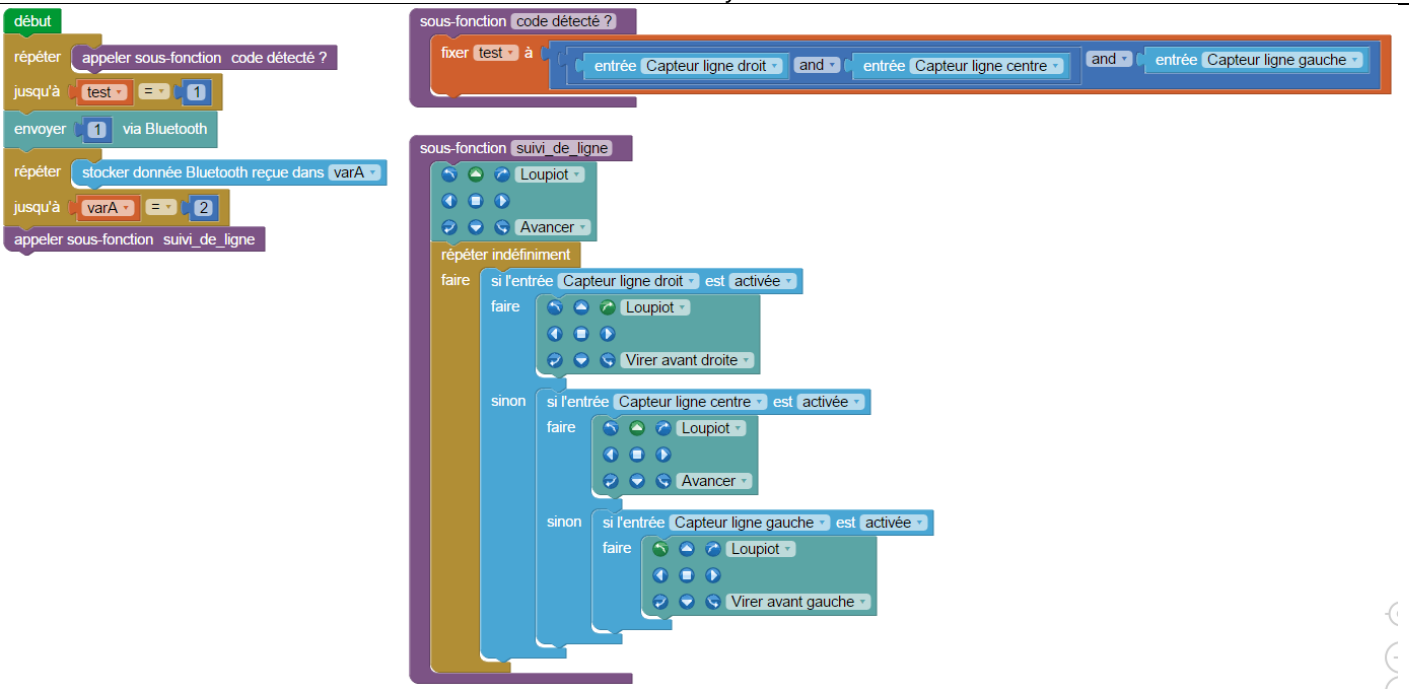
Notion(s) abordée(s) : Envoyer et recevoir dans données avec le robot.

Correction :



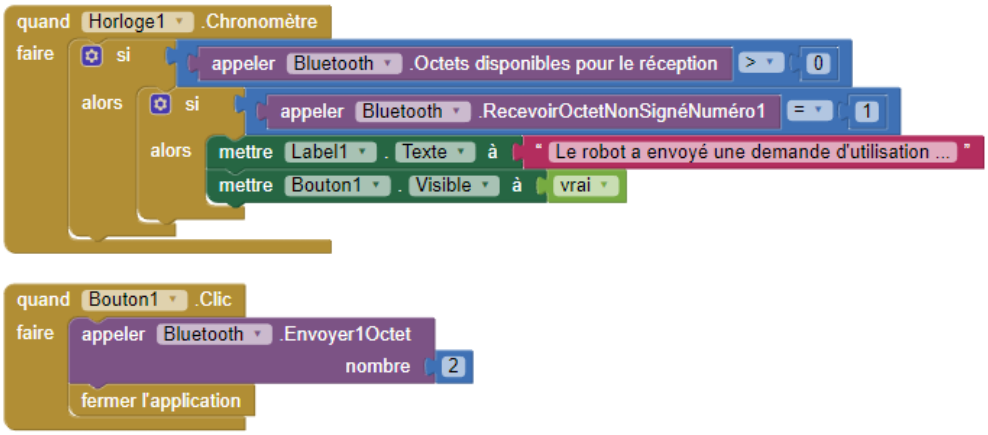
Visuel de la piste

Blockly



Fichier Blockly : LP_N3_A3.xml

AppInventor



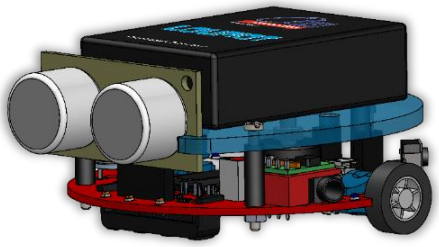
Fichier aia AppInventor :
Loupiot_Tx_Rx.aia / Loupiot_Tx_Rx.apk

Remarque(s) :

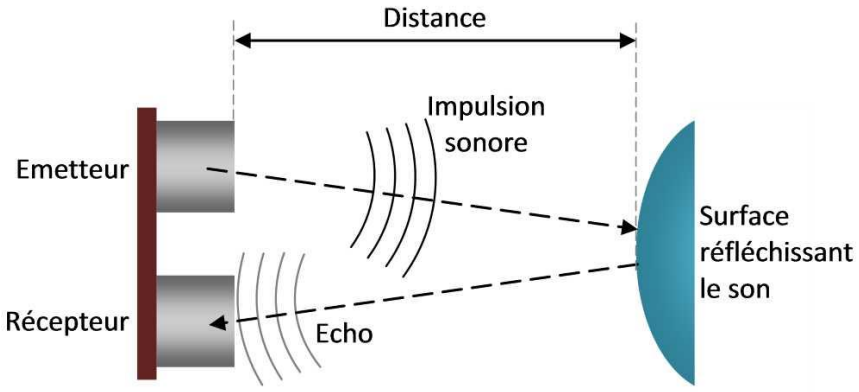
- Quand le bouton-poussoir est appuyé, le robot envoie le code « 1 » au smartphone et attend de recevoir le code « 2 » pour commencer à suivre une ligne.
  Quand le smartphone reçoit le code « 1 », il affiche un bouton permettant d'envoyer le code « 2 » au robot.

Niveau 3 B – Introduction à l'option capteur de distance à ultrasons

L'option capteur de distance à ultrasons permet au robot de connaître avec précision la distance en centimètre qui le sépare d'un objet suffisamment large (comme un verre d'eau ou un autre robot Loupiot).



Attention : ce capteur ne mesure pas en ligne droite mais en cône. Il peut donc détecter des objets légèrement excentrés de son axe.
Placé dans le vide, il mesure des distances allant de 2 à 255 cm.
Quand une mesure de distance est demandée, une LED rouge s'allume brièvement sur la face arrière du capteur pour indiquer que la mesure a bien été effectuée.
Si la LED est en permanence allumée cela veut dire que le capteur lance des mesures en permanence.



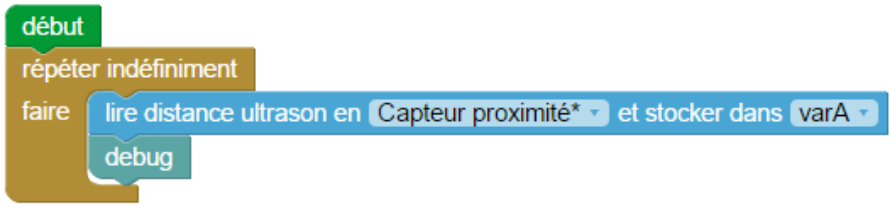
Exercice niveau 3 – B1 : Lire une distance avec le debug

Fichier modèle : LP_N3_B.xml

Objectif : Afficher la valeur mesurée par le capteur de distance à ultrasons sur l'ordinateur avec le debug.

Correction :

Blocs



Fichier Blockly : LP_N3_B1.xml

Remarque(s) :

- Garder le robot branché à l'ordinateur pour voir le résultat du debug sur l'ordinateur.
- Il n'y a pas de temps d'attente placé dans la boucle infinie.
  On peut s'attendre en conséquence à voir la LED rouge du capteur à ultrasons éclairer en continue car il devrait y avoir un grand nombre de mesures par secondes.
  Or, on remarque que la LED clignote plutôt rapidement mais pas en continue.
  C'est dû au bloc debug qui prend un temps d'exécution non négligeable pour transmettre les valeurs retenues dans la mémoire du robot vers l'ordinateur.

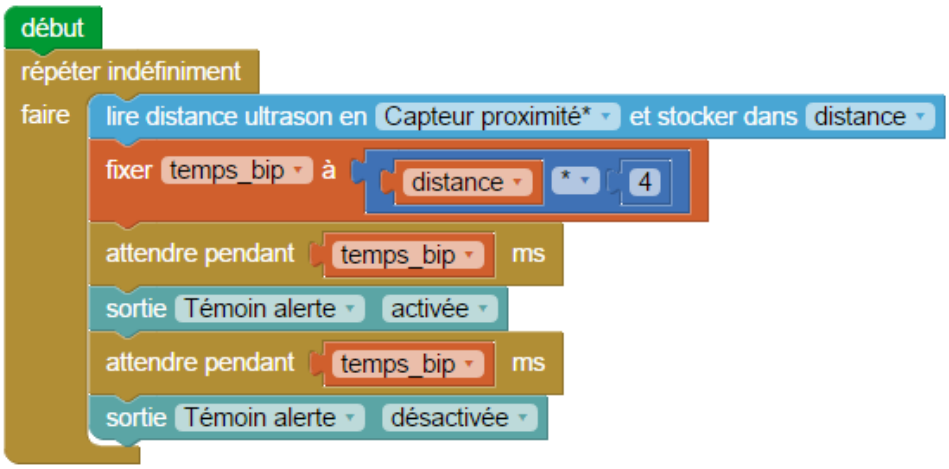
Exercice niveau 3 – B2 : Radar de proximité

Fichier modèle : LP_N3_B.xml

Objectif : Faire clignoter le témoin lumineux alerte avec une fréquence variant en fonction de la distance mesurée par le capteur à ultrasons (similaire à un radar de recul pour une voiture).

Correction :

Blocs



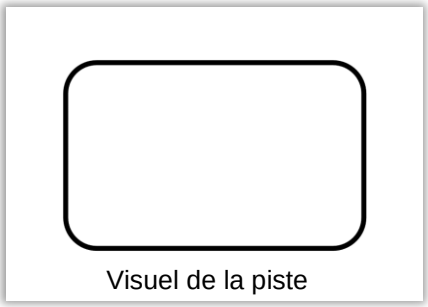
Fichier Blockly : LP_N3_B2.xml

Remarque(s) :

Exercice niveau 3 – B3 : Suivi de ligne avec évitement d'obstacle

Fichier modèle : LP_N3_B.xml

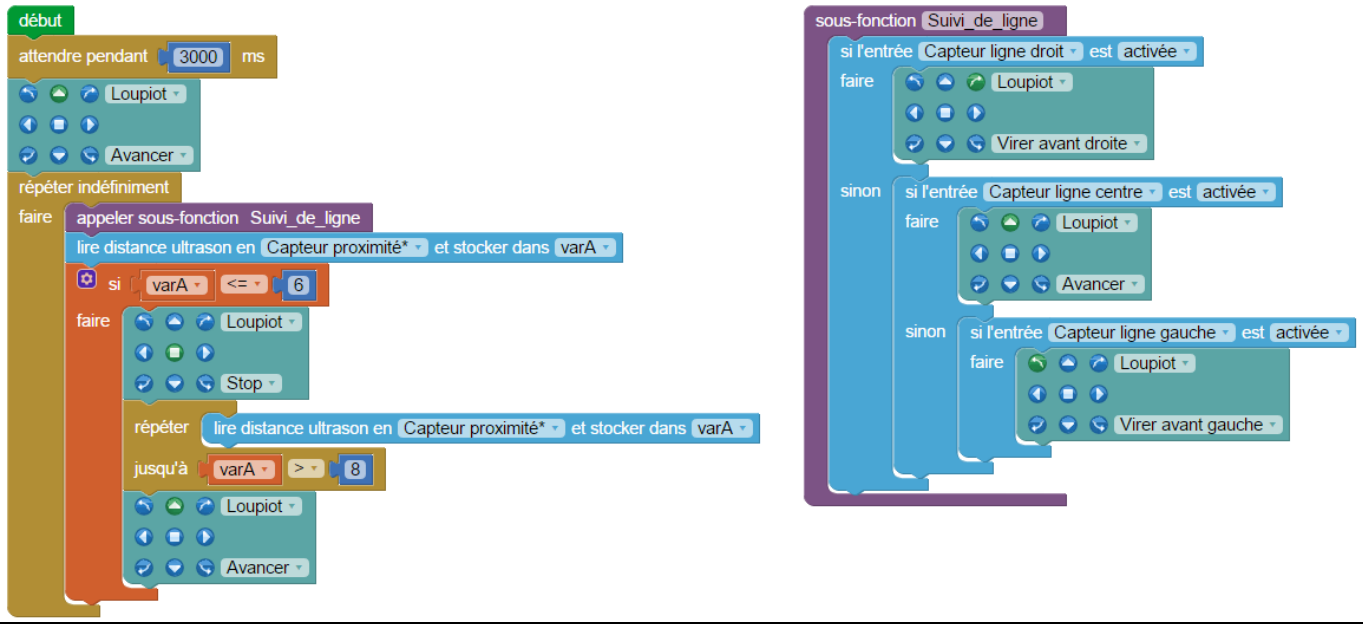
Objectif : Le robot suit une ligne fine et s'arrête quand un obstacle à moins de 6 cm est détecté.
(Note : Imprimer la piste "Piste_suivi_ligne_type1" donnée avec les programmes du niveau 2 B pour tester ce code.)



Visuel de la piste

Correction :

Blocs

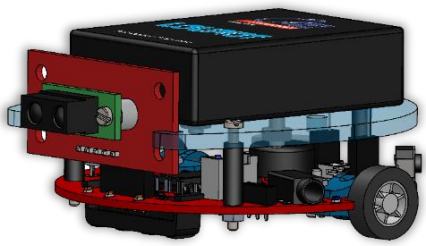


Fichier Blockly : LP_N3_B3.xml

Remarque(s) :

- Une procédure a été placée pour rendre le code principal plus lisible.
- Les valeurs du capteur à ultrasons étant assez instables d'une mesure à l'autre, la condition d'arrêt du robot est à 6 cm et celle de reprise à 8 cm. Cela évite que le robot ne s'arrête et ne redémarre instantanément à cause d'une variation de mesure. On appelle ce procédé une hystérésis.

Niveau 3 C – Introduction à l'option détection d'obstacle



L'option détection d'obstacle comprend un capteur de proximité infrarouge permettant au robot Loupiot de savoir si un obstacle se trouve à moins de 5 cm de lui.
Attention : Les surfaces noires et opaques peuvent ne pas être détectées par le capteur.
Quand un obstacle est détecté, une lumière rouge se déclenche sur le capteur pour signifier qu'il l'a bien vu.
Ce capteur marche en logique inverse, c'est-à-dire que sa patte reliée au robot est désactivée quand il voit un obstacle et est activée quand il n'y a pas d'obstacle.

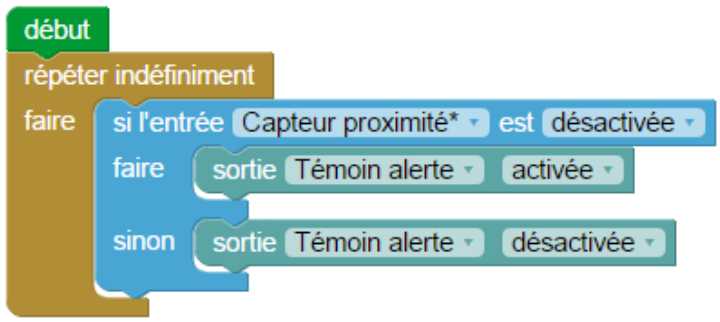
Exercice niveau 3 – C1 : Prévenir la présence d'un obstacle

Fichier modèle : LP_N3_C.xml

Objectif : Le témoin lumineux alerte s'allume quand un obstacle passe devant le capteur.

Correction :

Blocs



Fichier Blockly : LP_N3_C1.xml

Remarque(s) :

- La condition illustre bien la logique inverse du capteur : s'il y a un obstacle, le capteur est désactivé et s'il n'y a pas d'obstacle, le capteur est activé.

Exercice niveau 3 – C2 : Suivi de ligne avec évitement d'obstacle

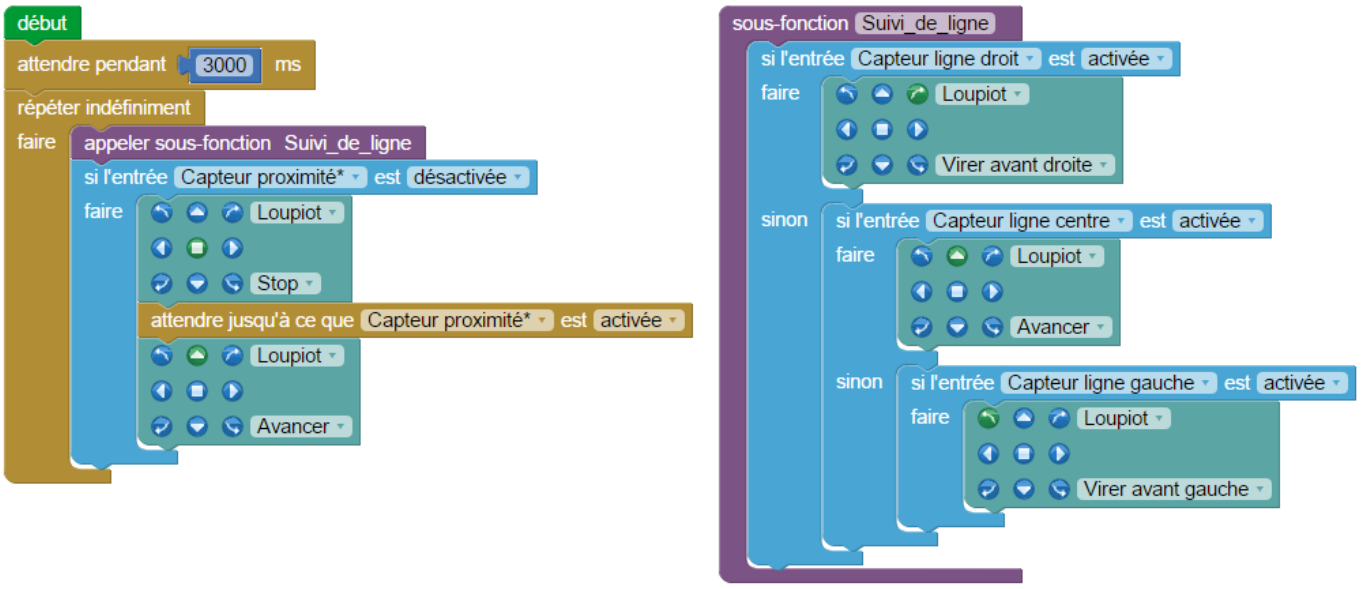
Fichier modèle : LP_N3_C.xml

Objectif : Le robot suit une ligne fine et s'arrête quand un obstacle à moins de 5 cm est détecté. Note : Imprimer la piste « Piste_suivi_ligne_type1 » donnée avec les programmes de niveau 2 B pour ce programme.

Visuel de la piste

Correction :

Blocs



Fichier Blockly : LP_N3_C2.xml

Remarque(s) :

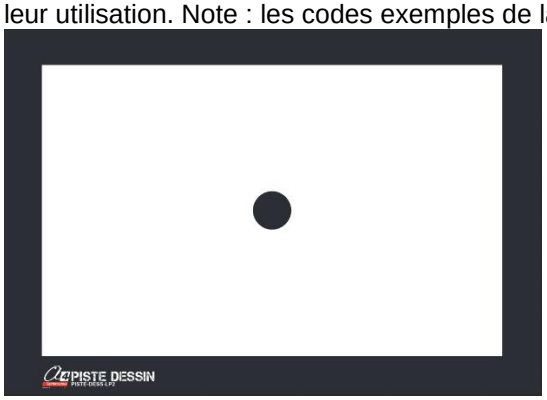
- Une procédure a été placée pour rendre le code principal plus lisible.

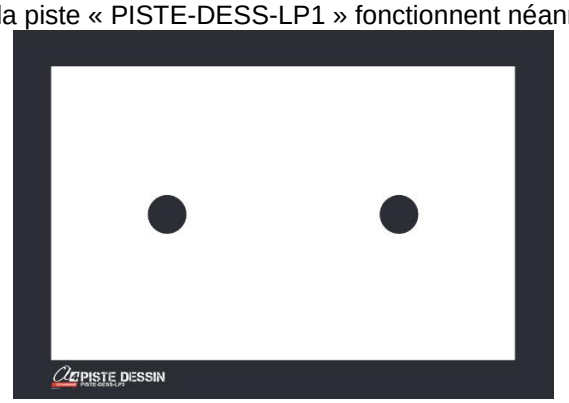
Niveau 3 D – Introduction à l'option porte-stylo

L'option porte-stylo permet au robot Loupiot de tracer des lignes grâce à une pièce clipsable à l'arrière du robot pouvant accueillir un stylo ou un marqueur de diamètre 9,5 mm.
L'option est fournie avec trois pistes de type tableau blanc permettant d'effacer les tracés précédents.

Pour tous les exemples suivants, utiliser la piste PISTE-DESS-LP1 pour tester les programmes en positionnant le robot au milieu de la feuille en début de programme.

Deux autres pistes sont fournies pour aller plus loin dans les expérimentations mais aucun exemple n'est donné pour leur utilisation. Note : les codes exemples de la piste « PISTE-DESS-LP1 » fonctionnent néanmoins sur celle-ci.





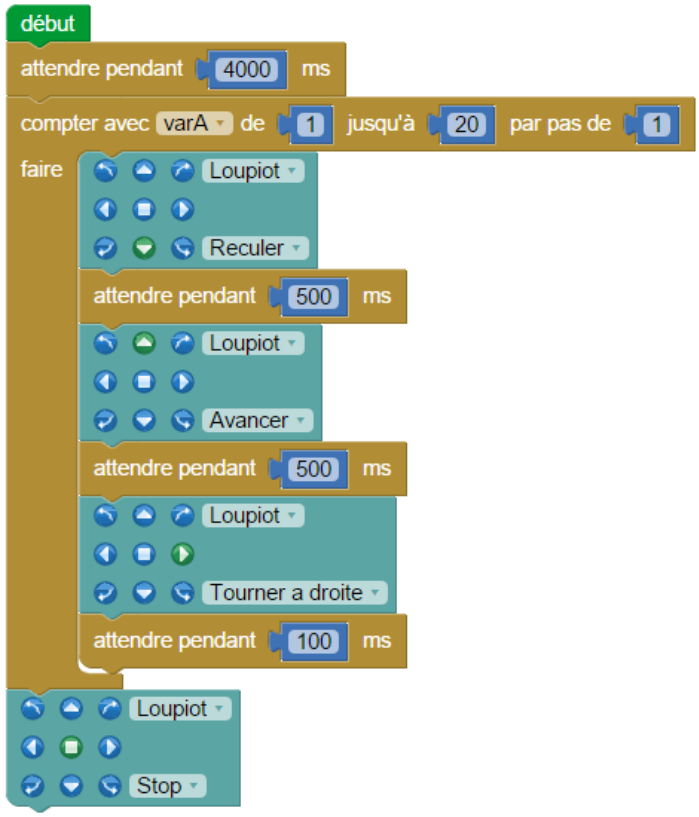
Exercice niveau 3 – D1 : Dessiner une forme géométrique

Fichier modèle : LP_N3_D.xml

Objectif : Dessiner un soleil en traçant un motif plusieurs fois.

Correction :

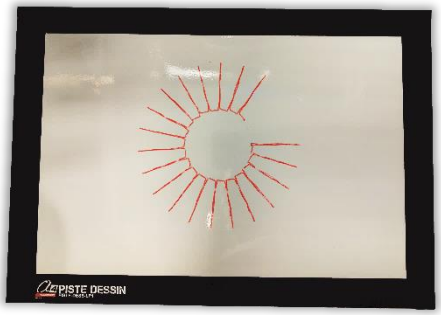
Blocs



Fichier Blockly : LP_N3_D1.xml

Remarque(s) :

- Une attente de 4 secondes est placée au début de chaque programmes de ce niveau afin de vous laisser le temps de placer le robot sur la piste.
- Ci-contre voici le résultat du motif dessiné par le robot.
- Note : vous pouvez modifier le nombre d'exécutions de la boucle « compter avec varA… » pour former un soleil complet. Ce nombre varie en fonction de la vitesse des moteurs et donc de l'état des batteries du robot.



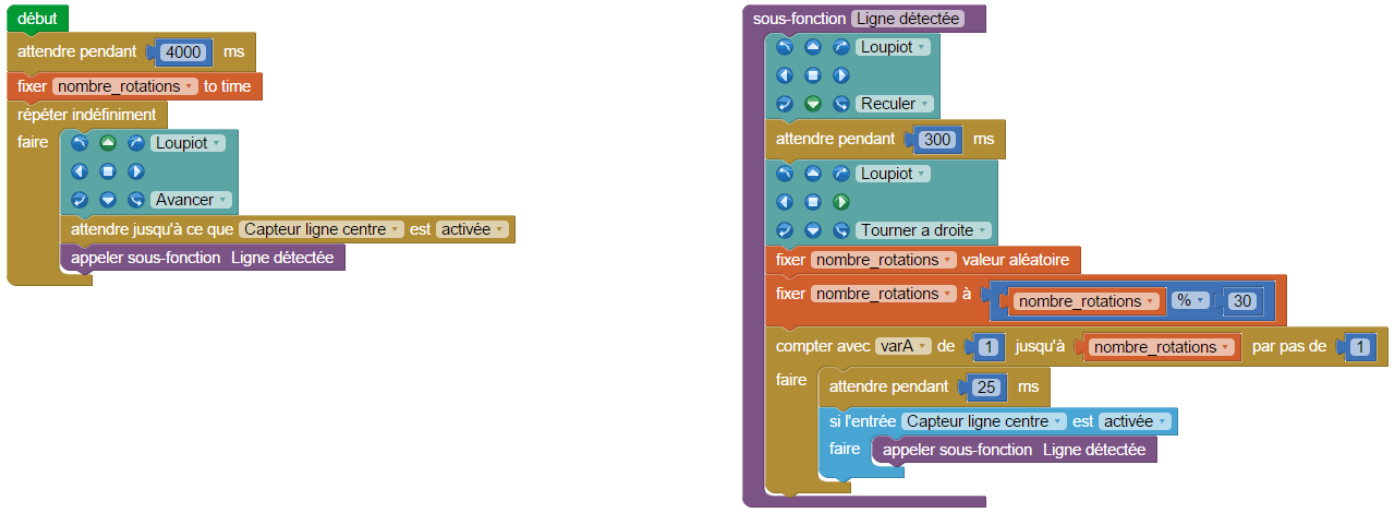
Exercice niveau 3 – D2 : Remplir au mieux une zone délimitée

Fichier modèle : LP_N3_D.xml

Objectif : Remplir au maximum une zone délimitée par un contour noir.
Note : partir du programme « LP-N2-C3 Prison » comme base de notre programme et l'améliorer pour optimiser la surface couverte par notre robot.

Correction :

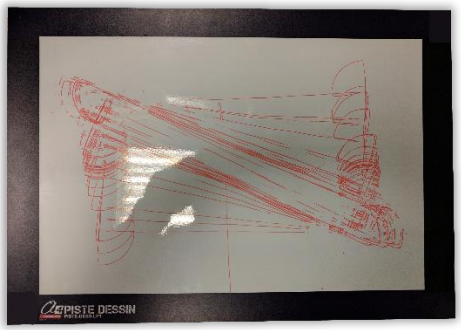
Blocs



Fichier Blockly : LP_N3_D2.xml

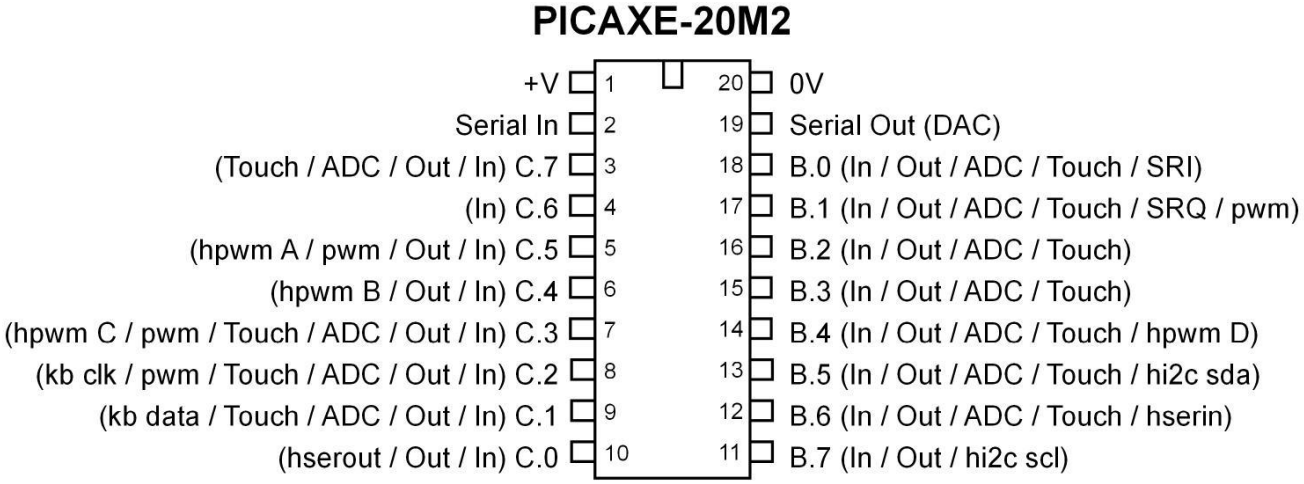
Remarque(s) :

- La variable « temps_rotations » va nous servir de variable aléatoire. Elle doit être initialisée par une valeur différente à chaque fois pour avoir une vraie suite de nombre aléatoire. C'est pourquoi on l'initialise avec le bloc « fixer temps_rotations to time » en début de programme. Ce bloc renvoie le temps depuis l'allumage. On considère que l'utilisateur n'appuiera jamais au même moment sur le bouton-poussoir du robot après l'avoir allumé pour lancer le programme.
- Pour améliorer le programme modèle LP-N2-C3 qui effectue toujours la même action quand il rencontre un bord noir, on introduit un angle de rotation aléatoire. Pour cela, on utilise la variable aléatoire « temps rotation » pour définir le nombre de répétitions de la boucle « compter avec varA … ».
- Voici ci-dessous deux photos montrant les résultats entre le programme de base LP-N2-C3 effectuant toujours la même action et le programme LP-N3-D2 incluant une action aléatoire. On voit clairement que le fait d'introduire une action aléatoire aide le robot à couvrir une plus grande surface.

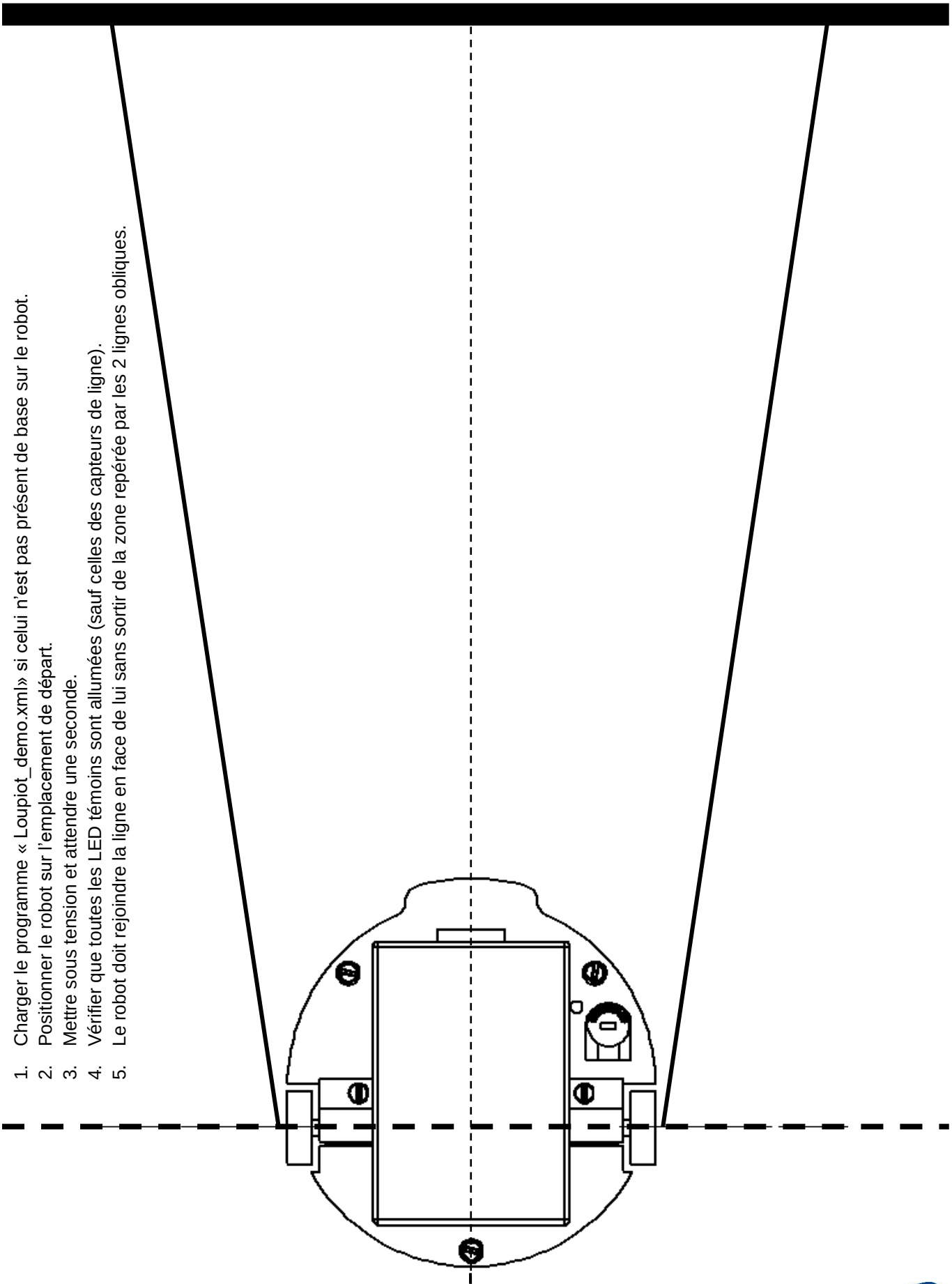


- On remarque que le programme pourrait être encore amélioré en retirant les valeurs aléatoires trop petites qui obligent le robot à tourner plusieurs fois de suite pour éviter une ligne noire. Voir l'exemple non corrigé LP-N3-D3.
- Le bloc « fixer … valeur aléatoire » renvoie une valeur comprise entre 0 et 65535. Or on veut que notre nombre de rotations varie seulement de 0 à 30 pour limiter les rotations du robot. On utilise pour cela le bloc modulo qui renvoie le reste de la division du nombre aléatoire par une valeur donnée (qui est ici de 30). Exemple 63 modulo 30 donne 3.

Schéma entrées / sorties Picaxe 20M2 :



Piste test + piste pour programmes démos



1. Charger le programme « Loupiot_demo.xml» si celui n'est pas présent de base sur le robot.
2. Positionner le robot sur l'emplacement de départ.
3. Mettre sous tension et attendre une seconde.
4. Vérifier que toutes les LED témoins sont allumées (sauf celles des capteurs de ligne).
5. Le robot doit rejoindre la ligne en face de lui sans sortir de la zone repérée par les 2 lignes obliques.

Prison (*version de base*)
Le robot reste à l'intérieur du périmètre délimité par la ligne noire.

Suivi de ligne (*version de base*)
Le robot suit la ligne noire.

Animation lumineuse (*version de base*)
Séquence de clignotement des LED.

Piste robot Loupiot v1

Programmes de démonstrations

Notice d'utilisation au verso de la feuille

Ressources disponibles sur www.a4.fr/wiki

Suivi de ligne avec capteur de proximité infrarouge *(option)*
Le robot suit la ligne jusqu'à la détection d'un obstacle.

Suivi de ligne avec capteur de distance ultrason *(option)*
Le robot suit la ligne jusqu'à la détection d'un obstacle.

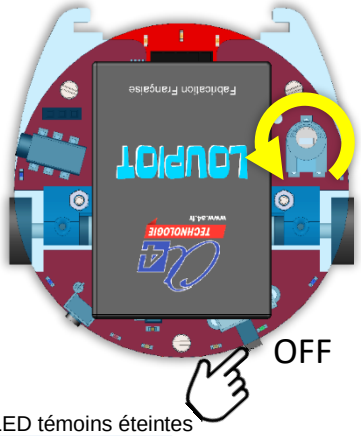
Bluetooth *(option)*
Piloter le robot à partir d'un smartphone avec l'application « Télécommande.apk » développée par A4 .

NOTICE D'UTILISATION DE LA PISTE DE DEMONSTRATION

Le robot loupiot est livré avec un programme de démonstration préchargé qui lui permet de lire des codes avec ses capteurs de ligne (voir au recto). Chaque code lance un programme spécifique (3 codes pour la version de base du robot et un code par option). Afin de tester ces programmes, veuillez suivre les étapes ci-dessous après avoir inséré 3 piles AAA dans le logement prévu à cet effet.

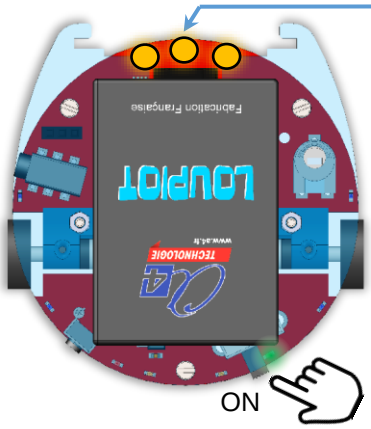
1. Régler les capteurs de ligne :

Les capteurs de ligne permettant de lire les codes sont sensibles à l'environnement lumineux et doivent être réglés avant chaque utilisation.



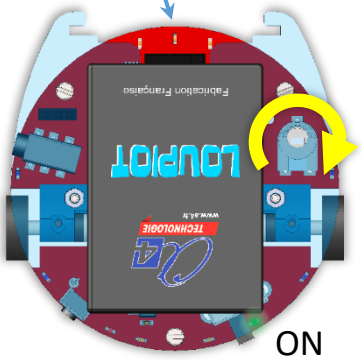
A - Positionner le robot sur la surface blanche où il va évoluer. Eteindre le robot. A l'aide d'un tournevis plat, tourner le potentiomètre totalement à gauche jusqu'à rencontrer la butée.

LED témoins des capteurs de ligne allumées



B - Allumer le robot : les 3 LED témoins des capteurs de ligne doivent être allumées.

LED témoins éteintes

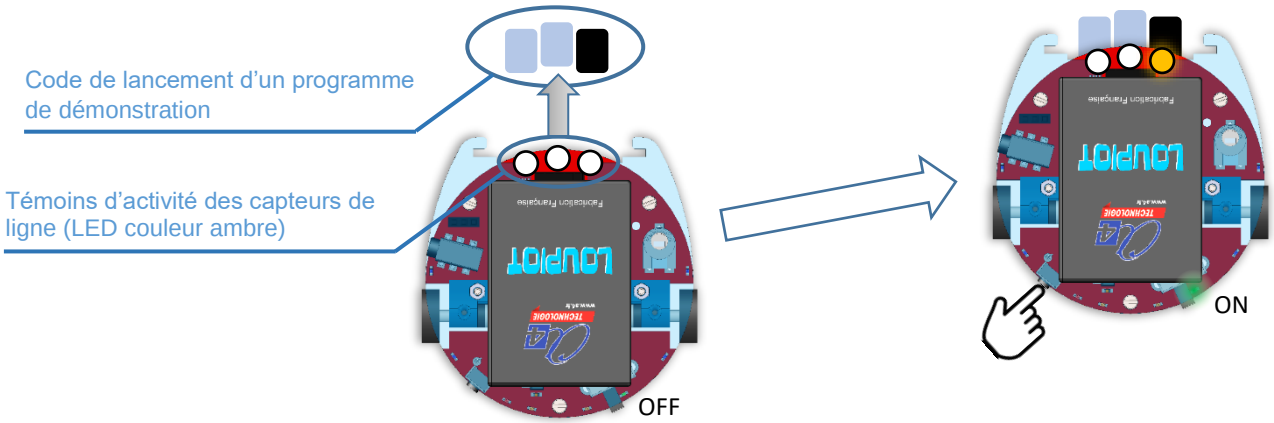


C - Tourner doucement le potentiomètre dans le sens inverse jusqu'à ce que les 3 LED s'éteignent.

D - Ajuster la sensibilité de détection en tournant le potentiomètre un peu plus à droite afin que les LED ne s'activent pas lorsque l'avant du robot est soulevé de 1 ou 2 mm (cela peut arriver quand il accélère de façon brutale). Pour tester le bon fonctionnement du réglage, placer les capteurs sur une ligne noire : les LED témoins doivent s'activer.

2. Lire un code et lancer un programme de démonstration :

A - Mettre le robot hors tension.
B - Positionner les capteurs de ligne sur un des codes au recto de la feuille.
C - Mettre le robot sous tension. Seules les LED témoins des capteurs de ligne au-dessus de pastilles noires doivent s'allumer.
D - Au bout d'une seconde, si un code a été lu, les témoins lumineux arrière clignotent. Le programme de démonstration sélectionné se lance après 3 secondes.



CONCEPTEUR ET FABRICANT DE MATÉRIELS PÉDAGOGIQUES