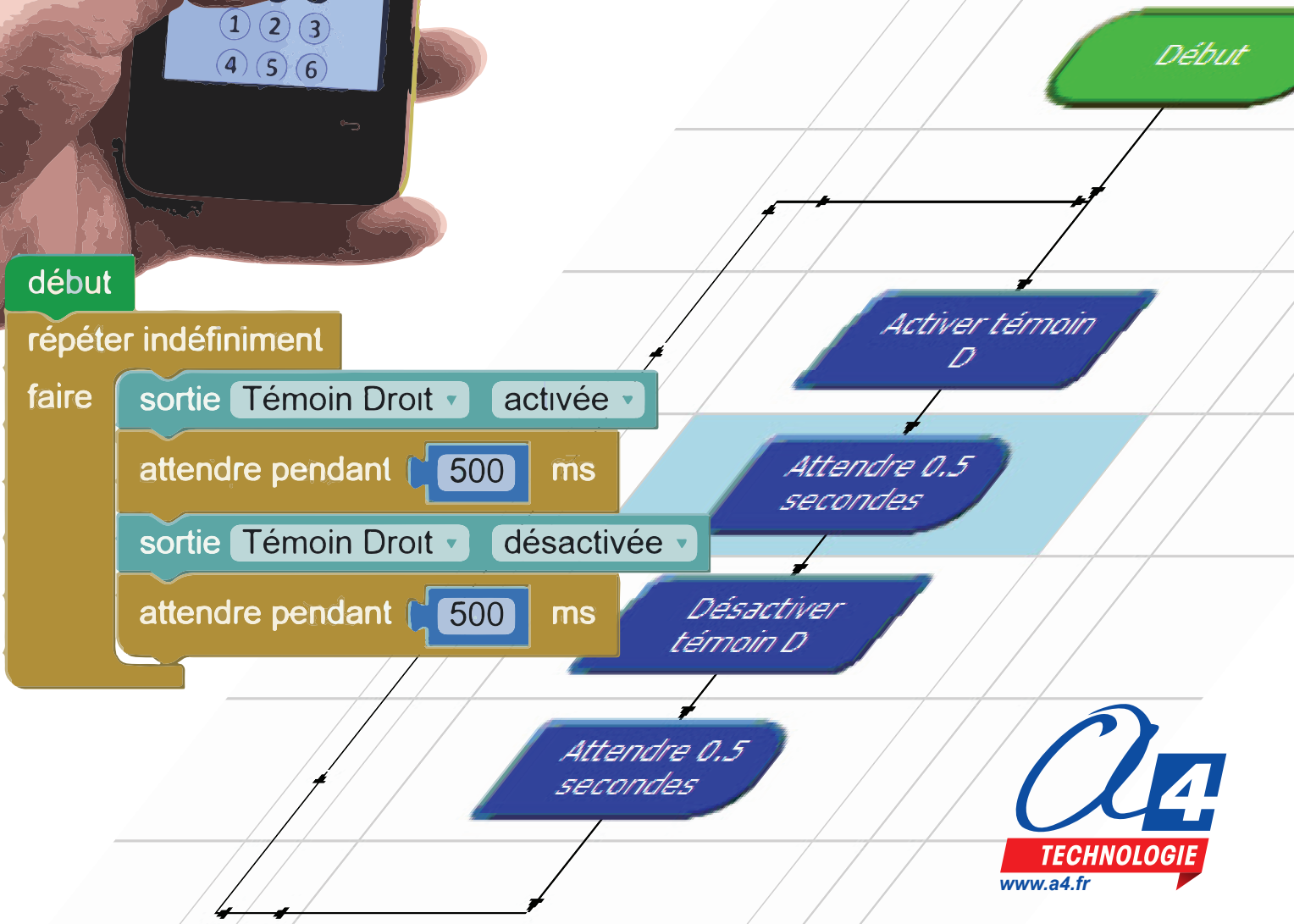
LOUPIOT V2

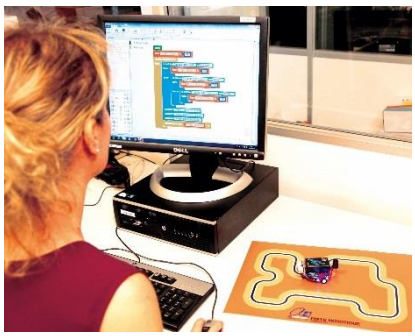
Robot programmable

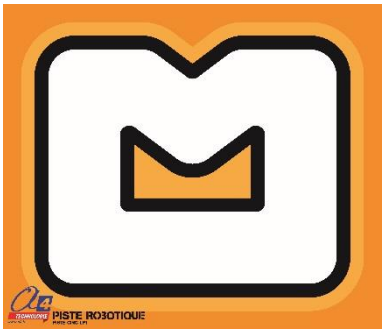


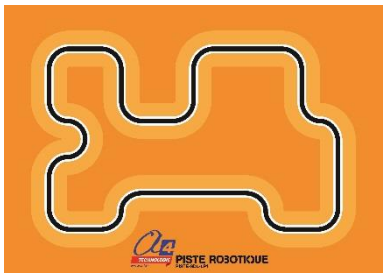
TECHNOLOGIE
www.a4.fr

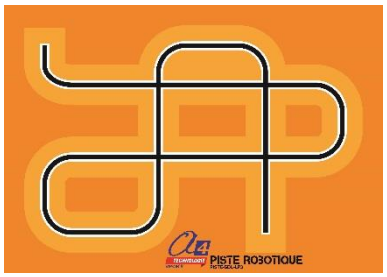
Activités autour du Loupiot

Activités sur pistes : suivi de ligne, détection d'obstacles, challenge robots aspirateurs …

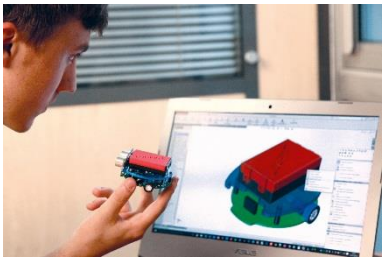








Impression 3 D – personnalisation du Loupiot



Ressources disponibles pour le projet K-LP-V2

Autour du projet Loupiot, nous vous proposons un ensemble de **ressources téléchargeables gratuitement sur le wiki : http://a4.fr/wiki/index.php/Loupiot**

Loupiot

- Fichiers SolidWorks du robot Loupiot et de ses options.
- Notice d'utilisation du robot Loupiot avec la piste de démonstration.
- Fichier STL pour la coque.

Logiciels Picaxe Editor 6 et App Inventor

- Procédure d'installation du driver pour le câble de programmation.
- Manuel d'utilisation Picaxe Editor 6.
- Notice d'utilisation App Inventor 2.

Activités / Programmation

- Fichiers modèles et fichiers de correction des programmes pour Picaxe EDITOR 6 (organigrammes et blocs) et AppInventor.
- Fichier pour la simulation Loupiot.
- Fichiers des différentes pistes utilisées dans les activités.

NOTE : Certains fichiers sont donnés sous forme de fichier.zip.

Logiciels, programmes, manuels utilisateurs téléchargeables gratuitement sur http://a4.fr/wiki/index.php/Loupiot

SOMMAIRE

Introduction

Le projet Loupiot s'adresse aussi bien à des utilisateurs totalement débutants (découverte progressive de la programmation par organigrammes et blocs) qu'à des utilisateurs avertis pour créer des scénarios de programmation élaborés allant jusqu'à l'interaction du robot avec un smartphone.

Le Loupiot

Loupiot a été conçu pour pouvoir être utilisé sur une petite surface, sur la table, à côté du poste informatique.
De base, il est équipé d'une détection de ligne haute sensibilité qui peut détecter des lignes tracées au feutre sur une feuille.
Il est également équipé de nombreux témoins lumineux qui permettent de visualiser l'activité des capteurs et des moteurs. La version 2 intègre un buzzer et un bouton-poussoir.
En option, Loupiot peut être équipé d'un module Bluetooth, d'un module infrarouge de détection d'obstacles, d'un télémètre à ultrasons et d'un porte-stylo.

Les environnements de programmation graphique

Tous les programmes correspondant aux activités menées autour du Loupiot ont été réalisés sous **PICAXE Editor 6**. En effet, ce logiciel de programmation graphique présente plusieurs **avantages** :

- Gratuit
- Blocs et organigrammes (proche algorigrammes).
- Personnalisation des noms des entrées/sorties.
- Personnalisation du jeu d'instructions.
- Mode de simulation visuelle à l'écran pour mettre au point et débugger les programmes.

Vous pouvez aussi utiliser **Blockly for Picaxe** : environnement de programmation par blocs simplifié (nombre de menus limité et personnalisation des entrées/sorties non disponibles).

Pour les activités menées avec un smartphone ou une tablette, les programmes et applications ont été réalisés sous **App Inventor 2**.
Il s'agit d'un environnement de développement pour concevoir des applications pour smartphone ou tablette Android.
Il a été développé par le MIT pour l'éducation. Il est gratuit et fonctionne via internet avec Blockly.

Le dossier

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.
Il est organisé en fonction des niveaux de programmation.

Niveau 1 :
Découverte progressive du jeu d'instructions et des fonctionnalités de base du Loupiot et maîtrise des principes fondamentaux pour concevoir un programme : séquences, boucles, structures conditionnelles (test) et variables.

Niveau 2 :
Approfondissement des principes de programmation abordés dans le niveau 1 en concevant des programmes plus élaborés qui répondent à des cas concrets d'utilisation du robot (version de base).

Niveau 3 :
Exemples d'utilisation des différentes options proposées autour du Loupiot : Bluetooth, télémètre à ultrasons, détection d'obstacles, porte-stylo.

Les fiches exercices

Pour chaque niveau de programmation, nous vous proposons des fiches exercices avec :
- un objectif : ce que doit faire le programme ;
- un fichier modèle : un programme vide avec un jeu d'instructions limité (suffisant pour réaliser l'exercice) ;
- un fichier de correction qui propose un exemple de programme réalisé sous Picaxe Editor 6 en blocs (extension .xml) et en organigrammes pour le niveau 1 uniquement (extension .plf).

Intérêt du fichier modèle :
- il évite aux utilisateurs de se perdre dans une multitude d'instructions ;
- il limite les propositions possibles ;
- il facilite la correction et l'analyse des erreurs.

Deux approches :

1. Avec les exemples de programmes, les utilisateurs découvrent les principes de la programmation graphique en organigrammes ou en blocs : chargement d'un programme, modification d'un programme et vérification sur le matériel (ex : modification des temps d'attente, etc.).
2. Les utilisateurs conçoivent eux-mêmes le programme pour atteindre l'objectif proposé, en organigrammes ou en blocs (à partir du fichier modèle). Ils peuvent ensuite le comparer au fichier de correction.

Principe de nommage des fichiers :

- **LP** pour Loupiot
- **N** : niveau de programmation 1-2-3
- **A-B-C** : jeu d'instructions du plus simple au plus avancé

Exemple : LP_N2_C3.xml
Correspond au niveau 2 avec le jeu d'instructions C, adapté aux objectifs « avancés » de ce niveau.

Prérequis

Pour la version de base :

- Installer le logiciel **Picaxe Editor 6** ou **Blockly for Picaxe :** http://www.picaxe.com/Software
- **Câble de programmation** Picaxe USB (Réf : CABLE-USBPICAXE).
- **3 piles** AAA de 1,5 Volts ou accu AAA 1,2 Volts.

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

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

Pour l'option Bluetooth :

- **Tablette ou smartphone** Android 5 ou + équipés de Bluetooth V3.
- Connexion internet pour accéder à **App Inventor :** http://ai2.appinventor.mit.edu/
- Compte Gmail requis.

Présentation du Loupiot

Version de base

Dimensions : Ø 75 x h. 38 mm. Roues du Loupiot : Ø 14 x 4,5 mm.

Microcontrôleur de type Picaxe 20M2.
16 entrées/sorties.
Capacité mémoire 2048 octets (env. 1000 lignes de programme).
Possibilité d'effectuer 8 tâches en parallèle.
Notice pour les microcontrôleurs Picaxe de type M2 : http://www.picaxe.com/docs/picaxem2.pdf
Schéma des entrées / sorties Picaxe 20M2 en annexe.

Capteurs
3 capteurs à réflexion infrarouge, à sensibilité réglable pour détecter des lignes noires fines (à partir de 2 mm de largeur).
Un bouton-poussoir.
Mesure de tension de l'alimentation pour intégrer la surveillance d'autonomie du robot dans un programme.
Horloge pour intégrer une mesure de temps depuis la mise sous tension.

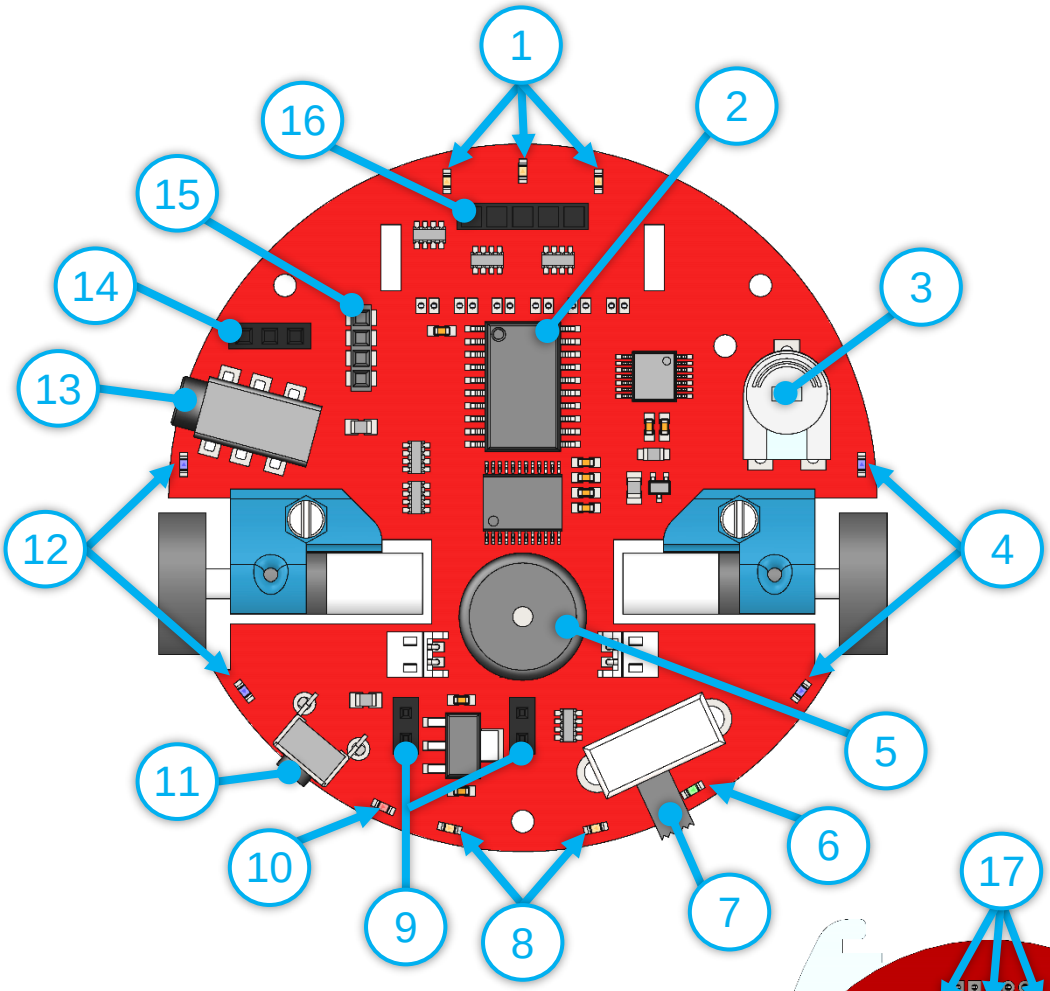
Actionneurs
3 témoins LED programmables.
Un buzzer pour jouer des notes de musique.
Deux moteurs avec vitesse et sens de rotation réglables + témoins de visualisation de leur sens de rotation.

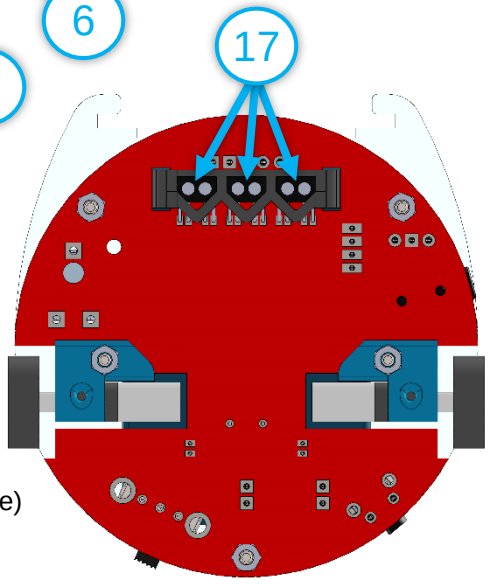
Caractéristiques des moteurs
Moteur à courant continu.
Dimensions : Ø 6 x L 19 mm.
Rapport de réduction : 136 :1.
500 tr/min à vide à 6 V.
Couple à l'arrêt du moteur : 550 g.cm à 6 V.
Consommation 45 mA à 6 V à vide.

Note :
- le buzzer et le témoin lumineux droit partagent une même entrée/sortie du microcontrôleur.
- le bouton-poussoir et le témoin lumineux gauche partagent une même entrée/sortie du microcontrôleur.

= Il n'est pas possible d'exploiter simultanément les fonctionnalités partageant une même entrée/sortie.

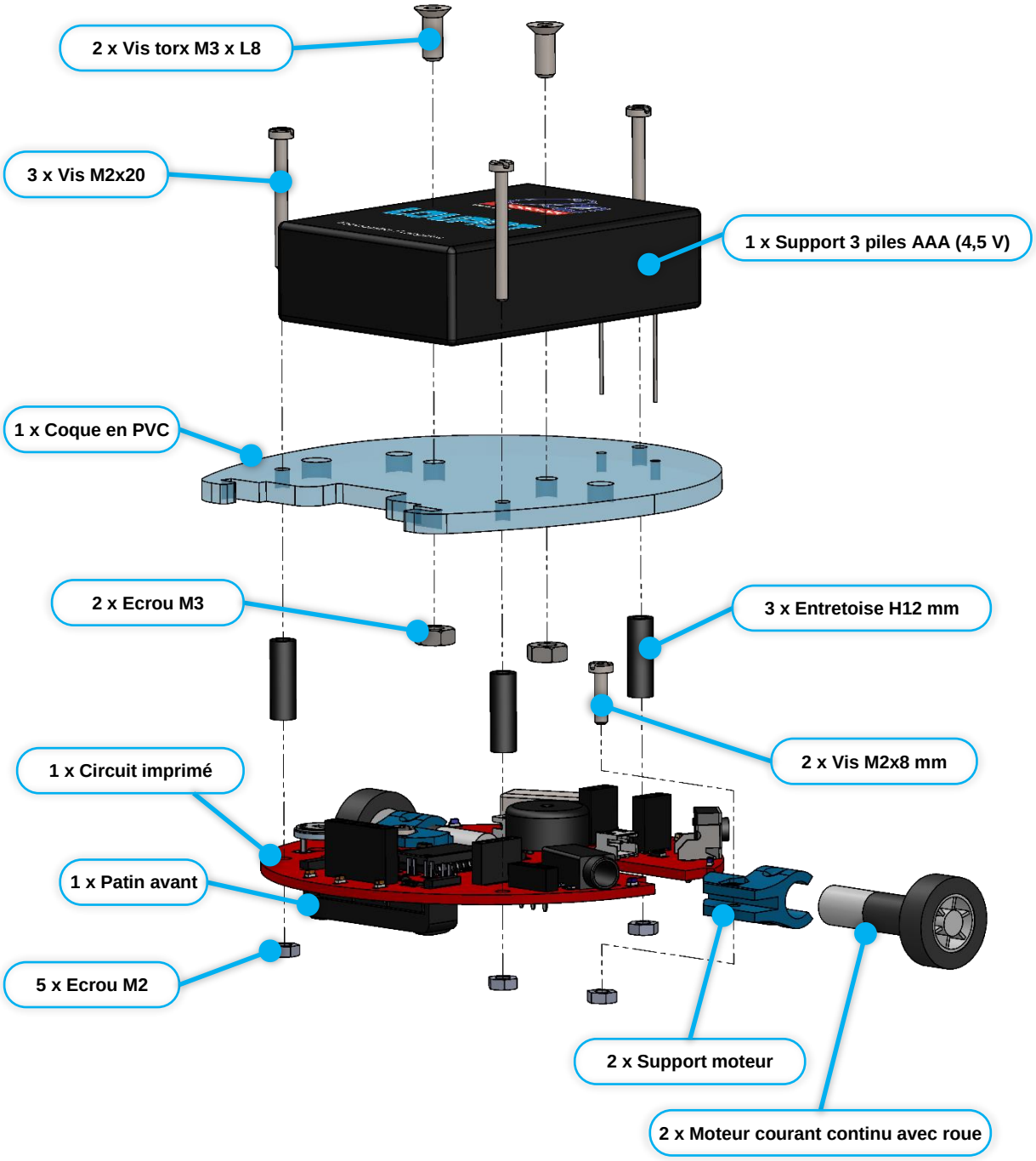
Composants principaux du circuit (vue de dessus et dessous)



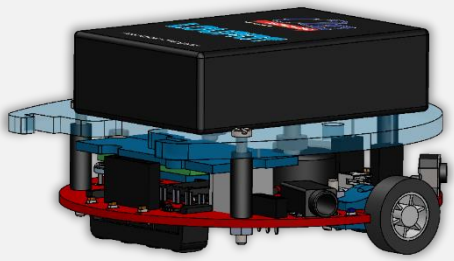


1) LED témoins activité capteurs de ligne
2) Microcontrôleur PICAXE 20M2
3) Potentiomètre pour réglage sensibilité capteurs de ligne
4) LED témoins du sens de rotation moteur droit
5) Buzzer
6) LED témoin de mise sous tension
7) Interrupteur M/A
8) LED témoins lumineux programmable droit et gauche
9) Connecteur d'alimentation (4,5 V)
10) LED témoin lumineux programmable alerte (ex : batterie faible)
11) Bouton-poussoir
12) LED témoins du sens de rotation moteur gauche
13) Connecteur câble de programmation
14) Connecteur module élévateur de tension pour option télémètre à ultrasons
15) Connecteur option Bluetooth
16) Connecteur option télémètre à ultrasons ou détection d'obstacle
17) Capteurs de ligne (droit, centre, gauche)

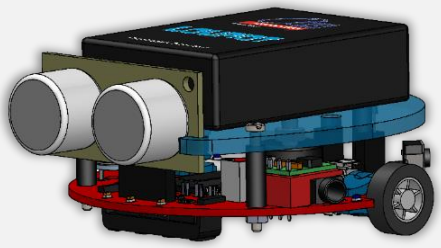
Eclaté du robot – Nomenclature (version de base)



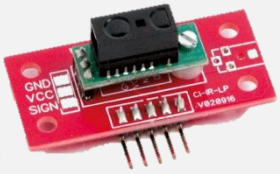
Options



Bluetooth :
Module Grove permettant la communication avec un smartphone ou une tablette Android.
Réf : S-113020008



Télémètre à ultrasons :
Renvoie une valeur de distance entre 0 et 2,5 m. (Valeur du capteur dans le vide, env. 0 à 0,75 cm quand placé sur le robot).
Réf : K-LP-US



Détection d'obstacles :
Un capteur de proximité infrarouge prévient quand un obstacle est détecté à moins de 5 cm.
Réf : K-LP-IR

Porte-stylo :
Pour positionner un stylo ou un feutre (Ø 9,5 mm) à l'arrière du Loupiot pour tracer des lignes.
Réf : K-LP-PS

Nous proposons des feutres et des pistes effaçables pour réaliser les activités.

Environnement de programmation graphique

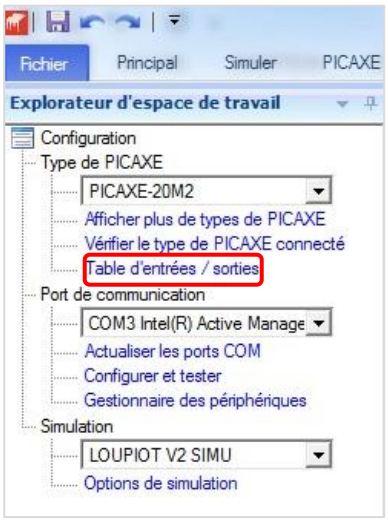
Tous les programmes correspondant aux activités menées autour du Loupiot ont été réalisés sous **PICAXE Editor 6**. En effet, ce logiciel de programmation graphique présente plusieurs avantages :

- Gratuit
- Blocs et organigrammes (proche algorigrammes).
- Personnalisation des noms des entrées/sorties.
- Personnalisation du jeu d'instructions.
- Mode de simulation visuelle à l'écran pour mettre au point et débugger les programmes.

Note : vous pouvez aussi utiliser **Blockly for Picaxe** : environnement de programmation par blocs simplifié (nombre de menus limité et personnalisation des entrées/sorties non disponibles).

Personnalisation des entrées/ sorties

Nous vous proposons le fichier **Loupiot_base.xml** dans lequel les noms des entrées/sorties ont été personnalisés pour une utilisation avec le Loupiot. Tous les programmes et activités proposés dans ce document se basent sur cette liste. Celle-ci reste modifiable à tout moment.
A partir de Picaxe Editor 6, dans l'explorateur d'espace de travail cliquer sur **Table d'entrées / sorties**.



Une fenêtre apparaît à partir de laquelle vous pouvez modifier les noms de toutes les entrées et sorties dans la zone « Mon étiquette ».

Valider en cliquant sur **OK**.

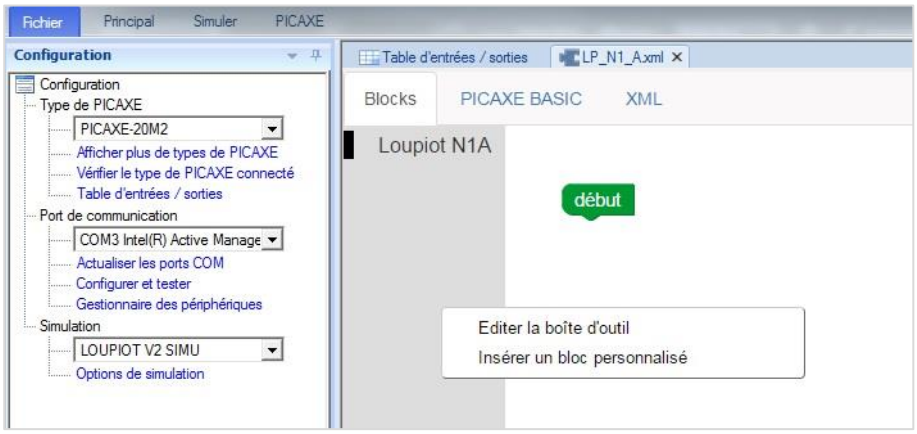
Table personnalisée des entrées et sorties du Loupiot

Etiquette Blockly	N° Broche	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 ultrasons / Détection d'obstacles à moins de 5 cm (OPTIONS)
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 *(voir chapitre « Instruction spéciale Moteurs » et annexe)*		
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

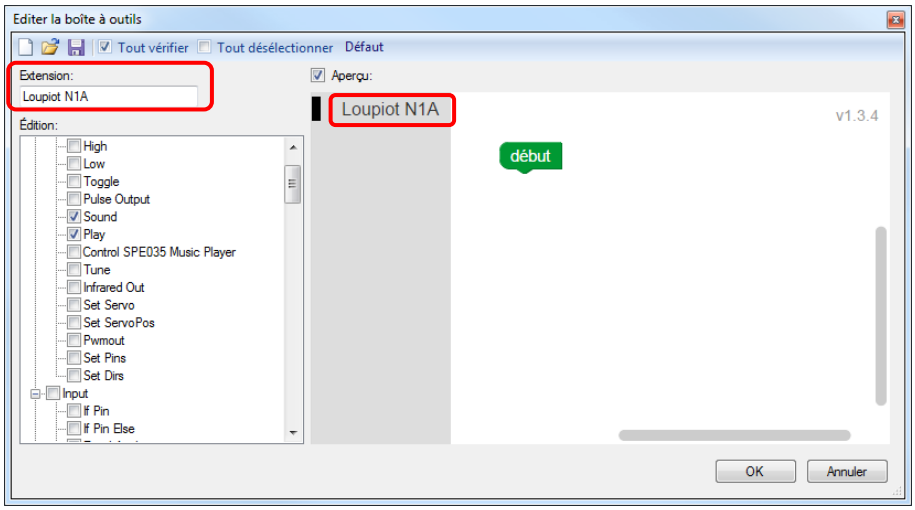
** Options du Loupiot.*

Personnalisation du jeu d'instructions

Vous pouvez personnaliser l'affichage du jeu d'instructions pour en limiter la quantité afin de faciliter la l'analyse et la correction des erreurs. Faire un clic droit sur la zone des blocs puis cliquer sur **Editer la boite d'outil**.



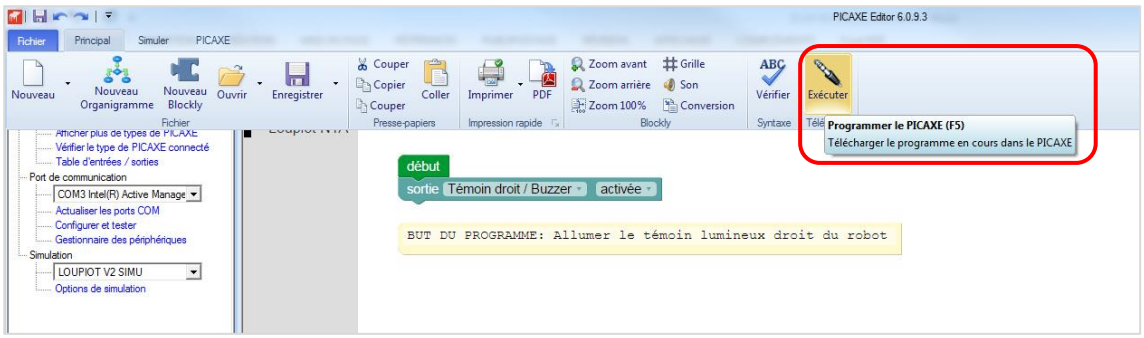
Une fenêtre s'ouvre à partir de laquelle vous pouvez sélectionner ou désélectionner les instructions de votre choix. Vous pouvez renommer le jeu d'instructions dans la zone « **Extension** ».



Valider en cliquant sur **OK**.

Procédure de chargement d'un programme

Commencer par relier le Loupiot à l'ordinateur avec le câble de programmation USB et le mettre sous tension.
A partir du Picaxe Editor 6, ouvrir un programme.



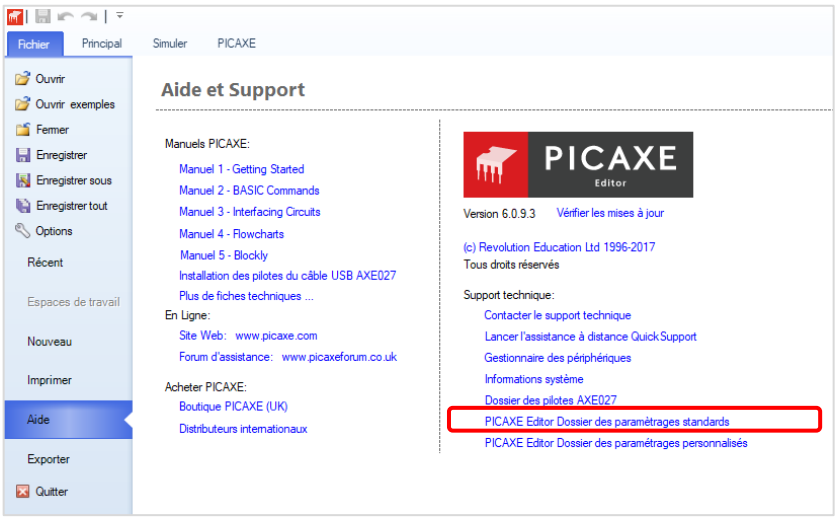
A partir du menu **Principal** ou du menu **PICAXE**, cliquer sur le bouton **Exécuter**. Vous pouvez également utiliser la touche **F5** de votre clavier.
Note : un programme téléchargé écrase le précédent.

Mode simulation

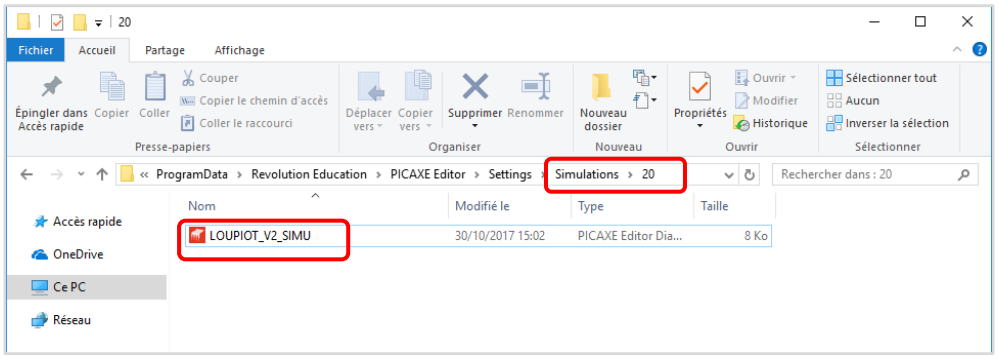
La simulation sur Picaxe EDITOR 6 permet de tester un programme avant de le téléverser dans le robot. Nous vous proposons une simulation propre au Loupiot (fichier **LOUPIOT_V2_SIMU.edd**). Elle n'est pas incluse de base sur le logiciel, il faut la télécharger et l'installer.

Installation de la simulation pour le Loupiot

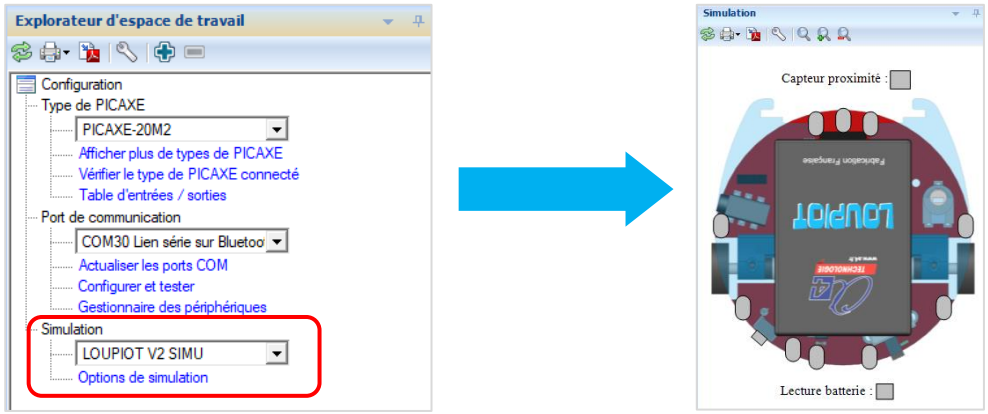
A partir de Picaxe EDITOR 6, dans le menu **Fichier / Aide**, cliquer sur « **Picaxe Editor Dossier des paramètres standards** ».



Aller dans **Simulations / 20** et placer le fichier « **LOUPIOT_V2_SIMU.edd** » préalablement téléchargé.

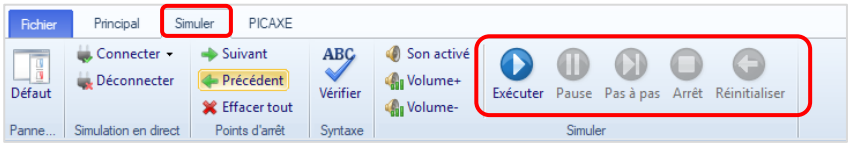


Relancer Picaxe Editor et aller dans l'onglet « Explorateur d'espace de travail » à gauche de l'écran.
Dans la rubrique **Simulation**, sélectionner « LOUPIOT_V2_SIMU » dans la liste déroulante.
Une image du robot apparait en dessous de l'onglet **Simulation**.



Lancer une simulation

Pour lancer et contrôler une simulation, utiliser les boutons **Exécuter / Pause / Pas à pas / Arrêt** à partir du menu **Simuler**.



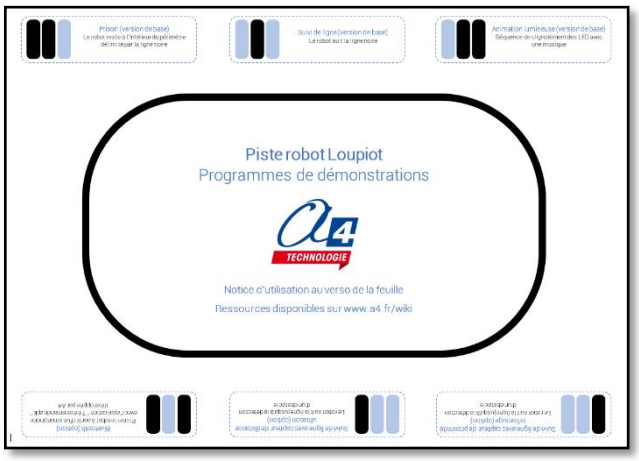
Ouvrir le programme « N1-A2.xml » et lancer la simulation pour tester l'activation / désactivation d'une sortie.
Ouvrir le programme « N1-C3.xml » et lancer la simulation pour tester l'activation / désactivation manuelle d'une entrée.
La simulation surligne les blocs dans l'espace de travail pour vous montrer où en est le programme.

Si vous cliquez sur les pastilles du Loupiot, cela active ou désactive ses entrées/sorties.
Vous pouvez le faire même durant la simulation.
Si lors de la simulation d'un code, un bloc désactive ou active une sortie alors celle-ci se modifie en temps réel sur les pastilles de l'image du Loupiot dans l'onglet **Simulation**.

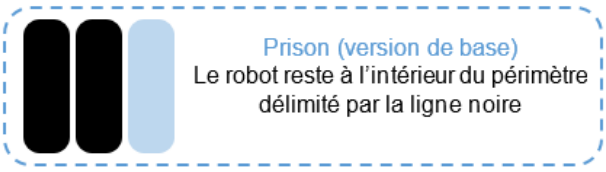
Mise en service du Loupiot

Le robot Loupiot est livré préprogrammé avec le fichier **Loupiot_demo.xml** et une piste de démonstration.
Plusieurs programmes de démonstration vous sont proposés :

- 3 pour la version de base du Loupiot
- un pour chaque option : Bluetooth, télémètre à ultrasons, détection d'obstacles.



Exemple de programme :



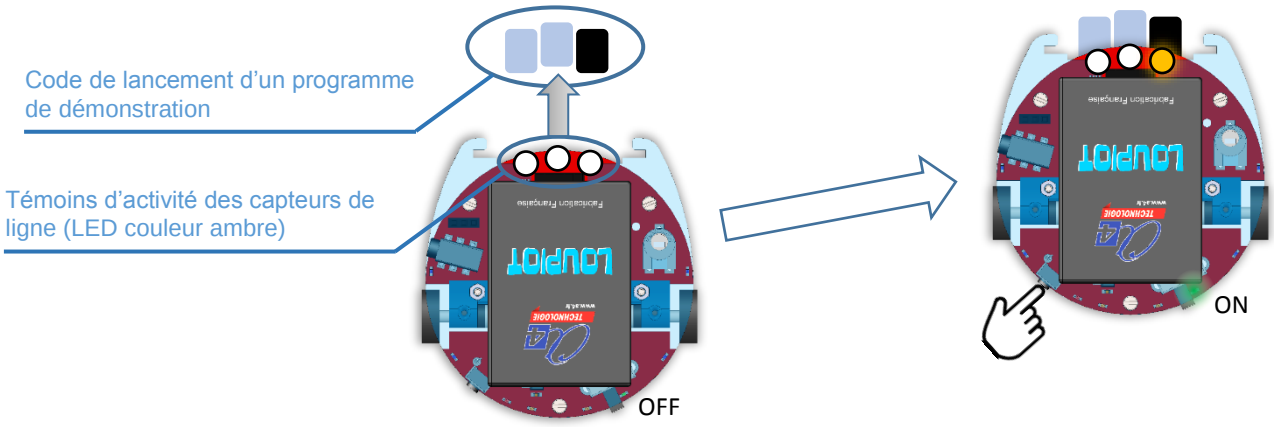
Pour garantir le bon fonctionnement du robot, il est indispensable d'ajuster correctement la sensibilité de détection des capteurs de lignes.

Programmes de démonstration

Pour lancer les programmes de démonstration des fonctionnalités du Loupiot :

1 - Mettre le robot sous tension.
2 - Positionner les capteurs de ligne sur un des codes.
Les LED témoins des capteurs de ligne au-dessus de pastilles noires doivent s'allumer.
3 - Appuyer sur le bouton-poussoir pour lancer le programme de démonstration sélectionné.
4 - Le buzzer commence à sonner. Le programme de démonstration sélectionné se lance après 3 secondes.

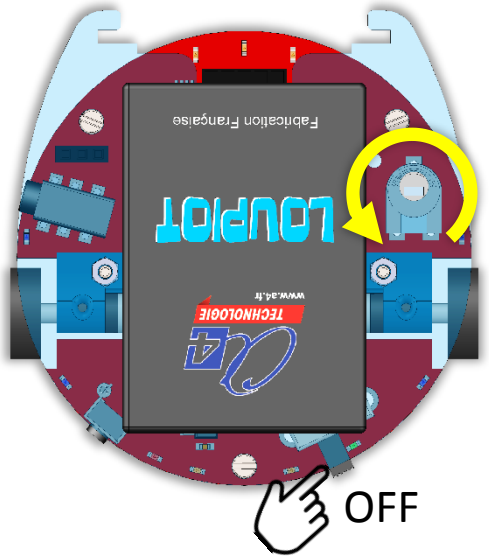
Pour essayer un autre programme de démonstration, éteindre le Loupiot et recommencer la procédure.



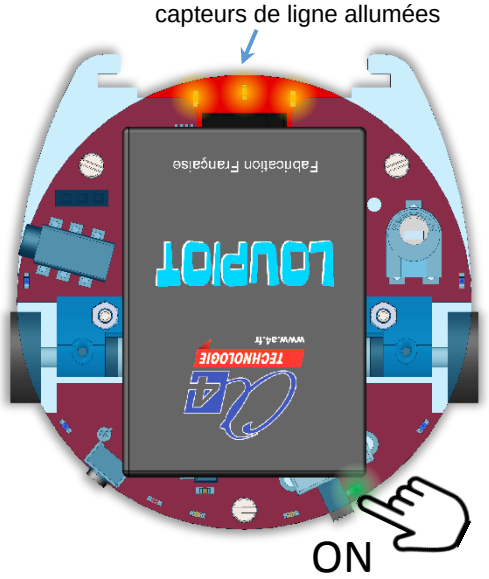
Note : Si le programme de base est écrasé, il est possible de le recharger dans le Loupiot.
Ce programme est disponible en téléchargement libre sur www.a4.fr.

Réglage des capteurs de ligne

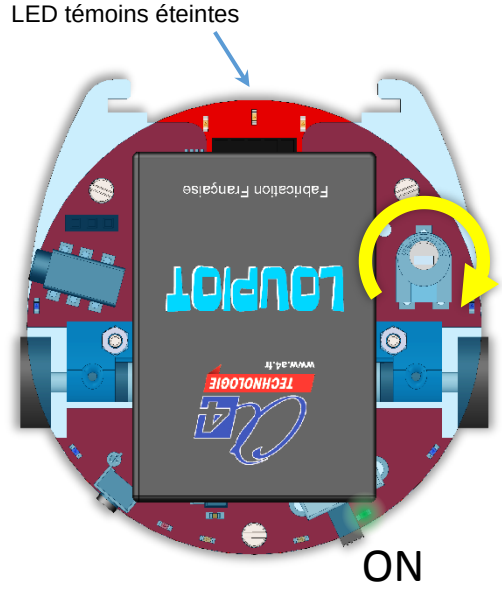
Le robot est livré préréglé. Cependant, les capteurs de ligne (situés à l'avant du robot sous leur LED témoin) sont sensibles au transport et à l'environnement lumineux. Il peut arriver que vous ayez à les re-régler. Pour cela, il suffit de suivre les étapes suivantes :



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.

Programmation niveau 1 (version de base)

Niveau 1 : découverte progressive du jeu d'instructions et des fonctionnalités de base du Loupiot et maîtrise des principes fondamentaux pour concevoir un programme : séquences, boucles, structures conditionnelles (test) et variables.

Liste des programmes Niveau 1

Nom du fichier	Description	Objectif
Niveau 1 A **Fichier modèle :** LP_N1_A.xml		
LP_N1_A1	Activer un témoin lumineux	Fonctionnalité matérielle abordée : - Allumage / extinction des témoins lumineux - Jouer un son avec le buzzer Notions de programmation abordées : - Séquence d'instructions - Temps d'attente - Boucle infinie et Boucle « For » - Activer / désactiver une sortie
LP_N1_A2	Activer / désactiver un témoin lumineux	
LP_N1_A3	Faire clignoter un témoin lumineux	
LP_N1_A4	Faire clignoter deux témoins lumineux en alternance	
LP_N1_A5	Faire clignoter un témoin lumineux 5 fois (méthode 1)	
LP_N1_A6	Faire clignoter un témoin lumineux 5 fois (méthode 2)	
LP_N1_A7	Jouer une musique	
LP_N1_A8	Faire biper le buzzer	
Niveau 1 B **Fichier modèle :** LP_N1_B.xml		
LP_N1_B1	Avancer	Fonctionnalité matérielle abordée : - Gestion des moteurs
LP_N1_B2	Avancer puis s'arrêter	
LP_N1_B3	Tourner à droite puis à gauche	
LP_N1_B4	Tourner en rond	
LP_N1_B5	Mouvement répété	
LP_N1_B6	Accélération brutale	
Niveau 1 C **Fichier modèle :** LP_N1_C.xml		
LP_N1_C1	Recopier l'état d'une entrée	Fonctionnalité matérielle abordée : - Lecture des entrées du robot (capteur de ligne / lecture batterie / bouton-poussoir) Notions de programmation abordées : - Debug - Structures conditionnelles
LP_N1_C2	Recopier l'état de plusieurs entrées	
LP_N1_C3	Avancer jusqu'à la ligne 1 (une condition)	
LP_N1_C4	Avancer jusqu'à la ligne 2 (conditions imbriquées)	
LP_N1_C5	Lecture batterie / Debug	
LP_N1_C6	Appuyer 4 fois sur le BP pour jouer une musique	

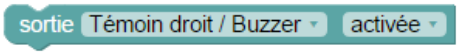
Exercice niveau 1 - A1 : Activer un témoin lumineux

Fichier modèle : LP_N1_A.xml

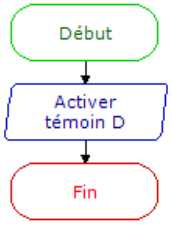
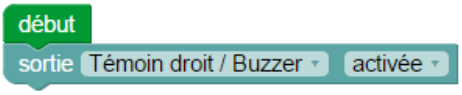
Objectif : allumer le témoin droit du robot.

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

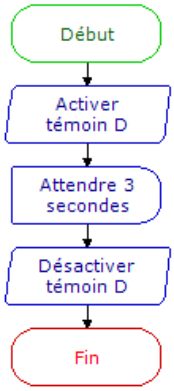
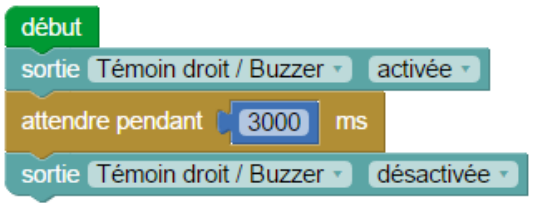
Fichier modèle : LP_N1_A.xml

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 suit.
- En programmation par blocs, la dernière instruction exécutée marque la fin du programme.

Exercice niveau 1 - A3 : Faire clignoter un témoin lumineux

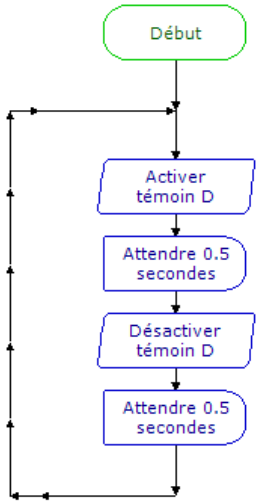
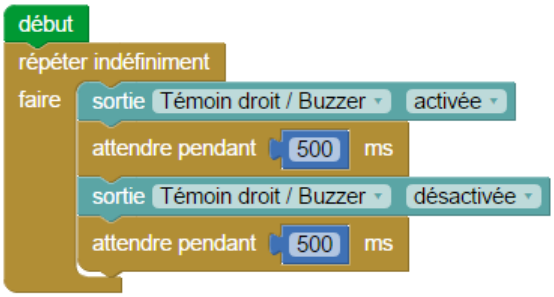
Fichier modèle : LP_N1_A.xml

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 le bloc « **répéter indéfiniment** » (boucle infinie) 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

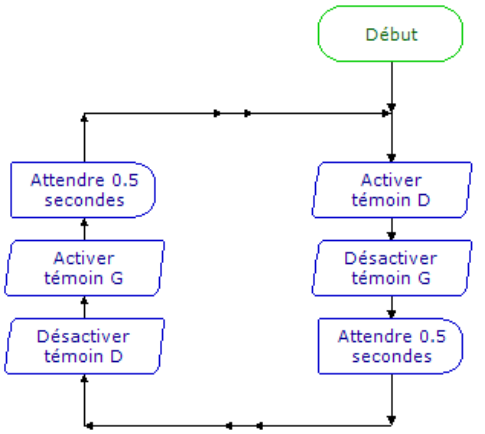
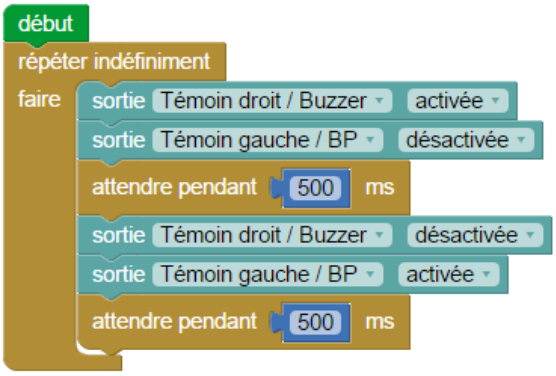
Fichier modèle : LP_N1_A.xml

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 - Méthode 1

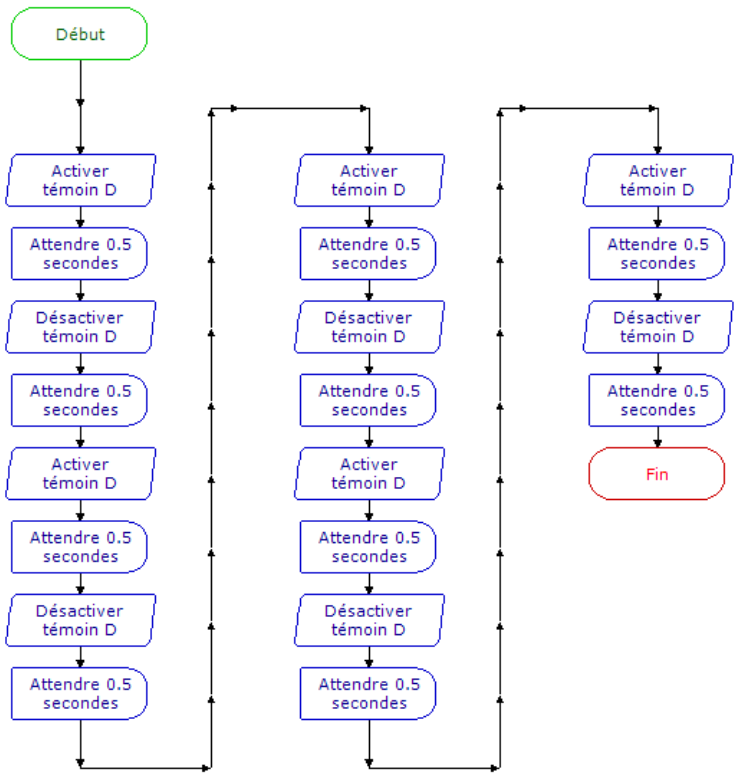
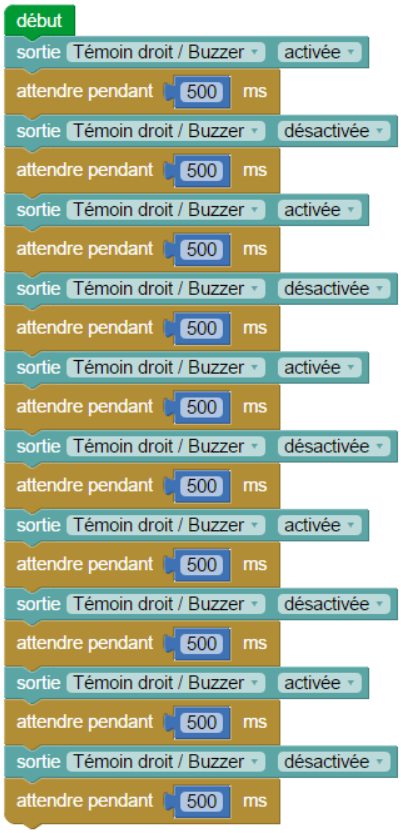
Fichier modèle : LP_N1_A.xml

Objectif : faire clignoter le témoin droit 5 fois avec un intervalle de 1 seconde.

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

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A5_Organigramme.plf	Fichier Blockly : LP_N1_A5.xml

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 - Méthode 2

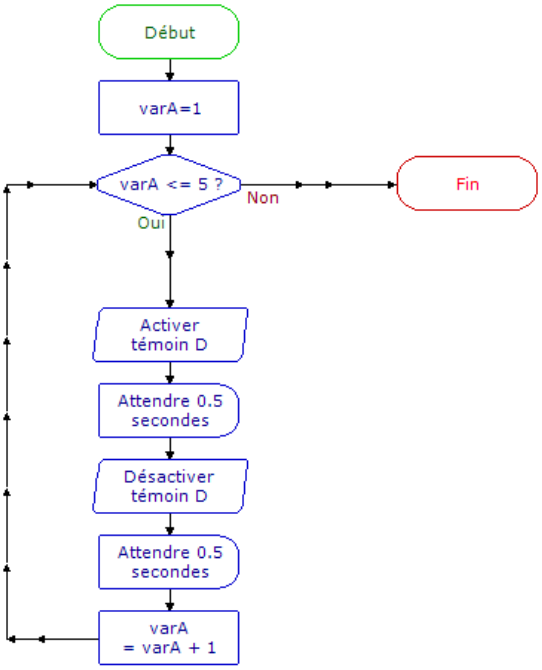
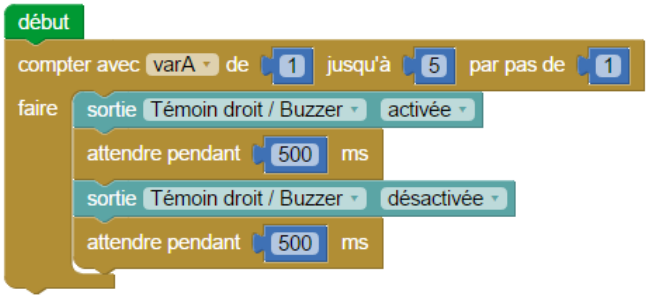
Fichier modèle : LP_N1_A.xml

Objectif : faire clignoter le témoin droit 5 fois avec un intervalle 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 - A7 : Jouer une musique

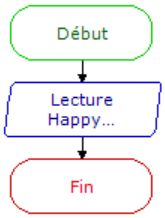
Fichier modèle : LP_N1_A.xml

Objectif : jouer une musique sur le buzzer.

Notion(s) abordée(s) : activer / désactiver une sortie à une fréquence donnée.

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A7_Organigramme.plf	Fichier Blockly : LP_N1_A7.xml

Remarque(s) :

- Le buzzer et le témoin droit sont branchés sur la même entrée/sortie du microcontrôleur. On ne peut donc pas utiliser les deux fonctionnalités en même temps. Ici, quand on choisit d'utiliser le buzzer, le contrôle de la LED devient impossible mais elle s'allume en fonction de la musique jouée.

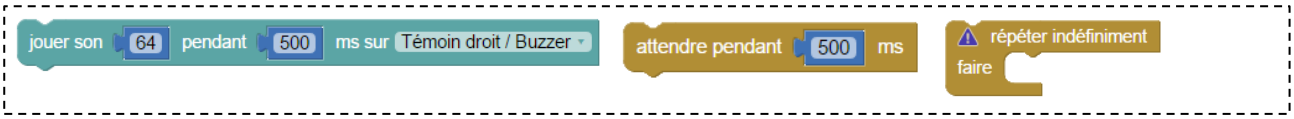
Exercice niveau 1 - A8 : Faire biper le buzzer

Fichier modèle : LP_N1_A.xml

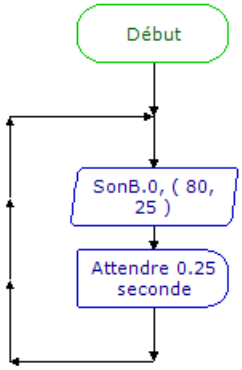
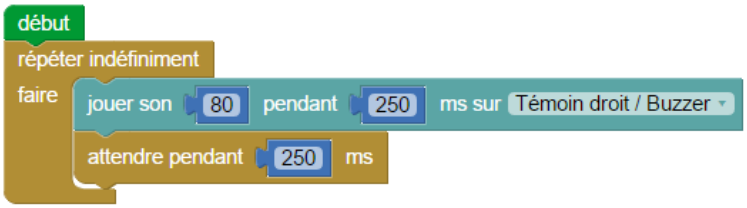
Objectif : créer un bip audible et répété indéfiniment.

Notion(s) abordée(s) : faire biper le buzzer.

Instruction(s) utilisée(s) :



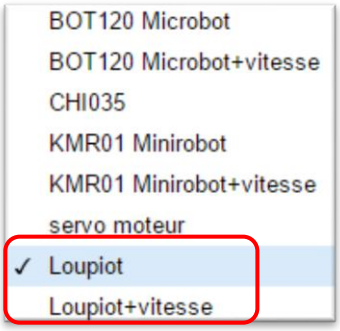
Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_A8_Organigramme.plf	Fichier Blockly : LP_N1_A8.xml

Remarque(s) :

- Le bloc « **jouer son … pendant … ms sur** ... » est conçue pour créer des bips audibles pour les jeux et les claviers par exemple.
- Pour jouer de vraies notes de musique, comme dans l'exemple précédent il faudra utiliser une autre instruction non proposée dans le fichier modèle (voir l'exemple du niveau 2 : Jouer la musique de Star Wars).

Instruction spéciale Moteurs



La rubrique « **Moteurs** » de la boîte à outils propose deux blocs pour gérer les déplacements du Loupiot :

- « **Loupiot** » pour gérer le sens de déplacement du robot.
- « **Loupiot + vitesse** » pour modifier la vitesse de fonctionnement par défaut du robot (128).

Contrôle du déplacement

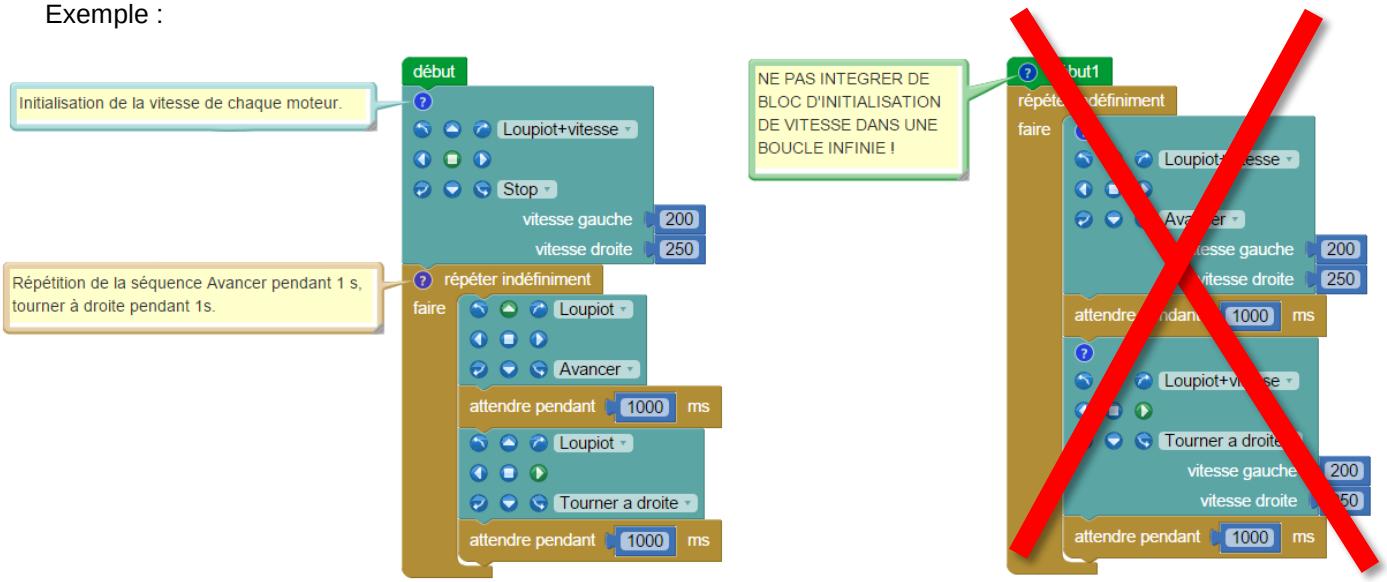
Une liste déroulante donne accès à des consignes de déplacement explicites « **Stop** », « **Avancer** », …
La consigne « **Tourner** » signifie que le robot fait une rotation centrée entre ses 2 roues (un moteur activé dans un sens, l'autre dans le sens opposé).
La consigne « **Virer** » signifie que le robot fait une rotation centrée sur une roue (un moteur activé, l'autre arrêté).
Par défaut, la vitesse de rotation de chaque moteur du robot est de 128 sur une échelle de 0 à 255. L'instruction « Loupiot + vitesse » ci-après peut être utilisée pour modifier cette valeur.

Contrôle de la vitesse

Ce bloc est réservé à la modification de la vitesse de déplacement du robot.
Par défaut, la vitesse est paramétrée à 128 pour chaque moteur.
La consigne de vitesse minimum est 80 et la consigne de vitesse minimum est 255.

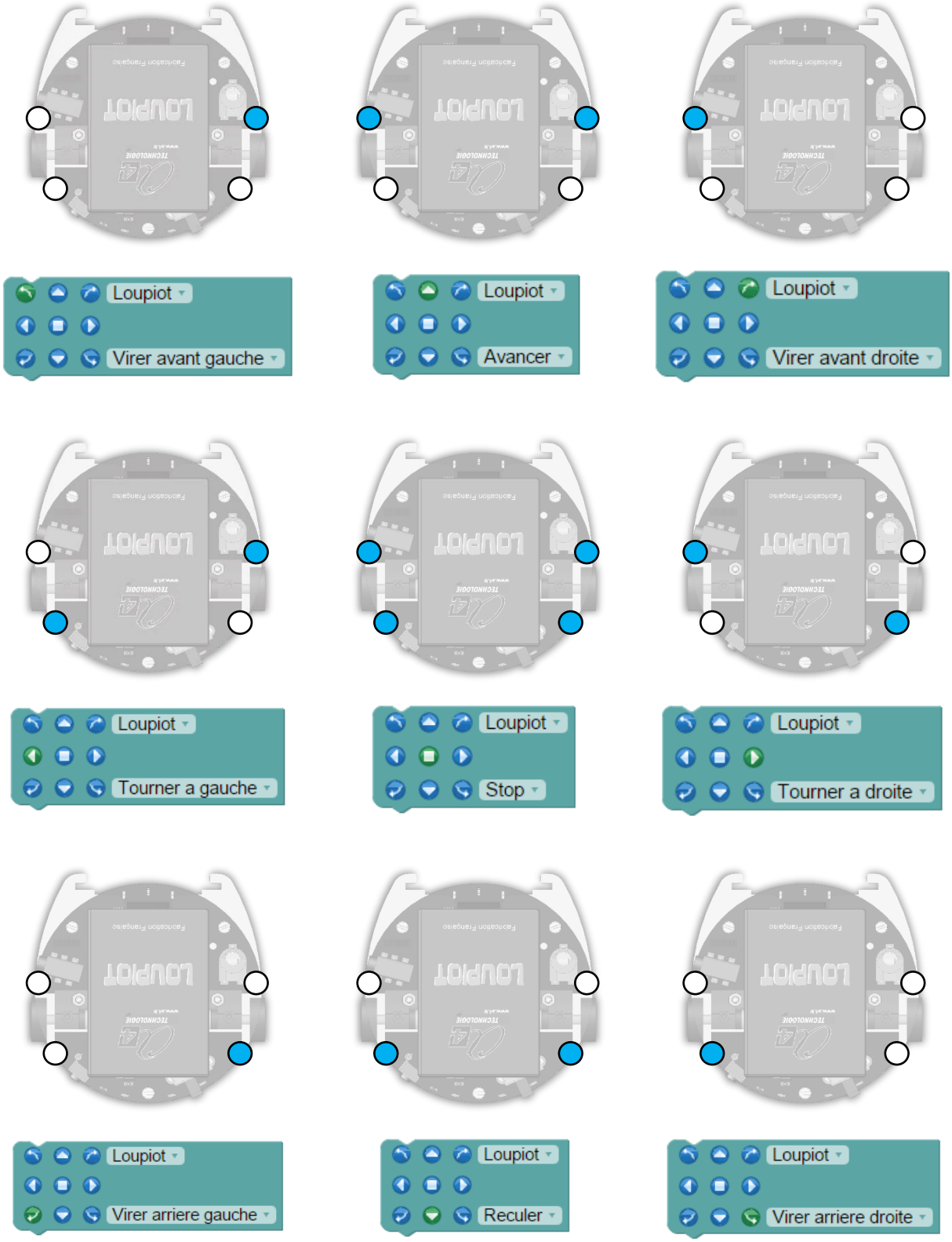
IMPORTANT !
Ce bloc s'utilise de préférence en début de programme juste après l'instruction « Début ».
La modification de vitesse des moteurs nécessite un certain temps avant d'être stabilisée.
Il faut éviter d'introduire l'instruction « Loupiot + vitesse » dans une boucle infinie en répétant son exécution fréquemment.
En effet, on risquerait d'observer un fonctionnement instable et erratique des moteurs.

Exemple :



Le tableau ci-dessous 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.



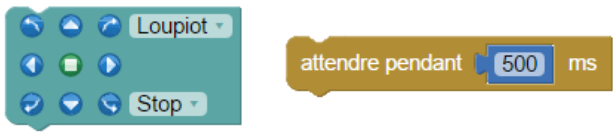
Exercice niveau 1 - B1 : Avancer

Fichier modèle : LP_N1_B.xml

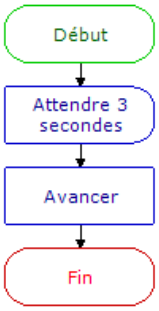
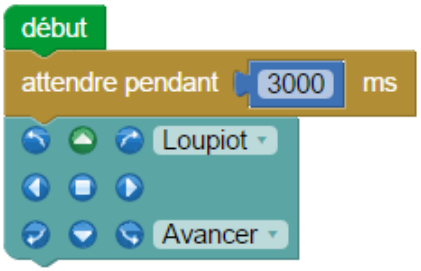
Objectif : faire avancer le robot au bout de trois secondes après la mise sous tension.

Notion(s) abordée(s) : utiliser l'instruction « Moteurs ».

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 placée en début de programme vous laisse le temps 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.

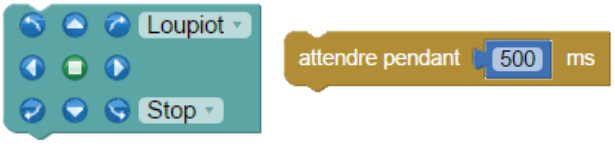
Exercice niveau 1 - B2 : Avancer puis s'arrêter

Fichier modèle : LP_N1_B.xml

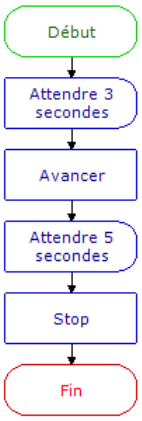
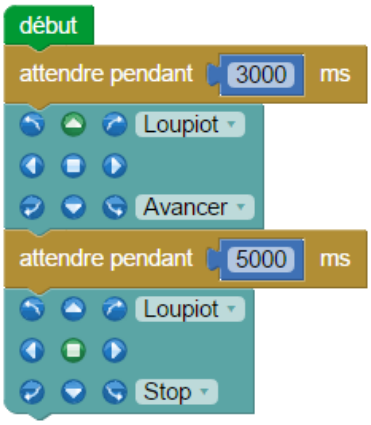
Objectif : Au bout de 3 secondes, faire avancer le robot pendant 5 secondes puis l'arrêter.

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) :

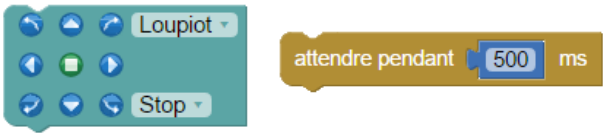
Exercice niveau 1 - B3 : Tourner à droite puis à gauche

Fichier modèle : LP_N1_B.xml

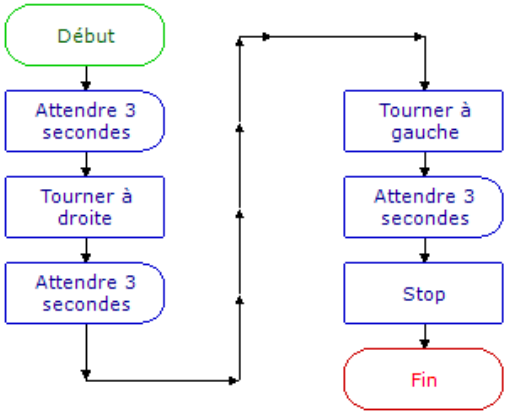
Objectif : Au bout de 3 secondes, faire tourner le robot sur lui-même à droite pendant 3 secondes, puis à gauche pendant 3 secondes, puis l'arrêter.

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).

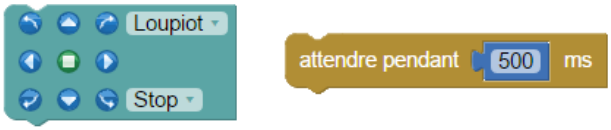
Exercice niveau 1 - B4 : Tourner en rond

Fichier modèle : LP_N1_B.xml

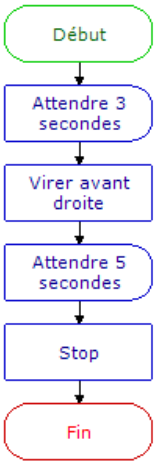
Objectif : faire tourner le robot en rond pendant 5 secondes puis l'arrêter.

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) :

- 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).

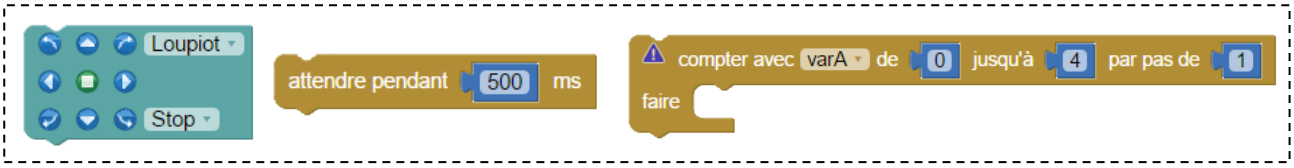
Exercice niveau 1 - B5 : Mouvement répété

Fichier modèle : LP_N1_B.xml

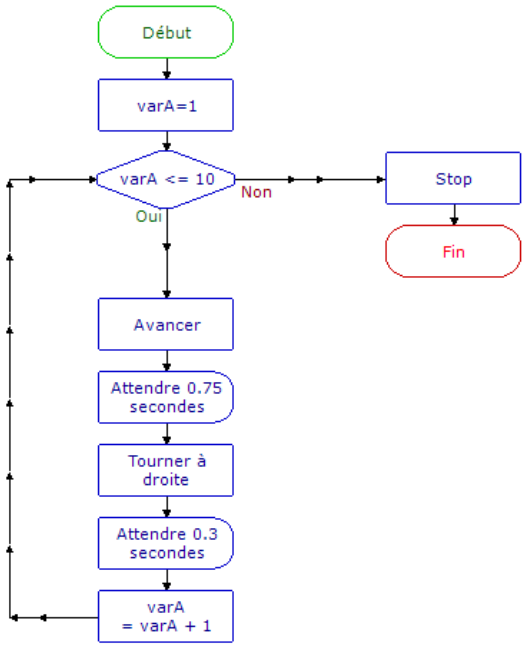
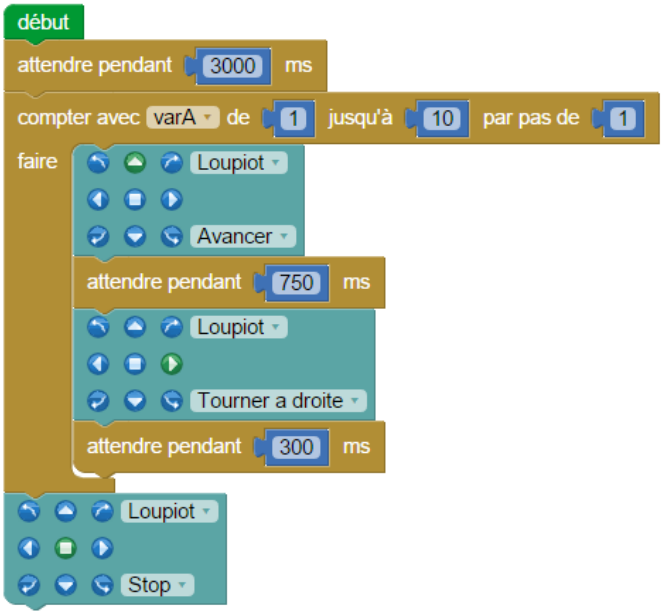
Objectif : répéter 10 fois l'action suivante : avancer puis tourner à droite afin d'effectuer un polygone.

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) :

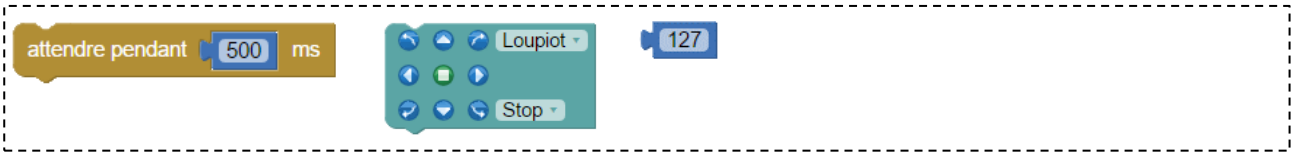
Exercice niveau 1 - B6 : Accélération brutale

Fichier modèle : LP_N1_B.xml

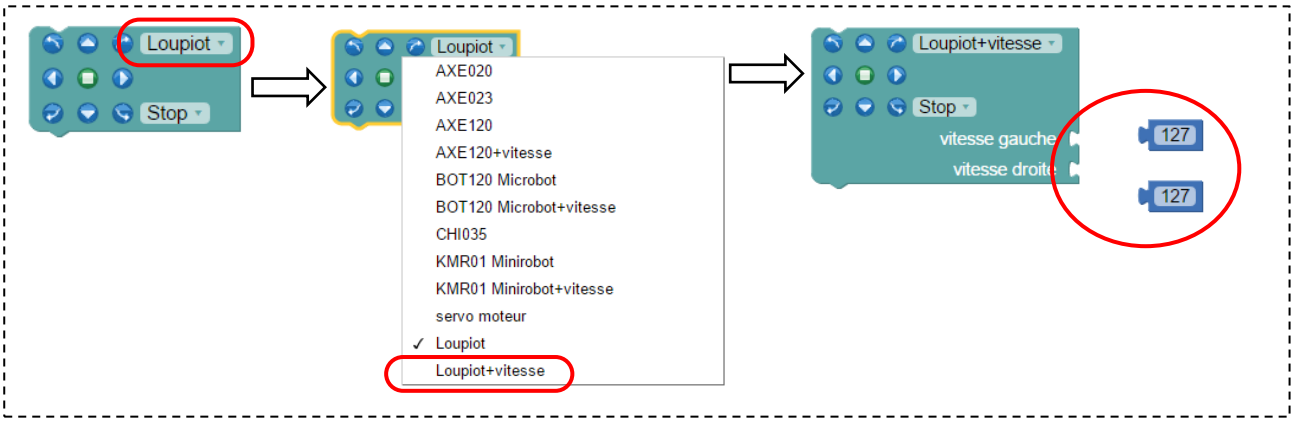
Objectif : Au bout de 3 secondes, faire avancer le robot à 50% de sa vitesse pendant 3 secondes, puis à 100% pendant 3 secondes avant de l'arrêter.

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

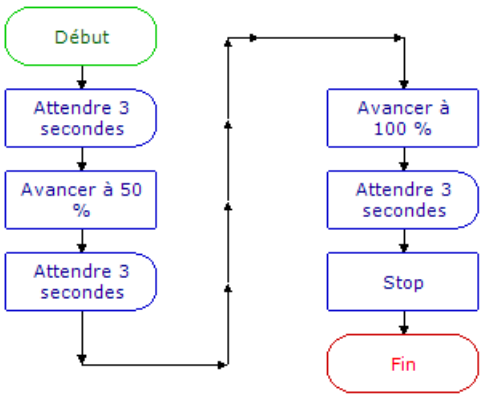
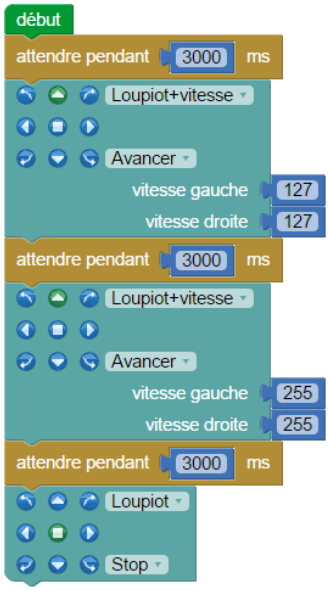
Instruction(s) utilisée(s) :



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 : Transposer l'état d'une entrée à une sortie

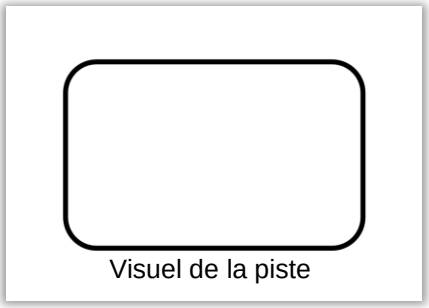
Fichier modèle : LP_N1_C.xml

Objectif : transposer l'état du capteur de ligne centre sur le témoin alerte.

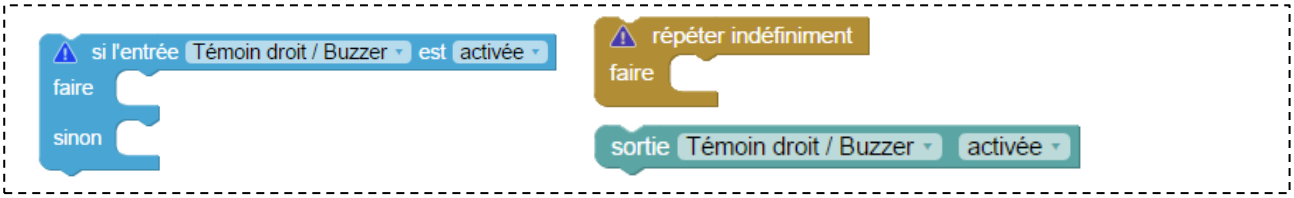
Pour vérifier la correspondance entre l'état du capteur et le témoin lumineux, déplacer le robot sur la ligne.

Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.

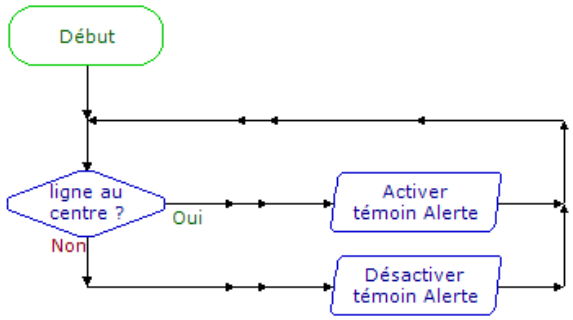
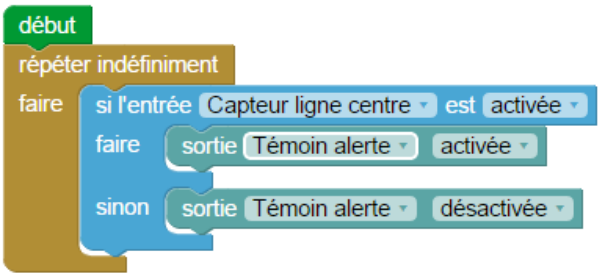
Notion(s) abordée(s) : structure conditionnelle / Lecture de l'état d'une entrée du robot.


Visuel de la piste

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 : Transposer l'état de plusieurs entrées

Fichier modèle : LP_N1_C.xml

Objectif : transposer l'état des capteurs de ligne droit, centre et gauche respectivement sur les témoins lumineux droit, alerte et gauche.

Pour vérifier la correspondance entre l'état du capteur et le témoin lumineux, déplacer le robot sur la ligne.

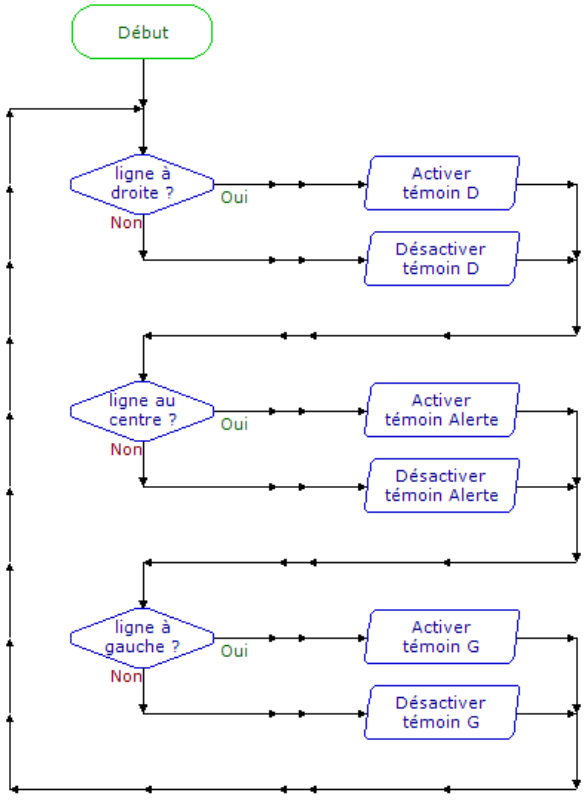
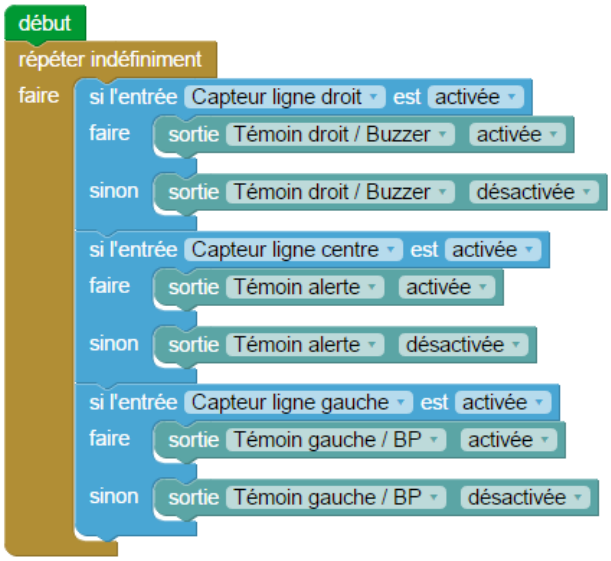
Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.

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) :

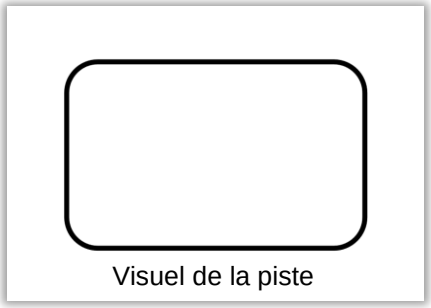
Exercice niveau 1 - C3 : Avancer jusqu'à une ligne (un capteur)

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.

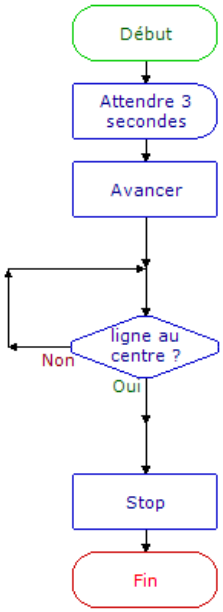
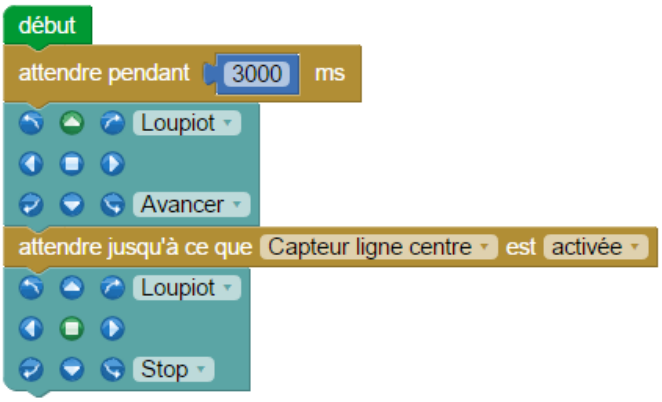
Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.

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


Visuel de la piste

Instruction(s) utilisée(s) :

Correction :

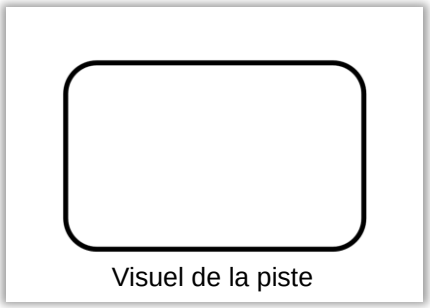
Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C3_Organigramme.plf	Fichier Blockly : LP_N1_C3.xml

Remarque(s) :

Exercice niveau 1 - C4 : Avancer jusqu'à une ligne (3 capteurs)

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.

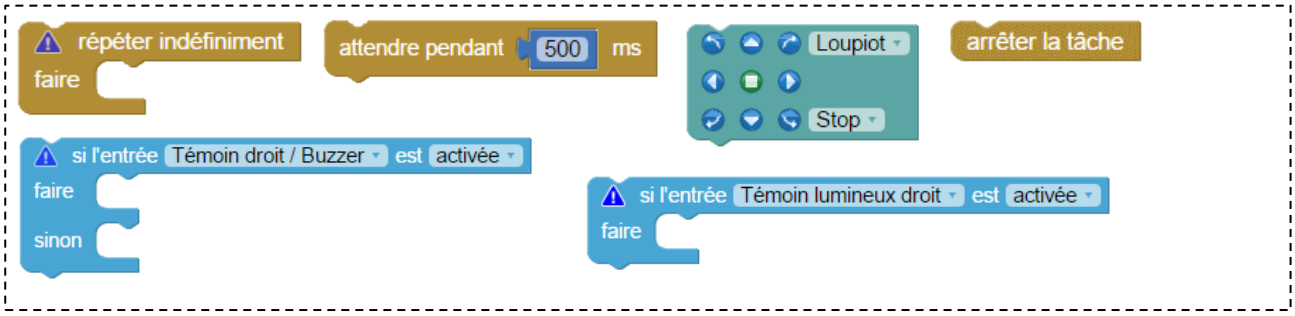

Visuel de la piste

Attention en fonction de l'inclinaison du Loupiot, il peut ne pas s'arrêter.

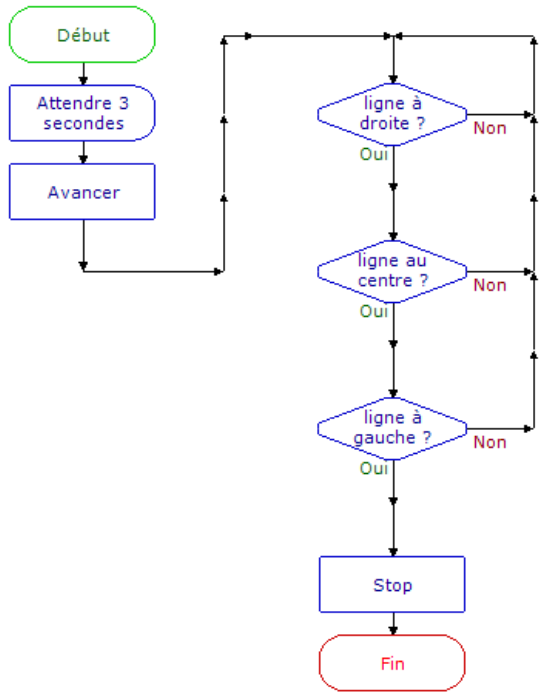
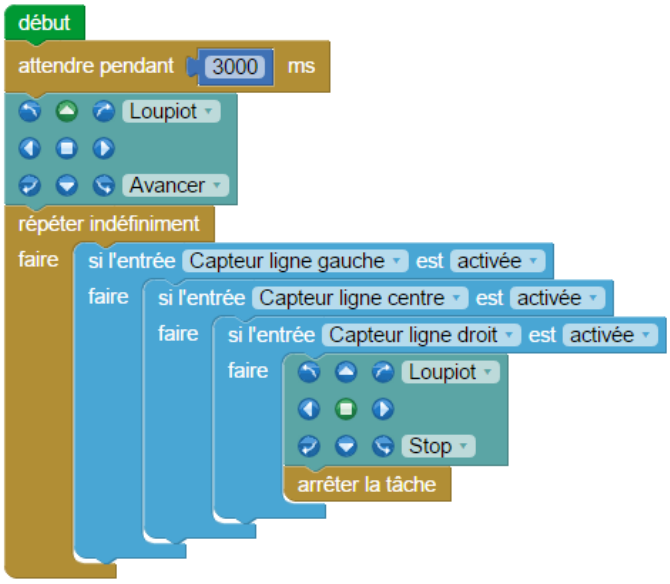
Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.

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

Instruction(s) utilisée(s) :



Correction :

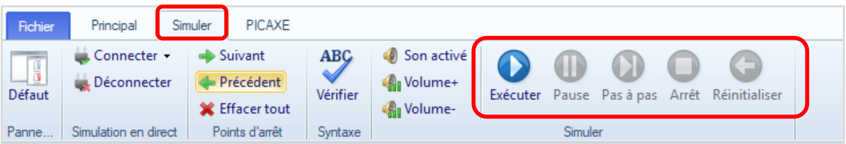
Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C4_Organigramme.plf	Fichier Blockly : LP_N1_C4.xml

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.

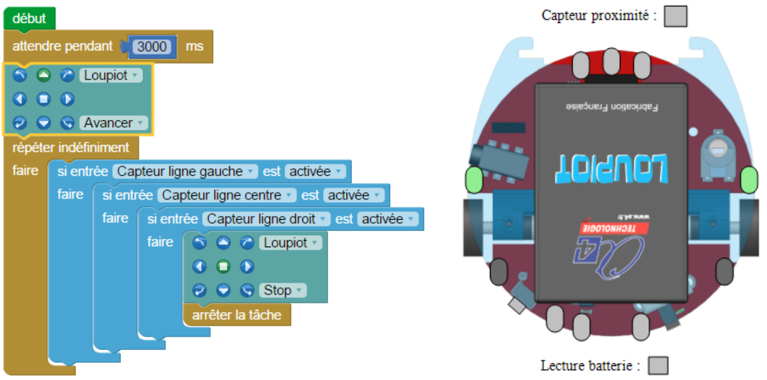
Exemple d'utilisation de la simulation

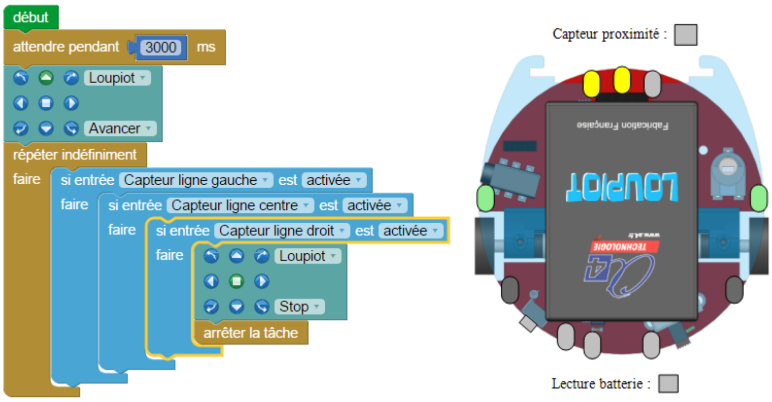
Pour lancer et contrôler une simulation, utiliser les boutons **Exécuter / Pause / Pas à pas / Arrêt** à partir du menu **Simuler**.

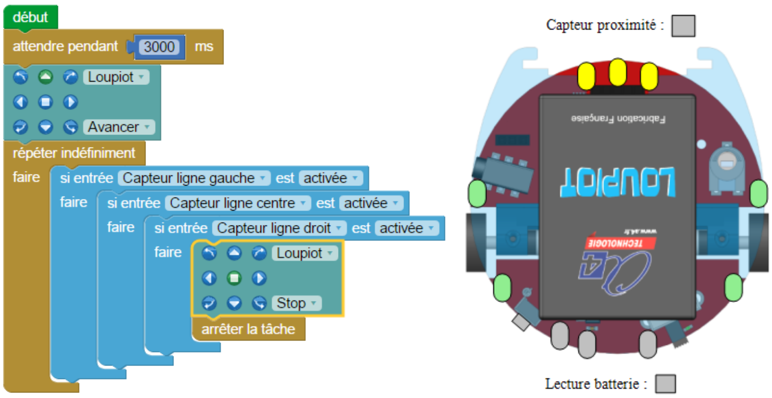


Ouvrir le programme « LP_N1_C4.xml » réalisé précédemment et lancer la simulation.
La première instruction va activer les pastilles avant des deux moteurs pour simuler la marche avant = on voit qu'elles passent au vert.
Pour valider la triple condition, il faut cliquer sur les pastilles des capteurs de ligne à l'avant du robot pour simuler leur activation.
= elles passent au jaune pour montrer qu'elles ont été activées par un élément extérieur au programme contrairement à la couleur verte.
Quand les 3 sont activées, on voit que le programme se termine et que les sorties des moteurs sont toutes activées pour montrer que les moteurs sont arrêtés. Voir ci-dessous quelques étapes de la simulation du programme :

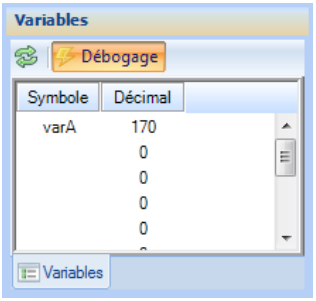
Note : La simulation surligne les blocs dans l'espace de travail pour vous montrer ou en est le programme.







Exercice niveau 1 - C5 : Lecture batterie / debug



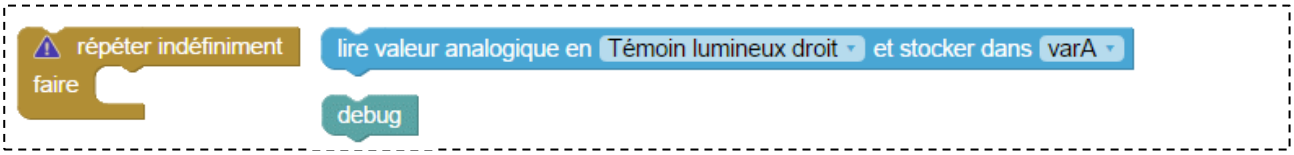
Fenêtre Variables

Fichier modèle : LP_N1_C.xml

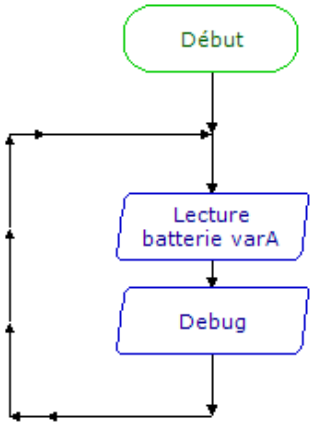
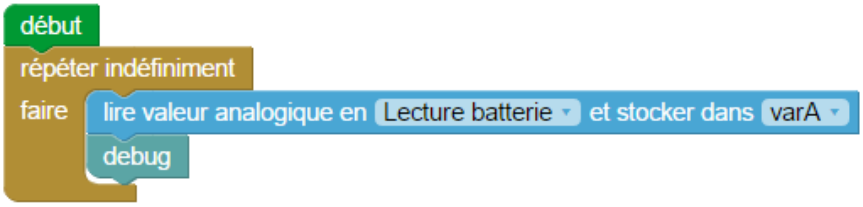
Objectif : lire le niveau de tension de la batterie du Loupiot et l'afficher dans la fenêtre Variables du logiciel.

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 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 PICAXE Editor.
- 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.

Exercice niveau 1 - C6 : Appuyer 4 fois sur le BP pour jouer une musique

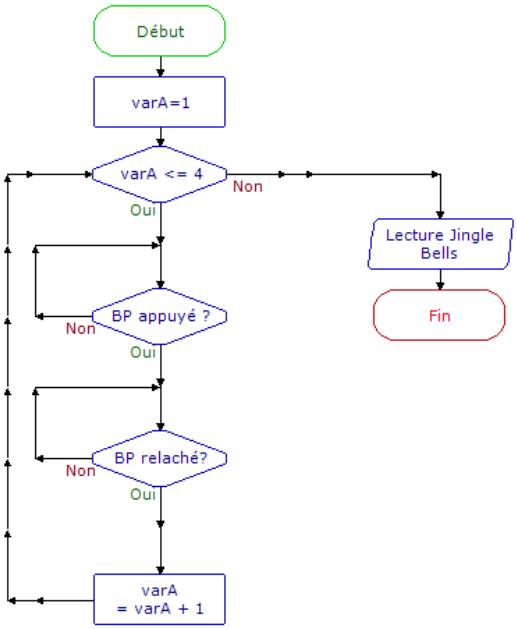
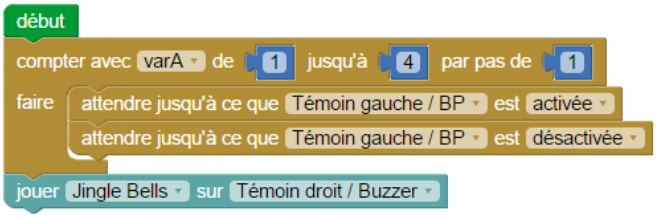
Fichier modèle : LP_N1_C.xml

Objectif : Attendre que le bouton-poussoir ait été appuyé 4 fois pour lancer une musique.

Notion(s) abordée(s) :

Instruction(s) utilisée(s) :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : LP_N1_C6_Organigramme.plf	Fichier Blockly : LP_N1_C6.xml

Remarque(s) :

- On place ici deux attentes à la suite car l'appui d'un bouton est caractérisé par deux étapes : l'appui et le relâchement du bouton.

Programmation niveau 2 (version de base)

Niveau 2 : approfondissement des principes de programmation abordés dans le niveau 1 en concevant des programmes plus élaborés qui répondent à des cas concrets d'utilisation du robot.

Liste des programmes du niveau 2

Nom du fichier	Description	Objectif
Niveau 2 A **Fichier modèle** : LP_N2_A.xml		
LP_N2_A1	Chenillard	Fonctionnalité matérielle abordée : - Utilisation concrète des témoins lumineux - Utilisation concrète du buzzer Notions de programmation abordées : - Utilisation approfondie des variables
LP_N2_A2	Clignotement en fonction de la position d'une ligne	
LP_N2_A3	Accélération / Décélération du clignotement d'une LED	
LP_N2_A4	Jouer la musique de Star Wars	
Niveau 2 B **Fichier modèle** : LP_N2_B.xml		
LP_N2_B1	Suivi d'une ligne fine	Fonctionnalité matérielle abordée : - Utilisation concrète de la gestion des moteurs Notions de programmation abordées : - Procédures
LP_N2_B2	Suivi d'une ligne large	
LP_N2_B3	Accélération / décélération	
LP_N2_B4	Accélération / décélération avec procédure	
Niveau 2 C **Fichier modèle** : LP_N2_C.xml		
LP_N2_C1	Détecter 3 fois un code	Fonctionnalité matérielle abordée : - Utilisation concrète des capteurs du robot Notions de programmation abordées : - Opérations booléennes
LP_N2_C2	Aller – retour sur une ligne	
LP_N2_C3	Prison	
LP_N2_C4	Clavier musical codé	

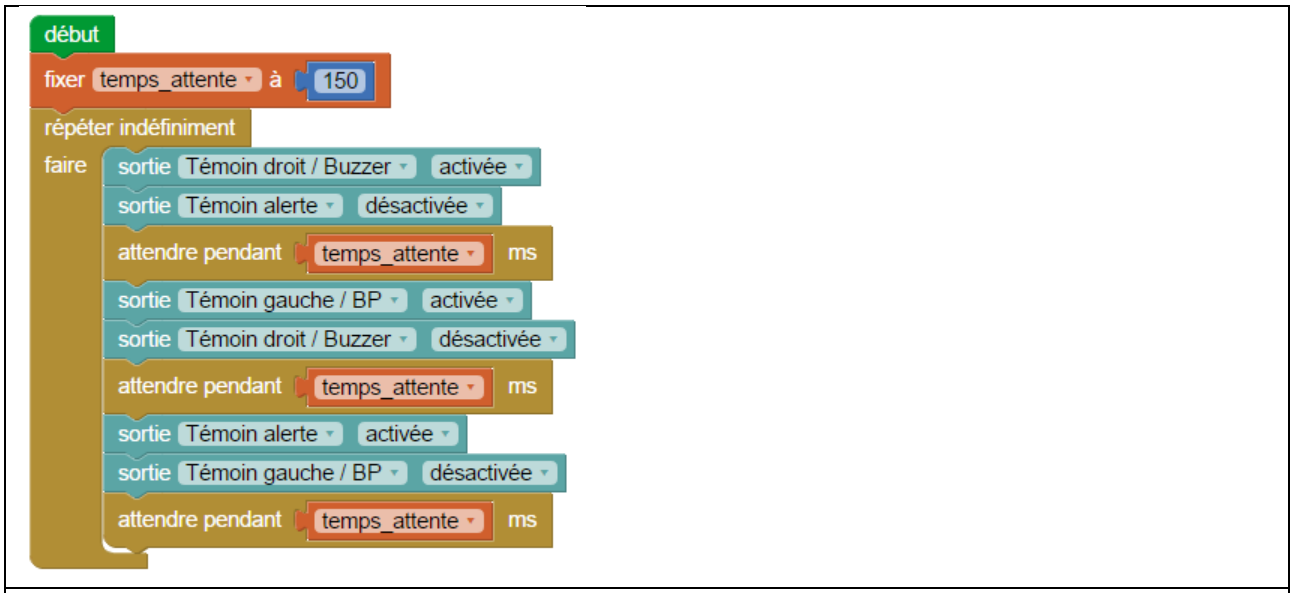
Exercice niveau 2 - A1 : Chenillard

Fichier modèle : LP_N2_A.xml

Objectif : Faire clignoter les 3 témoins lumineux les uns à la suite des autres.

Instruction(s) utilisée(s) :

Correction :



Fichier Blockly : LP_N2_A1.xml

Remarque(s) :

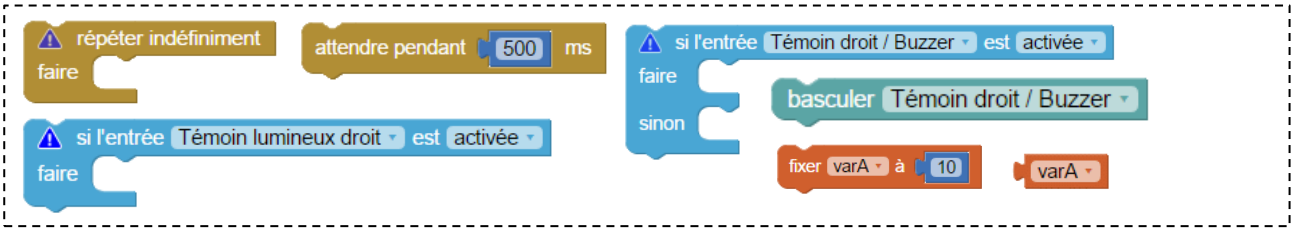
- On utilise ici une variable pour fixer le même temps à toutes les attentes. Dans ce cas, il n'y a qu'un seul champ de texte à modifier et dans le cas contraire 3.
- Modifier la valeur de la variable du temps d'attente pour voir comment le programme réagit.
- Pour modifier le nom d'une variable ou en créer une nouvelle, veuillez lire la documentation du logiciel.

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

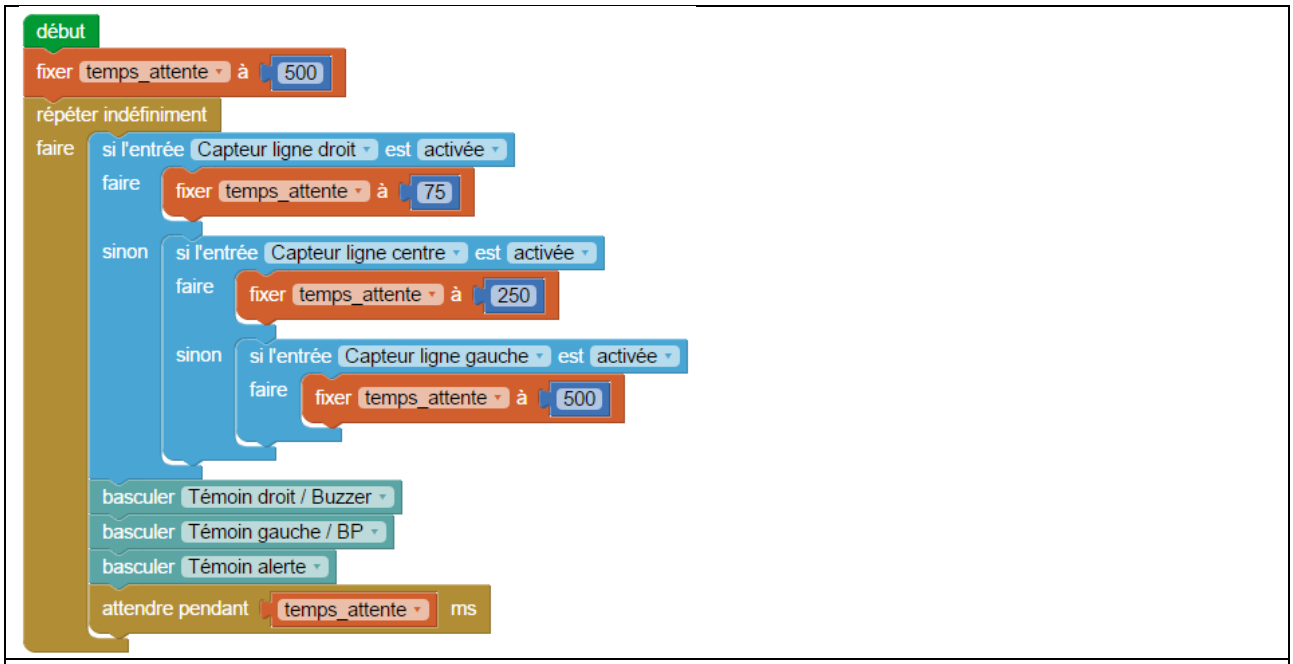
Fichier modèle : LP_N2_A.xml

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) :



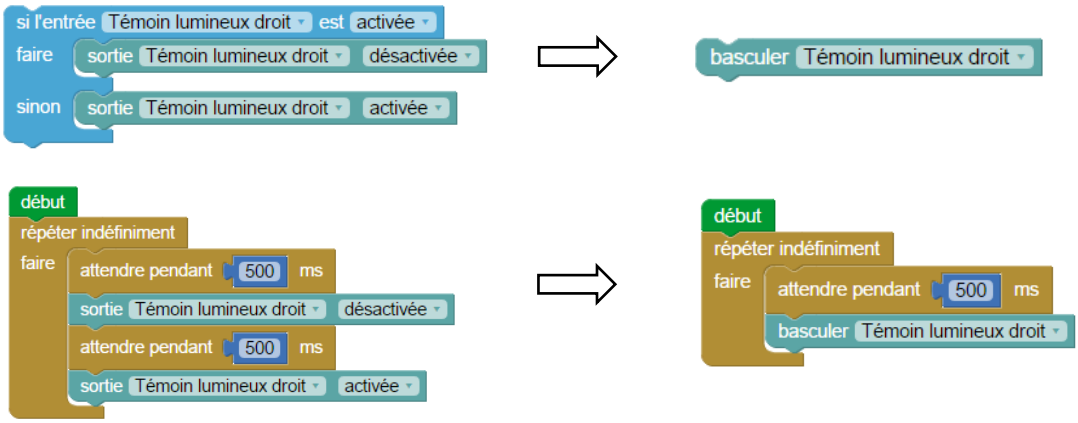
Correction :



Fichier Blockly : LP_N2_A2.xml

Remarque(s) :

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

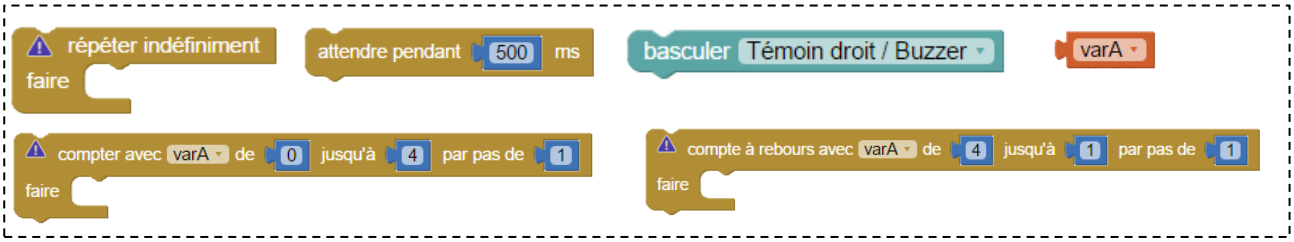


Exercice niveau 2 - A3 : Accélération / décélération du clignotement d'un témoin lumineux

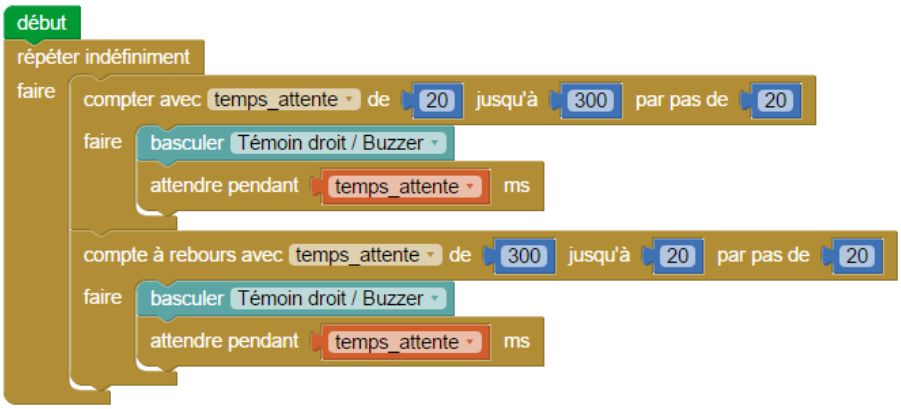
Fichier modèle : LP_N2_A.xml

Objectif : Accélérer puis décélérer le clignotement d'un témoin lumineux en fonction du temps d'attente (plus il est important, plus le clignotement accélère).

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_A3.xml

Remarque :

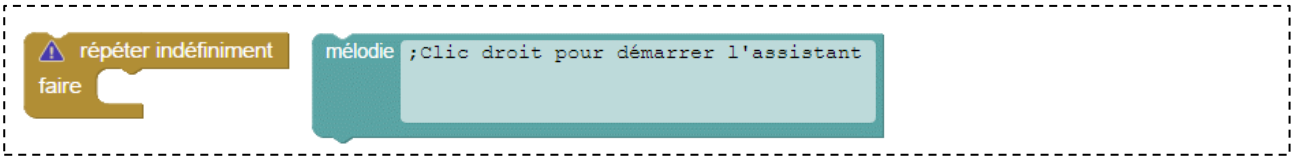
- La variable utilisée dans la boucle « **compter** … » prendra différentes valeurs correspondant au compte à rebours (300, 280, …. 40, 20, 0).

Exercice niveau 2 - A4 : Jouer la musique de Star Wars

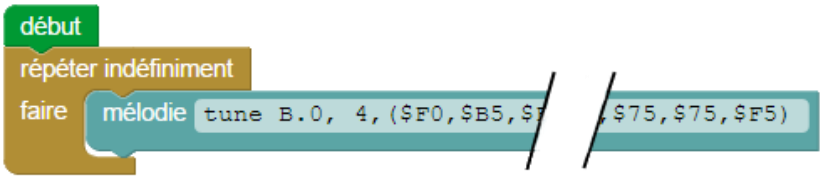
Fichier modèle : LP_N2_A.xml

Objectif : Jouer la musique de Star Wars à partir du bloc « mélodie ».

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_A4.xml

Remarque :

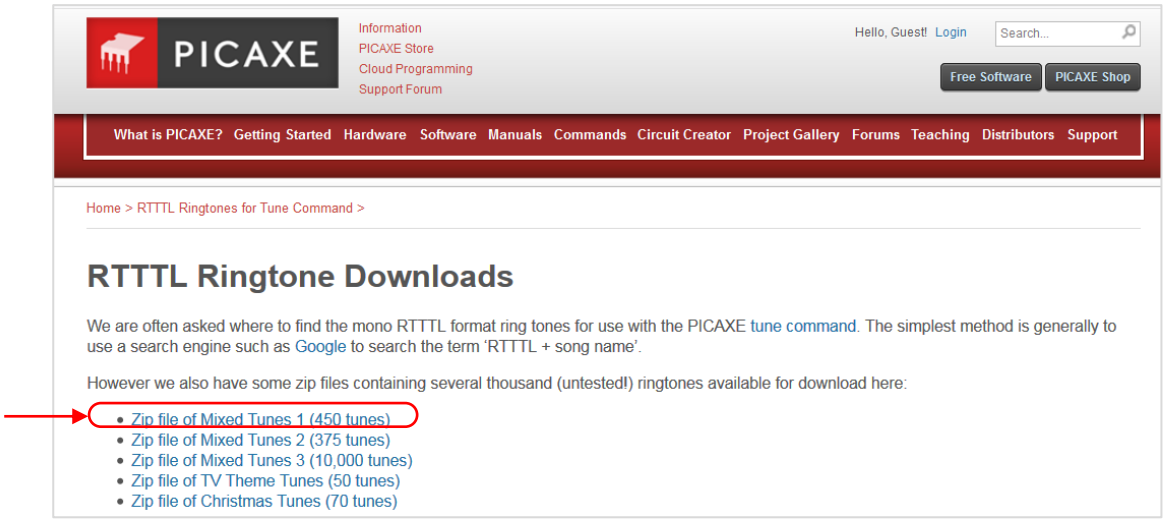
- Vous pouvez créer vous-même votre musique à partir de l'assistant ou tester des musiques téléchargeables sur www.picaxe.com
- Plus d'infos sur la commande « tune » sur : http://www.picaxe.com/BASIC-Commands/Digital-InputOutput/tune

Utiliser l'assistant Ring Tone Tune Wizard pour créer une mélodie

Le fichier RTTTL.txt d'une musique est indispensable pour créer le texte à insérer dans le bloc mélodie. Vous pouvez créer ce fichier à partir de l'assistant ou bien vous rendre sur le site Picaxe pour télécharger les zips de musiques précompilés en RTTTL.txt. Nous allons voir ici la deuxième méthode.

Etape 1 :

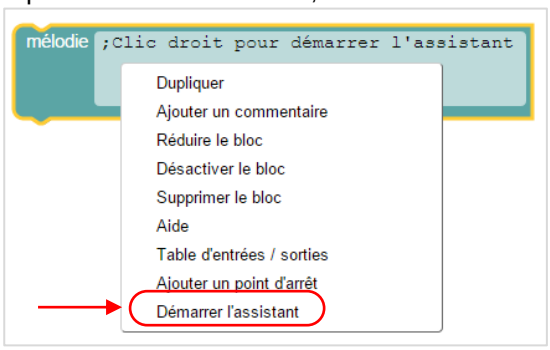
Télécharger le fichier zip contenant les musiques en format RTTTL à l'adresse suivante : www.picaxe.com/RTTTL-Ringtones-for-Tune-Command



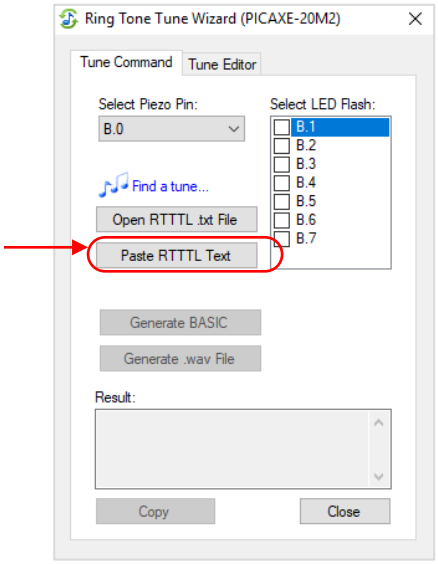
Cliquer sur : Zip file of Mixed Tunes 1 (450 tunes) pour télécharger et décompresser le fichier sur votre ordinateur.

Etape 2 :

A partir de Picaxe Editor 6, faire un clic droit sur le bloc « mélodie » puis cliquer sur **Démarrer l'assistant**.



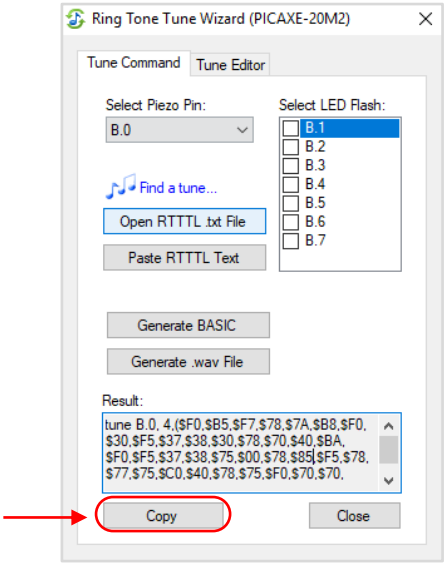
A partir de la fenêtre **Ring Tone Tune Wizard**, cliquer sur **Open RTTTL.txt file**

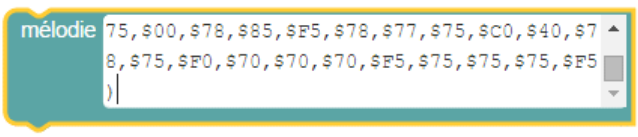


Ouvrir le fichier **starwar.txt** contenu dans le zip précédemment téléchargé.
Remarque : La broche B.0 correspondant au buzzer est sélectionnée par défaut.

Etape 3 :

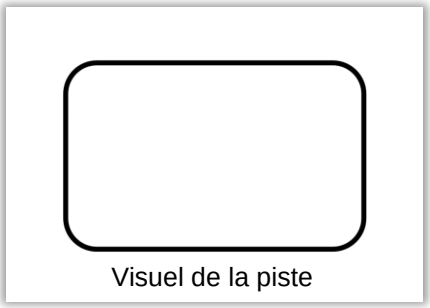
Copier le texte généré après l'ouverture du fichier en cliquant sur **Copy** et le coller dans le bloc « mélodie » avec les touches **Ctrl + V**.





Note : Si vous souhaitez composer votre mélodie à partir de l'assistant. A partir de l'onglet **Tune Editor**, composez votre mélodie. Revenez sur l'onglet **Tune Command** puis cliquez sur **Generate BASIC**. Copiez le texte de la zone **Result** et collez dans le bloc **Mélodie**.

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

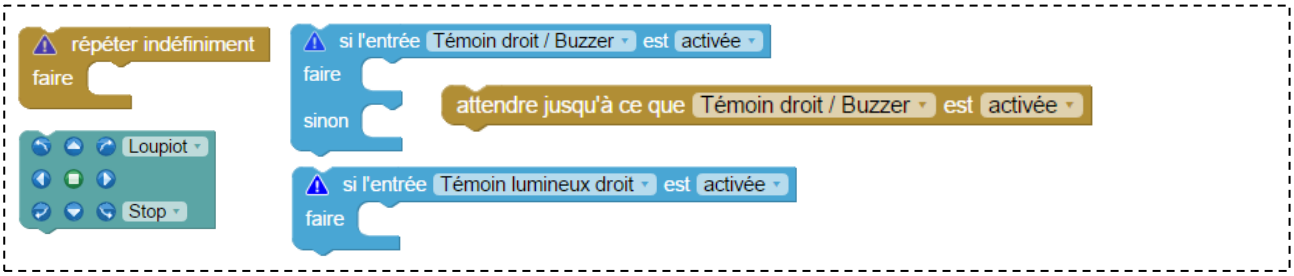

Visuel de la piste

Fichier modèle : LP_N2_B.xml

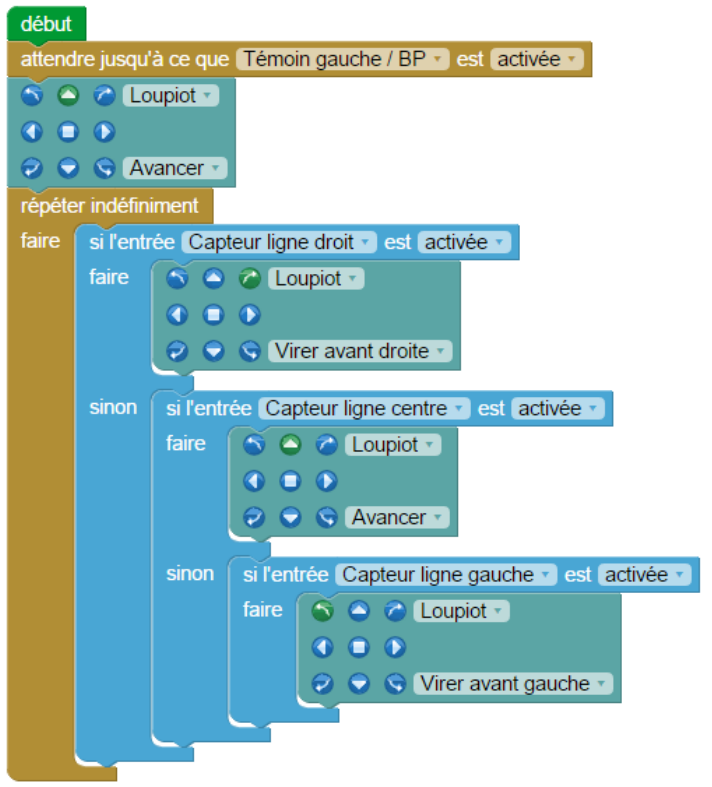
Objectif : Suivre une ligne faisant la largeur d'un capteur de ligne à l'appui du bouton-poussoir.

Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.

Instruction(s) utilisée(s) :



Correction :

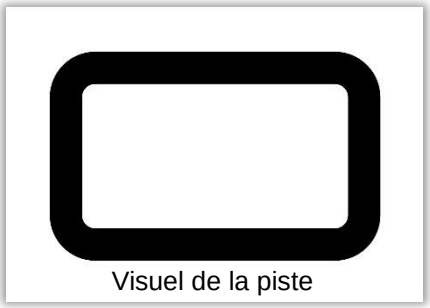


Fichier Blockly : LP_N2_B1.xml

Remarque(s) :

- Pensez à bien régler vos capteurs de ligne lors du test de ces programmes (voir chapitre Mise en service du Loupiot).
- Pour tous les programmes utilisant les moteurs du robot, nous placerons une attente pour lancer le programme (quand le bouton-poussoir est appuyé). 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

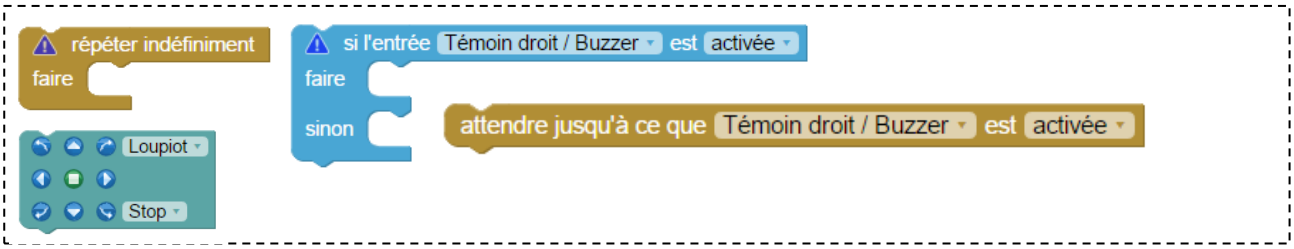

Visuel de la piste

Fichier modèle : LP_N2_B.xml

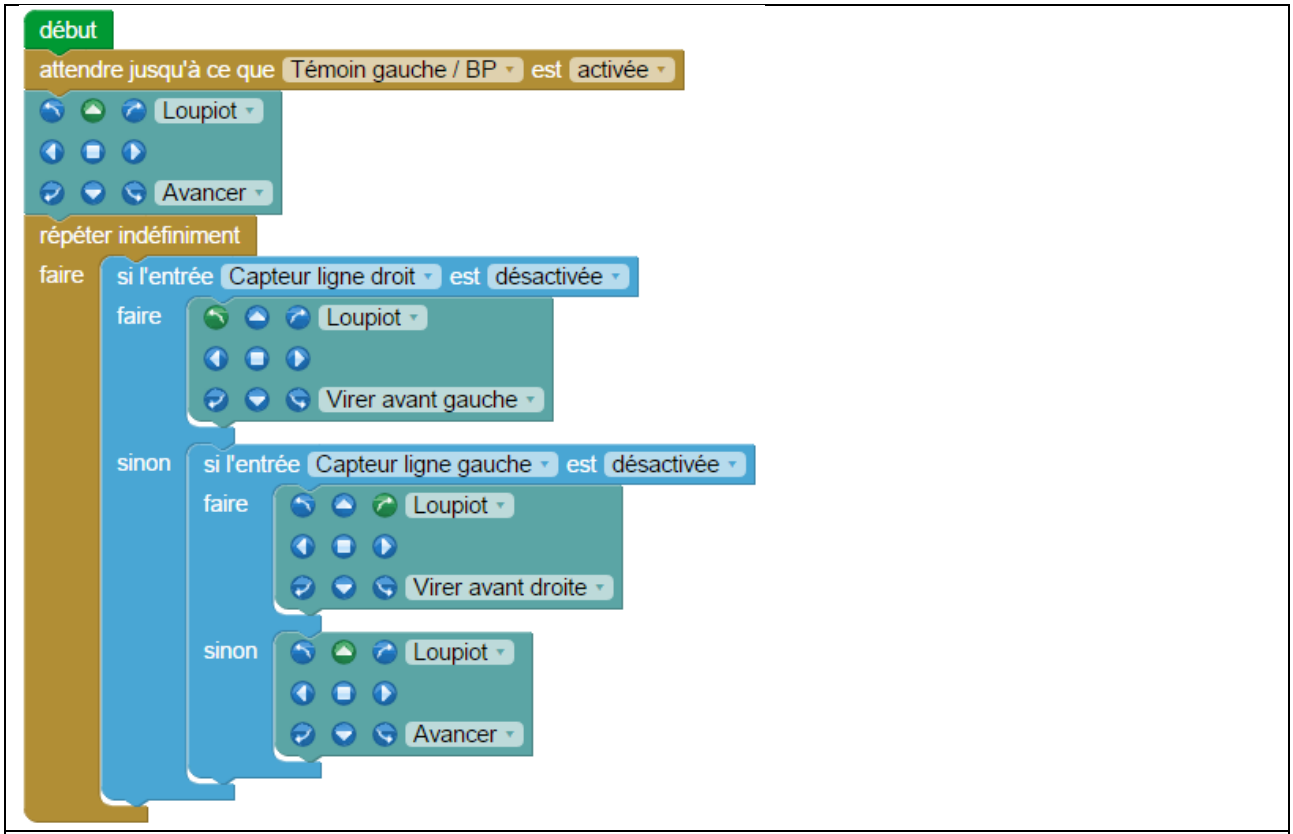
Objectif : Suivre une ligne faisant la largeur des 3 capteurs de ligne à l'appui du bouton-poussoir.

Note : Imprimer la piste "Piste_suivi_ligne_type2" pour tester ce code.

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_B2.xml

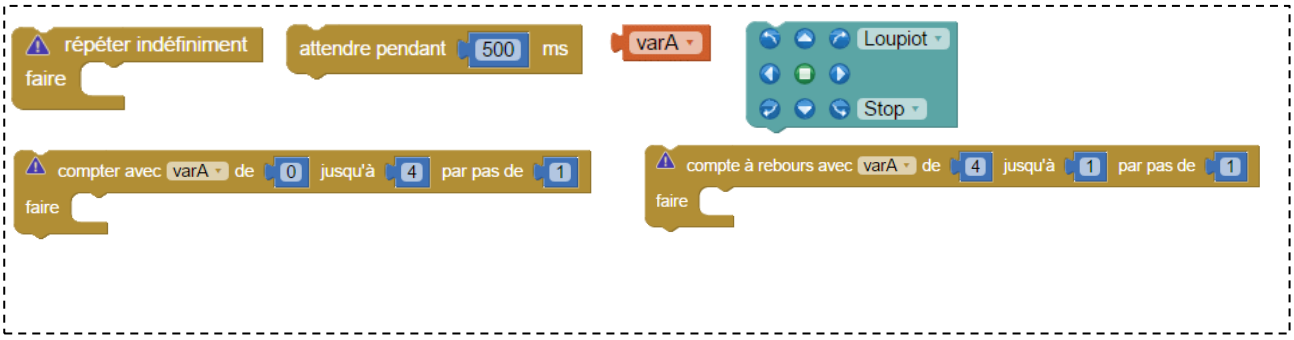
Remarque(s) :

Exercice niveau 2 - B3 : Accélération / décélération

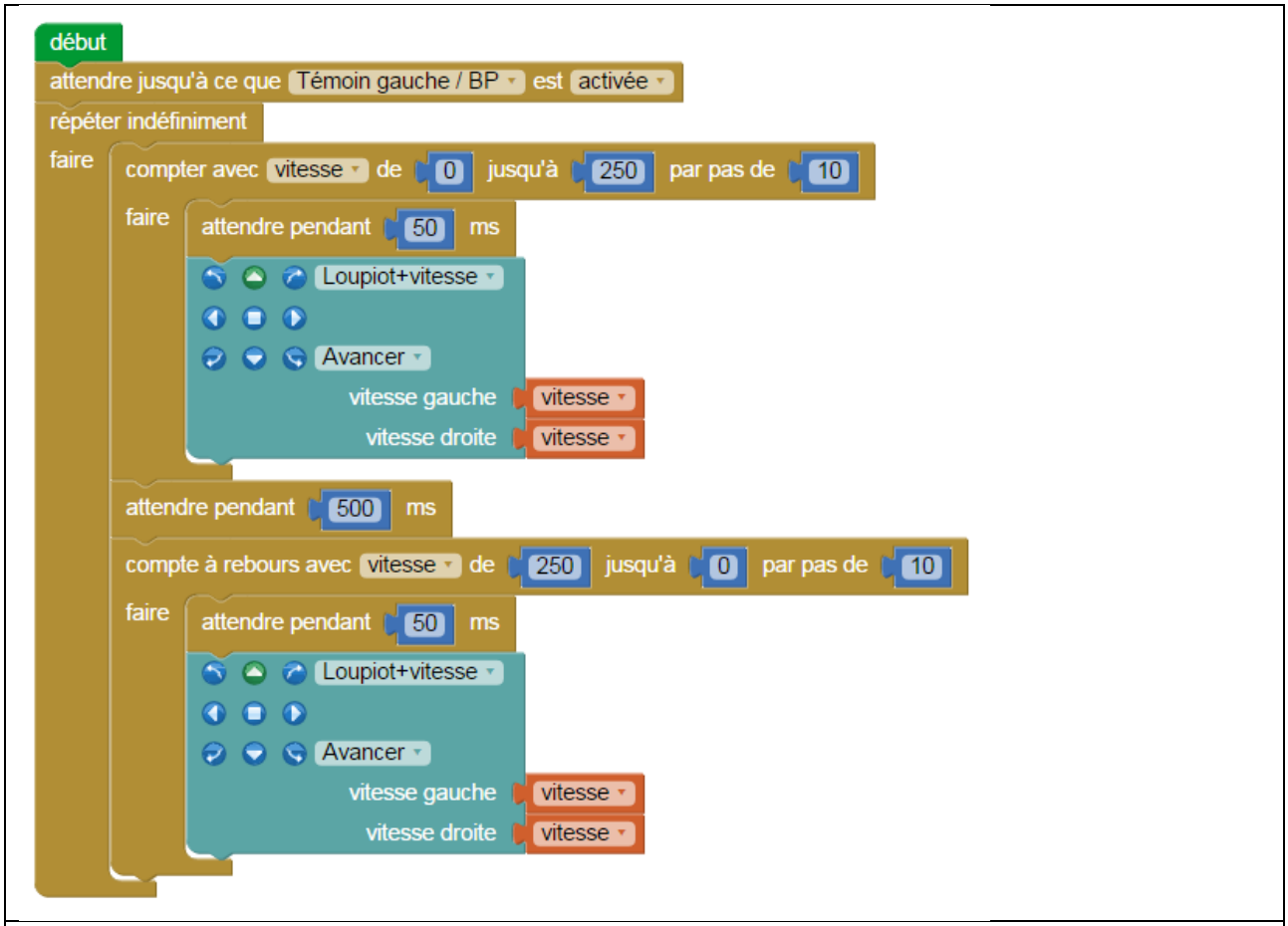
Fichier modèle : LP_N2_B.xml

Objectif : Faire accélérer puis décélérer le robot tout en avançant à l'appui du bouton-poussoir.

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_B3.xml

Remarque(s) :

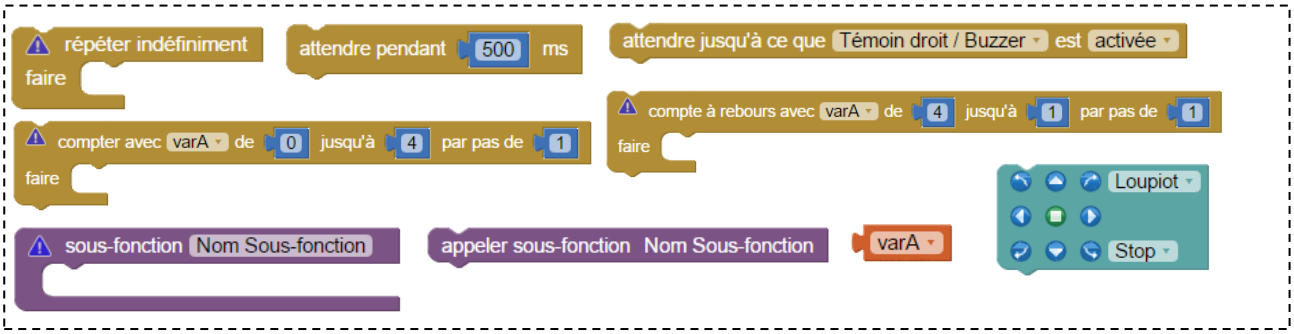
- Une boucle « répété n fois » allant de 0 à 250 par pas de 10 est exécutée 26 fois (0, 10, …, 240, 250).
- Une accélération ou une décélération dure 1,3 secondes = 26 x 50 ms (le temps d'attente placée dans la boucle « répété n fois »).

Exercice niveau 2 - B4 : Accélération / décélération avec procédure

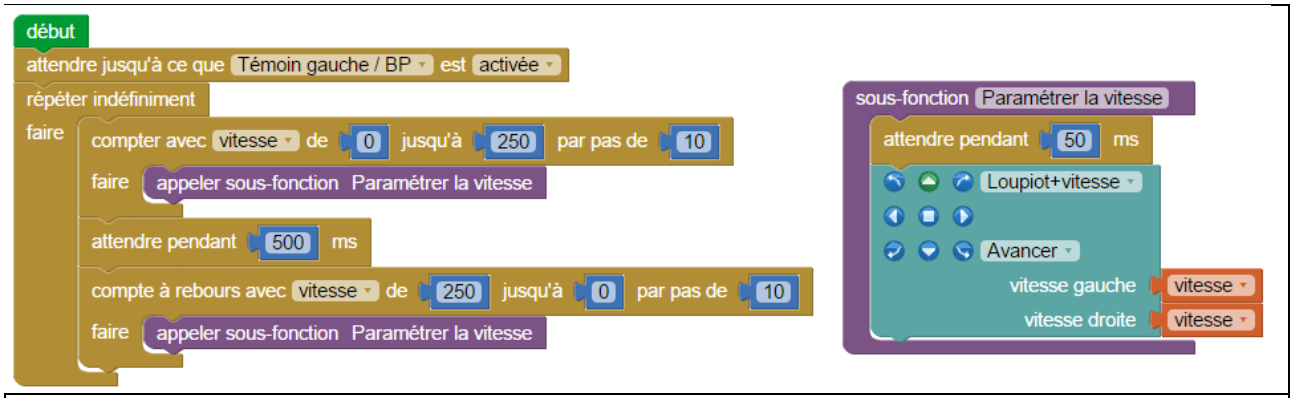
Fichier modèle : LP_N2_B.xml

Objectif : A l'appui du bouton-poussoir, faire accélérer puis décélérer le robot tout en avançant en utilisant des procédures pour rendre le programme plus lisible.

Instruction(s) utilisée(s) :



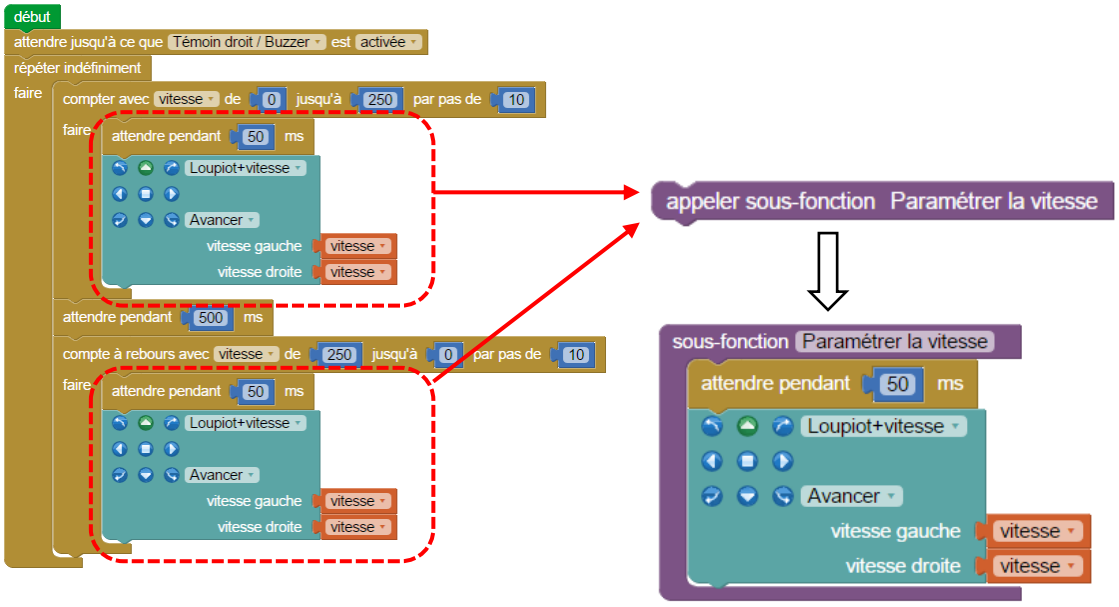
Correction :



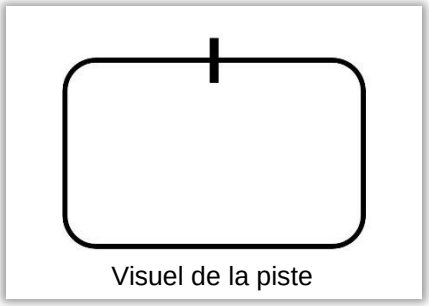
Fichier Blockly : LP_N2_B4.xml

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

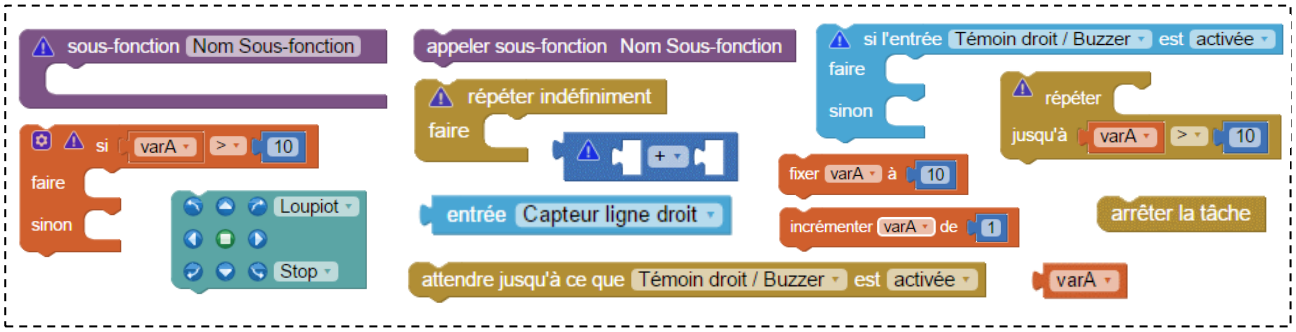

Visuel de la piste

Fichier modèle : LP_N2_C.xml

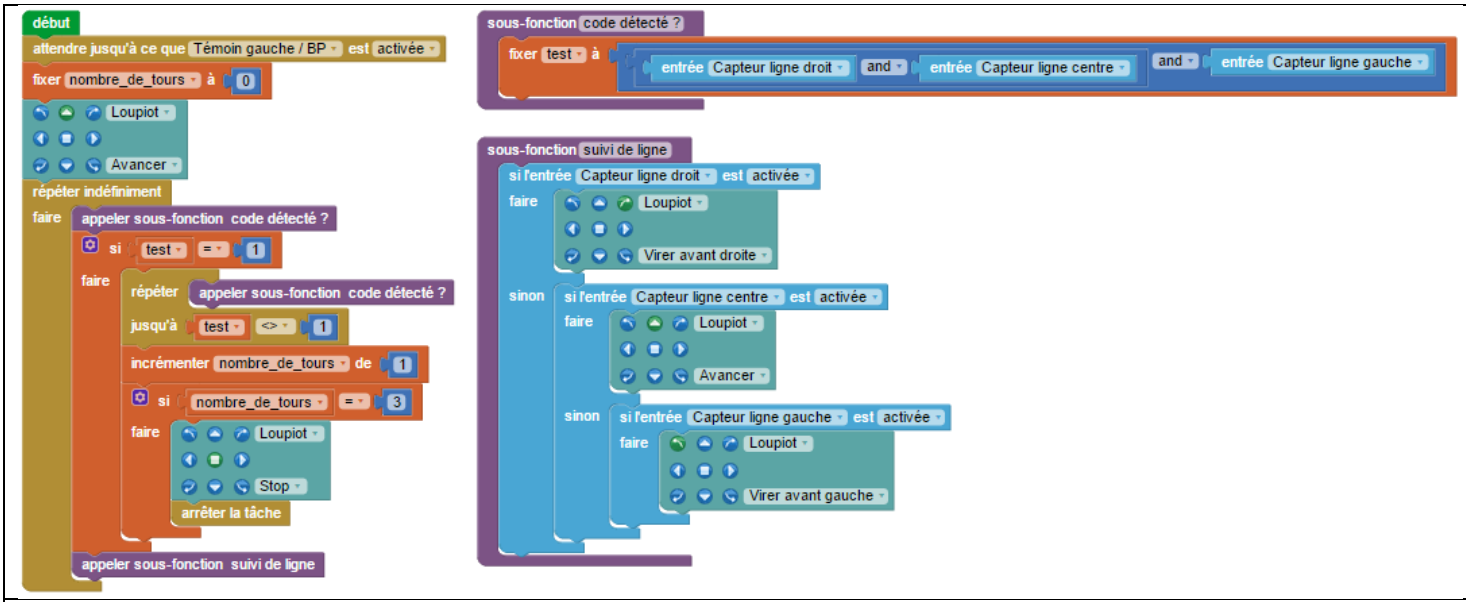
Objectif : A l'appui du bouton-poussoir, faire suivre une ligne fine au robot. Un code (3 capteurs de ligne activés) est placé sur la ligne et doit être détecté trois fois par le robot pour le stopper.

Note : Imprimer la piste "Piste_suivi_ligne_avec_code" pour tester ce code.

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_C1.xml

Remarque(s) :

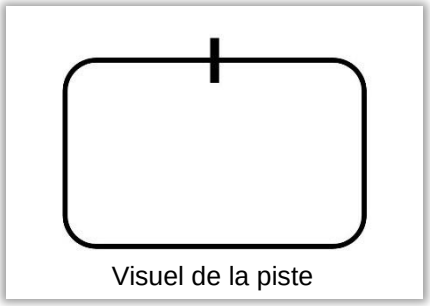
- Pour tous les programmes utilisant les moteurs du robot, nous placerons une attente pour lancer le programme quand le bouton-poussoir est appuyé. Cela évite que le robot démarre directement après le transfert du programme et arrache le câble de programmation.
- La procédure « code détecté ? » comporte une opération booléenne : elle renvoie dans la variable « test » la valeur « 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

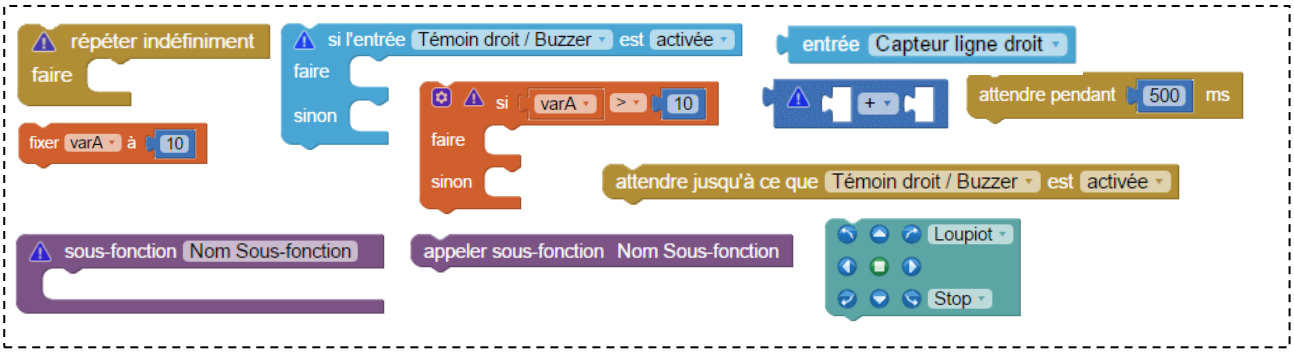
Objectif : A l'appui du bouton-poussoir, faire suivre une ligne fine au robot.
Lorsqu'un code (3 capteurs de ligne activés) est détecté, le robot fait demi-tour et repart sur la ligne en sens inverse.

Note : Imprimer la piste "Piste_suivi_ligne_avec_code" pour tester ce code.

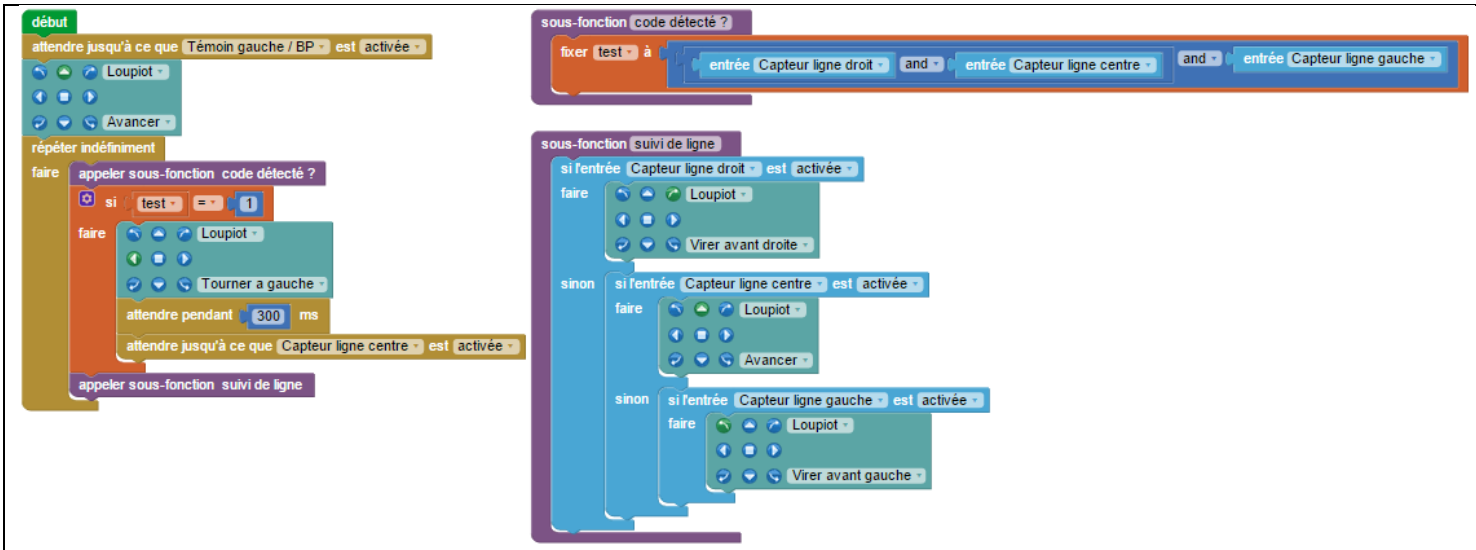


Visuel de la piste

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

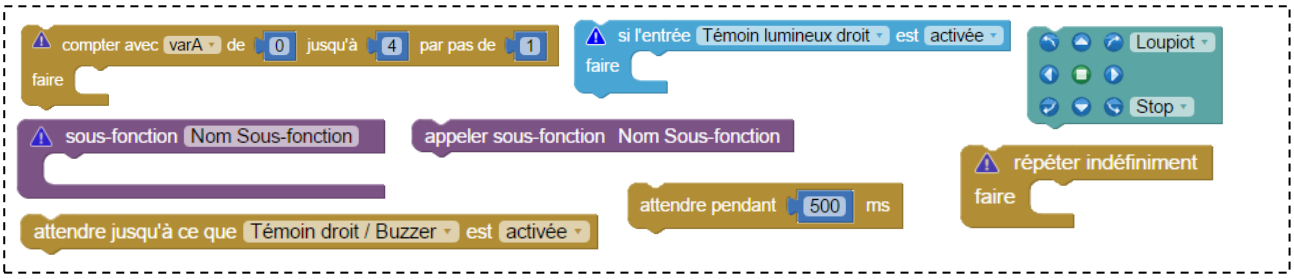
Fichier modèle : LP_N2_C.xml

Objectif : à l'appui du bouton-poussoir, empêcher le robot de sortir d'un périmètre délimité par une ligne noire.

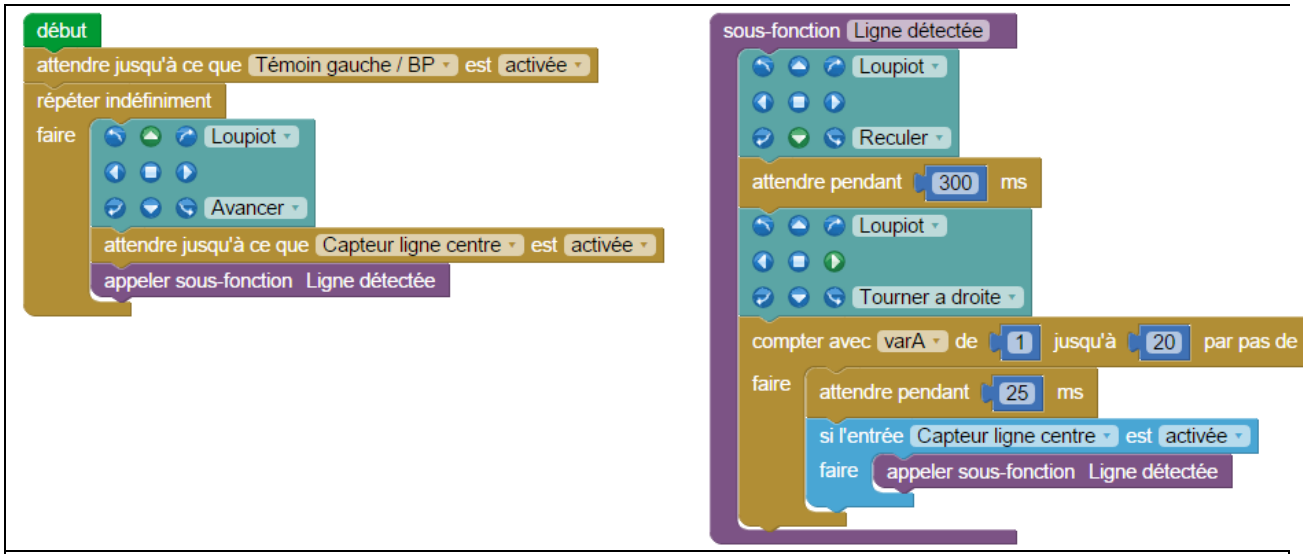
Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.

Visuel de la piste

Instruction(s) utilisée(s) :



Correction :



Fichier Blockly : LP_N2_C3.xml

Remarque(s) :

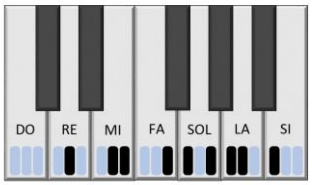
- Lorsqu'une ligne noire est détectée par le capteur de ligne centre, la procédure « Ligne détectée » est déclenchée : le robot va reculer pour s'éloigner de la ligne pendant 0,3 seconde. Il va ensuite changer de trajectoire pour éviter de retoucher la ligne en tournant à droite.
- Pendant que le robot 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.

Exercice niveau 2 - C4 : Clavier musical codé

Fichier modèle : LP_N2_C.xml

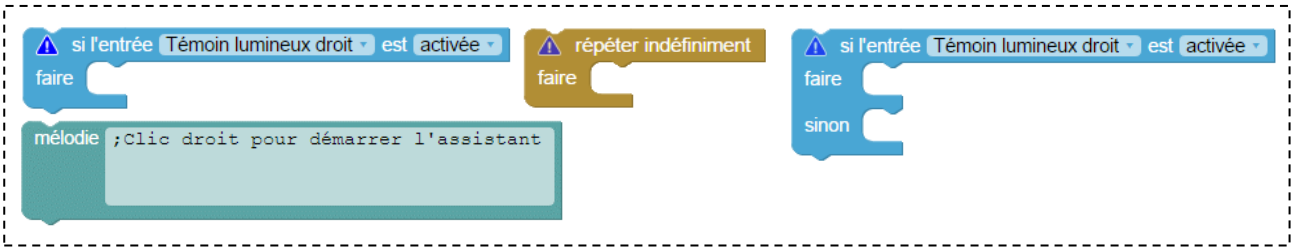
Objectif : Scanner des codes pour jouer des notes de musique.

Note : imprimer la feuille « piano_loupiot » puis positionner le Loupiot sur les différents codes pour jouer la note correspondante.

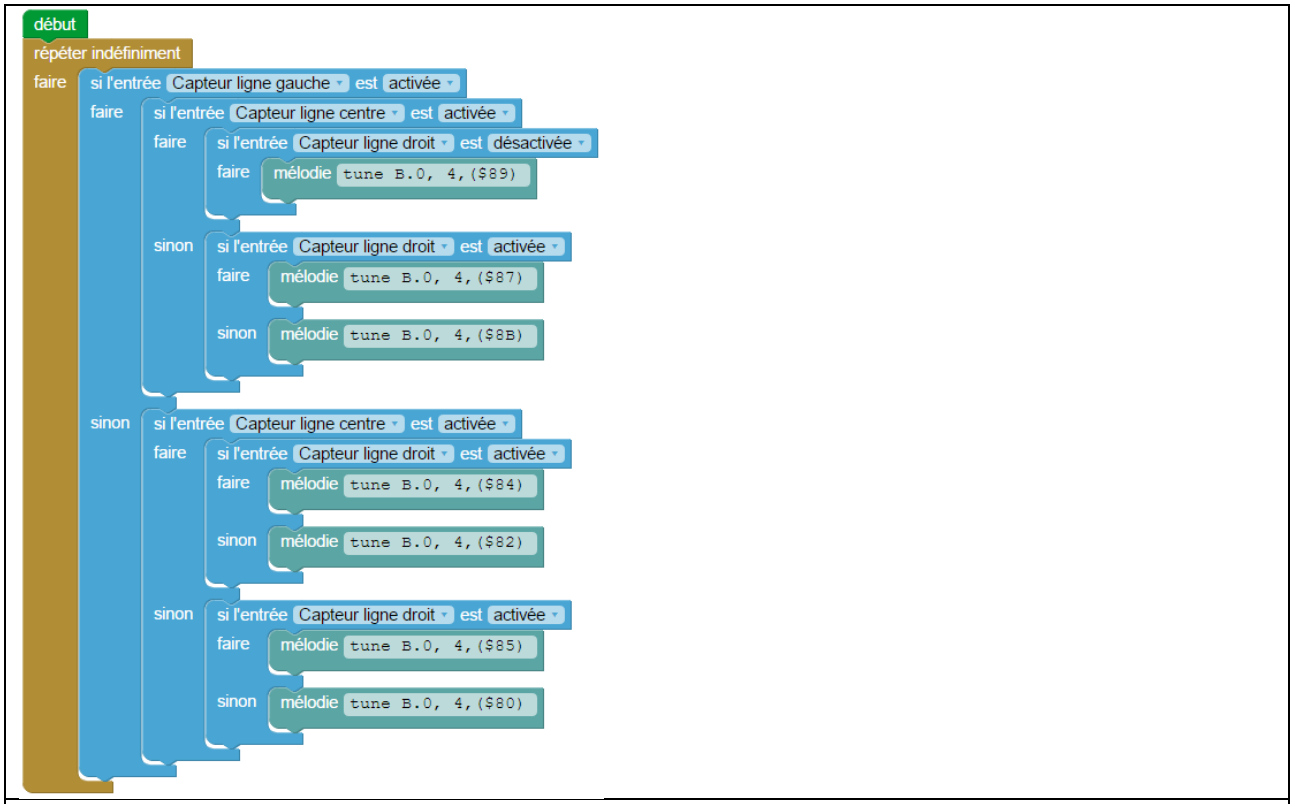


Visuel de la feuille

Instruction(s) utilisée(s) :



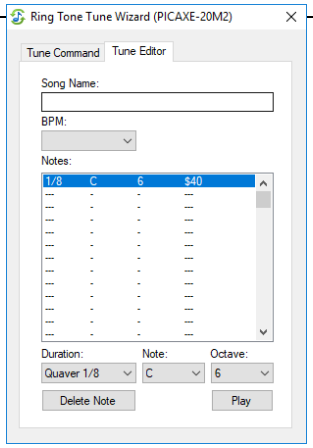
Correction :



Fichier Blockly : LP_N2_C4.xml

Remarque(s) :

- Pour générer les notes, on utilise l'assistant Ring Tone Tune wizard (ci-contre) vu dans l'exercice Niveau 2 – A4 Jouer la musique Star Wars.



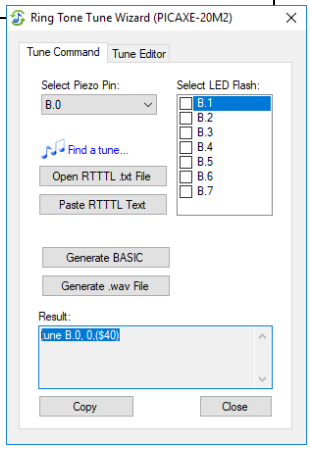


Tableau de correspondance des notes Françaises et Anglaises						
DO	RE	MI	FA	SOL	LA	SI
C	D	E	F	G	A	B

Programmation niveau 3 (version de base + options)

Niveau 3 : exemples d'utilisation des différentes options proposées autour du Loupiot : Bluetooth, télémètre à ultrasons, détection d'obstacles, porte-stylo.

Liste des programmes du niveau 3

Nom du fichier	Description	Objectif
Niveau 3 A - option Bluetooth **Fichier modèle** : LP_N3_A.xml		
LP_N3_A1	Recevoir une donnée	Fonctionnalité matérielle abordée : - Option Bluetooth Notions de programmation abordées : - Communication sans fil
LP_N3_A2	Envoyer une donnée	
LP_N3_A3	Envoyer et recevoir des données	
LP_N3_A4	Afficher l'état des capteurs de ligne	
LP_N3_A5	Contrôler le Loupiot avec une télécommande	
LP_N3_A6	Contrôler le Loupiot à la voix en Bluetooth	
LP_N3_A7	Mesurer la vitesse de base du Loupiot en cm/s	

Niveau 3 B - option télémètre à ultrasons **Fichier modèle**: LP_N3_B.xml		
LP_N3_B1	Lire une distance avec le debug	Fonctionnalité matérielle abordée : - Option télémètre à ultrasons
LP_N3_B2	Radar de proximité	
LP_N3_B3	Suivi de ligne avec évitement d'obstacle	

Niveau 3 C - option détection d'obstacle **Fichier modèle**: LP_N3_C.xml		
LP_N3_C1	Prévenir la présence d'un obstacle	Fonctionnalité matérielle abordée : - Option détection d'obstacle
LP_N3_C2	Suivi de ligne avec évitement d'obstacle	

Niveau 3 D - option porte-stylo **Fichier modèle**: LP_N3_D.xml		
LP_N3_D1	Dessiner une forme géométrique	Fonctionnalité matérielle abordée : - Option porte-stylo
LP_N3_D2	Remplir au mieux une zone délimitée	

Niveau 3 E - option pistes robotiques (circuit / suivi de ligne) **Fichier modèle**: LP_N3_E.xml		
LP_N3_E_CIRC-LP1	Evoluer sur la piste circuit CIRC-LP1	Fonctionnalité matérielle abordée : - Option pistes robotiques
LP_N3_E_SDL_LP1	Suivre une ligne sur la piste SDL-LP1	
LP_N3_E_SDL_LP2	Suivre une ligne sur la piste SDL-LP2	
LP_N3_E_SDL_LP3	Suivre une ligne sur la piste SDL-LP3	

Niveau 3 A - Option Bluetooth

L'option Bluetooth permet d'établir une communication sans fil bidirectionnelle entre un smartphone Android et le robot Loupiot. Les applications proposées sont réalisées sous AppInventor 2.

Pour utiliser les programmes / applications de l'option Bluetooth, il faut :

1) Charger le programme (fichier **.xml**) dans le Loupiot à partir de Picaxe Editor 6.
2) Activer le Bluetooth et appairer votre smartphone ou tablette au Loupiot.
3) Installer l'application correspondante (fichier **.apk**) sur votre smartphone.
4) Lancer l'application et utiliser le Loupiot en Bluetooth.

Note : pour pouvoir installer les applications d'A4, il faut autoriser l'installation d'applications de sources inconnues dans les paramètres de sécurité de votre smartphone.

Important : si vous rencontrez des problèmes de connexion, il peut s'avérer nécessaire de relancer l'appairage, notamment si vous utilisez plusieurs robots ou plusieurs smartphones.

Pour créer / modifier une application avec AppInventor :

- avoir un compte Gmail pour la création d'une session.
- télécharger le fichier modèle « **Bluetooth_base.aia** ».

Il s'agit du fichier de base proposé par A4 pour la communication Bluetooth. Il peut être apparenté au fichier 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.

Exercices

Pour chaque exercice, nous proposons des fichiers de correction :

- un fichier **.xml** à charger dans le robot.
- un fichier **.apk** à installer sur le smartphone.
- le fichier **.aia** modifiable avec AppInventor.

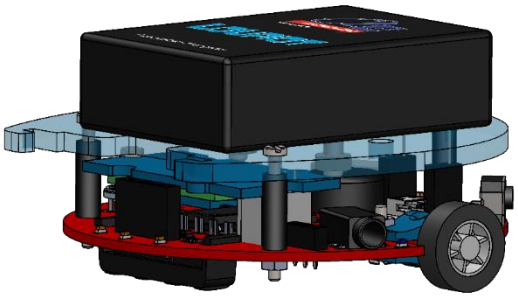
Les trois premiers programmes (**LP-N3-A1 à LP-N3-A3**) sont donnés dans le but d'apprendre à créer une application AppInventor puis le code correspondant pour le robot.

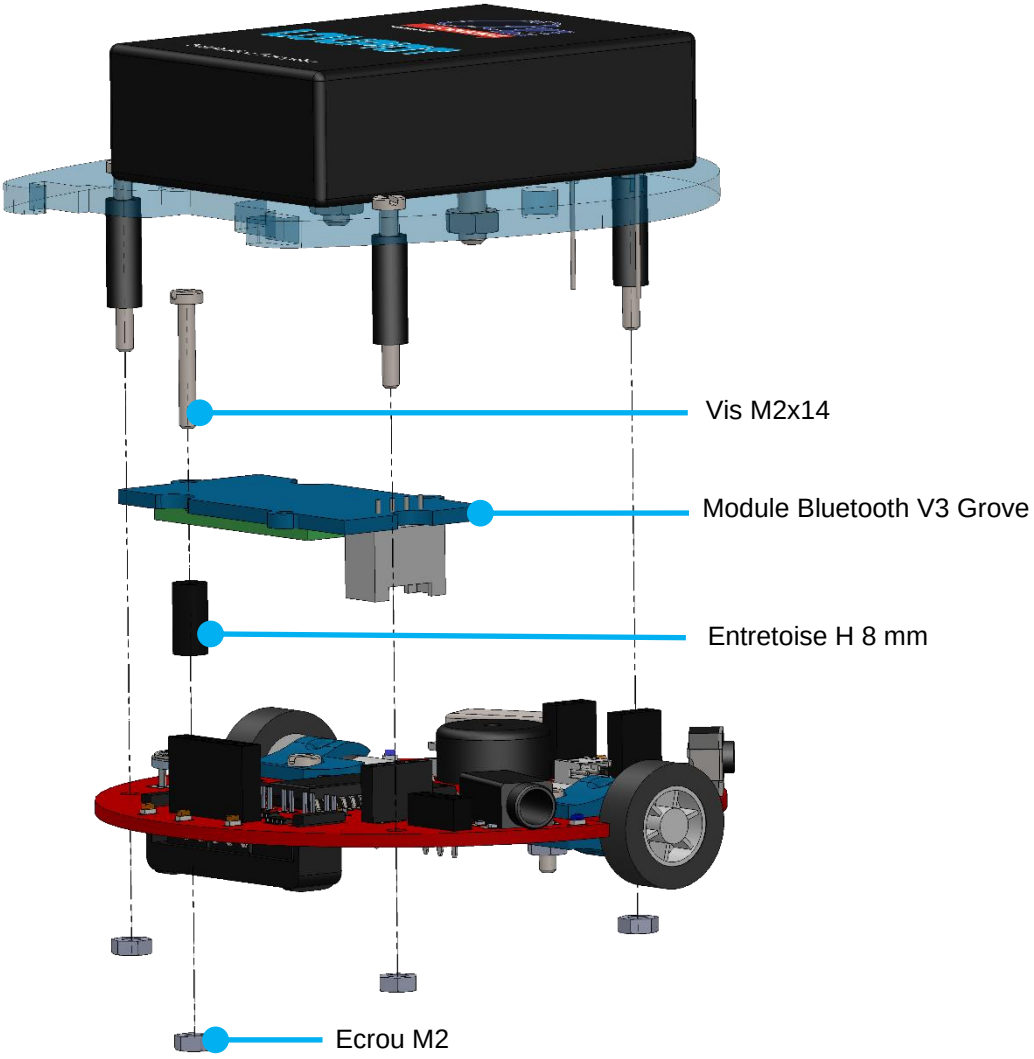
Quatre autres programmes (**LP-N3-A4 à LP-N3-A7**) sont donnés mais leurs codes ne sont pas commentés.
Les explications portent uniquement sur le principe général de l'exemple et sur le fonctionnement de l'application AppInventor afin que vous puissiez utiliser directement la version compilée.
Le but est de se concentrer sur le codage du robot en utilisant l'application déjà faite.
Le fichier contenant le code de l'application AppInventor est donné afin que vous puissiez regarder le fonctionnement par vous-même si cela vous intéresse.

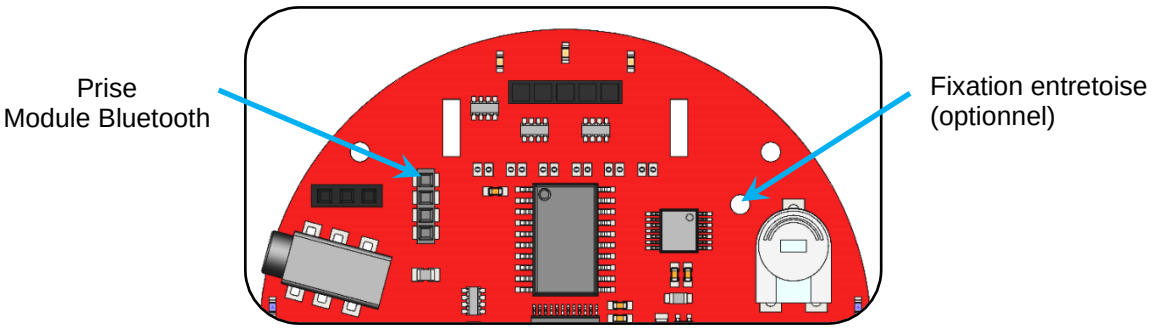
Chaque application possède deux versions : une application avec des graphismes simples et une application avec des graphismes plus évolués pour que vous puissiez voir comment agir sur les paramètres graphiques et modifier le design de l'interface utilisateur.

Montage option Bluetooth

1) Démonter la coque du Loupiot afin de pouvoir positionner le module à son emplacement sur le circuit.
2) Brancher le module Bluetooth sur le connecteur prévu sur le circuit (il est également possible de monter une entretoise à l'autre extrémité du capteur pour améliorer sa fixation comme indiqué ci-dessous).
3) Remonter la coque. (Vérifier que les picots du support de piles se sont bien insérés dans les connecteurs d'alimentation).







Exercice niveau 3 - A1 : Recevoir des données

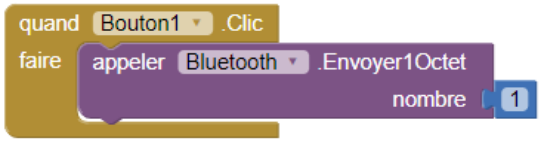
Fichier modèle : LP_N3_A.xml / Bluetooth_base.aia

Objectif : Envoyer à partir d'un bouton sur l'application Android, une consigne pour faire avancer le robot pendant une seconde.

Notion(s) abordée(s) : Réception de données du smartphone vers le robot.

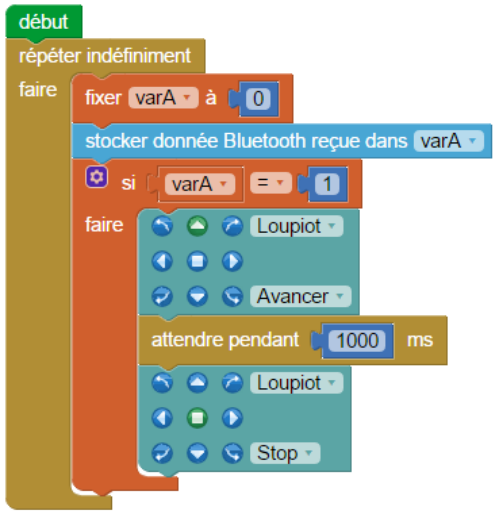
Correction :

AppInventor



Fichier AppInventor : LP_N3_A1.aia / LP_N3_A1.apk

Blockly



Fichier Blockly : LP_N3_A1.xml

Remarque(s) :

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

Exercice niveau 3 - A2 : Envoyer des données

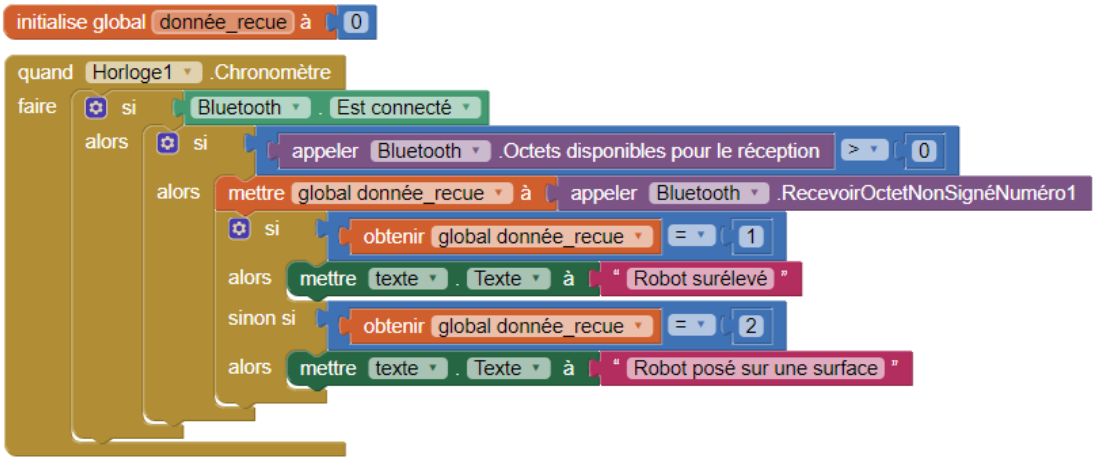
Fichier modèle : LP_N3_A.xml / bluetooth_base.aia

Objectif : Afficher sur le smartphone un message indiquant si le robot détecte une surface claire ou foncée.

Notion(s) abordée(s) : Envoi de données du robot vers le smartphone.

Correction :

AppInventor

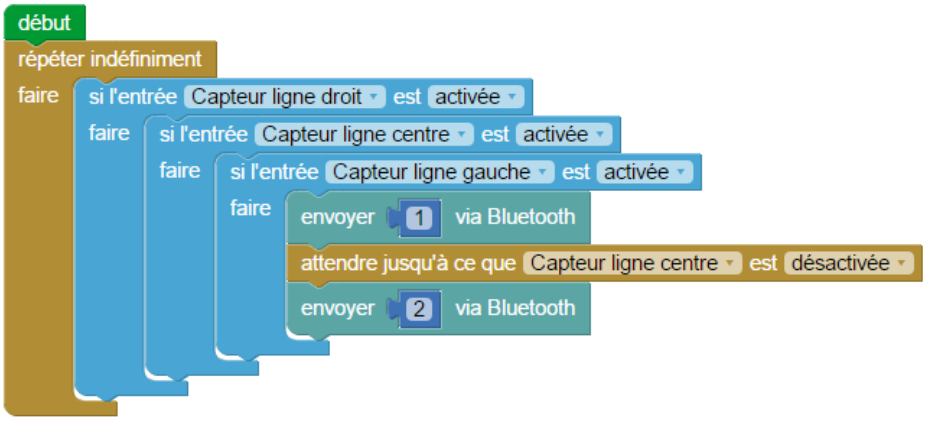


Fichier AppInventor : LP_N3_A2.aia / LP_N3_A2.apk

Remarque(s) :

- La première condition contenue dans le bloc du composant horloge sert à vérifier que le robot est bien connecté avant d'utiliser les blocs reliés au composant Bluetooth.
- 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.
- Un composant horloge a été placé dans la façade « designer » d'AppInventor. Il permet dans la façade « block » d'obtenir le bloc « quand horloge1.chronomètre » qui exécute les blocs qu'il contient toutes les 50 ms (temps de déclenchement paramétrable dans les paramètres du composant).

Blockly



Fichier Blockly : LP_N3_A2.xml

Exercice niveau 3 - A3 : Envoyer et recevoir des données

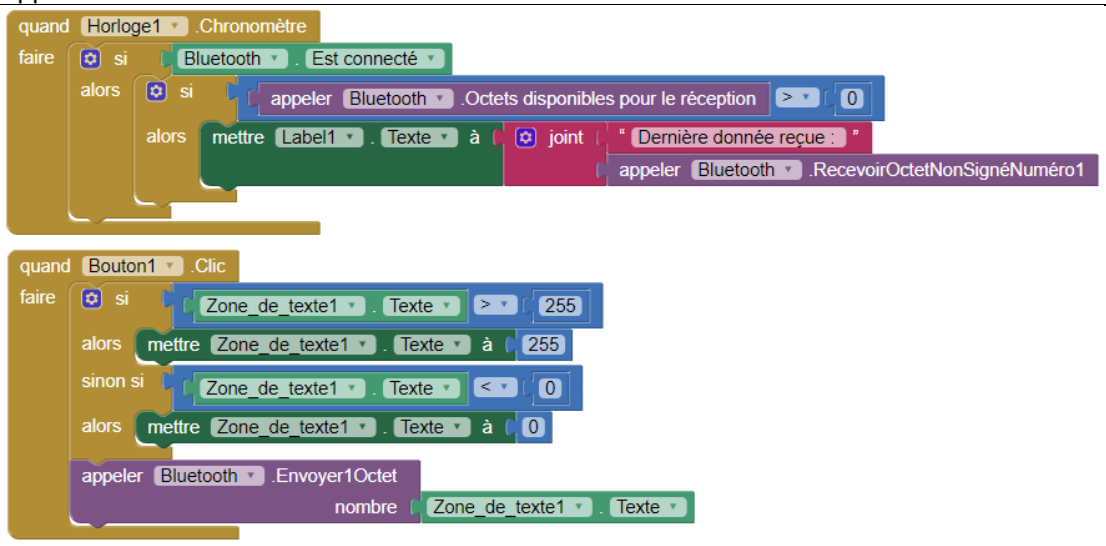
Fichier modèle : LP_N3_A.xml / bluetooth_base.aia

Objectif : Le robot renvoi au smartphone la donnée qu'il reçoit de celui-ci.

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

Correction :

AppInventor

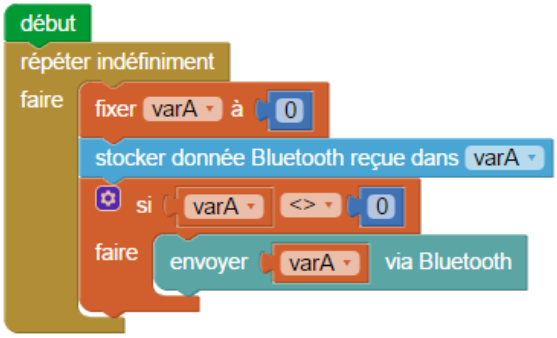


Fichier AppInventor : LP_N3_A3.aia / LP_N3_A3.apk

Remarque(s) :

- Quand on envoie une donnée en appuyant sur le « bouton1 » la première étape, pour éviter une erreur de l'application, est de vérifier si la valeur rentrée par l'utilisateur dans la « **Zone_de_texte1** » est bien comprise entre 0 et 255 (intervalle des valeurs envoyables). Sinon on arrondit à la valeur limite la plus proche.
- Comme pour l'exercice précédent, un composant horloge a été placé dans la façade « designer » pour pouvoir demander au smartphone toutes les 50 ms si une donnée a été reçue en provenance du robot.

Blockly



Fichier Blockly : LP_N3_A3.xml

Ex supplémentaire niv.3 - A4 : Afficher l'état des capteurs de ligne

Fichier corrigé :
- Code du robot : LP_N3_A4_Methode_1.xml / LP_N3_A4_Methode_2.xml
- Application AppInventor de base : LP_N3_A4_V1.aia / LP_N3_A4_V1.apk
- Application AppInventor avec graphismes améliorés : LP_N3_A4_V2.aia / LP_N3_A4_V2.apk

Objectif : Le smartphone affiche l'état des capteurs de ligne.

Fonctionnement de l'application AppInventor :
L'application LP_N3_A4.apk attend de recevoir des données pour paramétrer l'état de ses afficheurs correspondant au 3 capteurs de ligne.

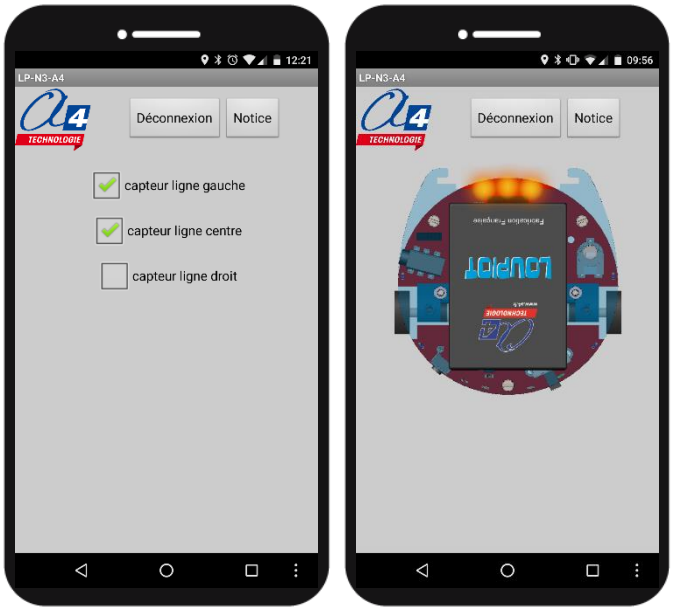
Voici la liste des consignes à envoyer au smartphone en fonction de l'état de chaque capteur :
- Si le capteur de ligne droit est activé, envoyer la consigne « 4 », sinon envoyer la consigne « 5 ».
- Si le capteur de ligne centre est activé, envoyer la consigne « 3 », sinon envoyer la consigne « 4 ».
- Si le capteur de ligne gauche est activé, envoyer la consigne « 1 », sinon envoyer la consigne « 2 ».

Il existe deux méthodes pour envoyer des données au smartphone :

- **Méthode 1** (LP_N3_A4_Methode_1.xml) :
  Méthode la plus simple.
  Le robot envoie l'état de ses capteurs de ligne tous les x temps (ici x = 100 ms).
  L'inconvénient de cette méthode est que le smartphone peut vite se retrouver saturé. Les données peuvent arriver trop vite pour qu'il ait le temps de les traiter.
  De plus, même si l'état des capteurs de ligne du robot ne change pas, celui-ci redonne en boucle la même information au smartphone qui va refaire la même action.

- **Méthode 2** (LP_N3_A4_Methode_2.xml) :
  Le robot communique avec le smartphone seulement lorsqu'il y a un changement sur l'un de ses capteurs de ligne. Le robot retient dans des variables l'état de ses capteurs et les compare avec leurs nouveaux états pour savoir s'il y a eu un changement.

Capture d'écran de deux versions de l'application :



Ex supplémentaire niv.3 - A5 : Contrôler le Loupiot avec une télécommande

Fichier corrigé :

- Code du robot : LP_N3_A5.xml
- Application AppInventor de base : LP_N3_A5_V1.aia / LP_N3_A5_V1.apk
- Application AppInventor avec graphismes améliorés : LP_N3_A5_V2.aia / LP_N3_A5_V2.apk

Objectif : Contrôler les différents mouvements du robot et sa vitesse à partir d'une application télécommande AppInventor.

Fonctionnement de l'application AppInventor :

Les 4 boutons de direction de la télécommande envoient chacun une consigne lorsqu'ils sont enfoncés :

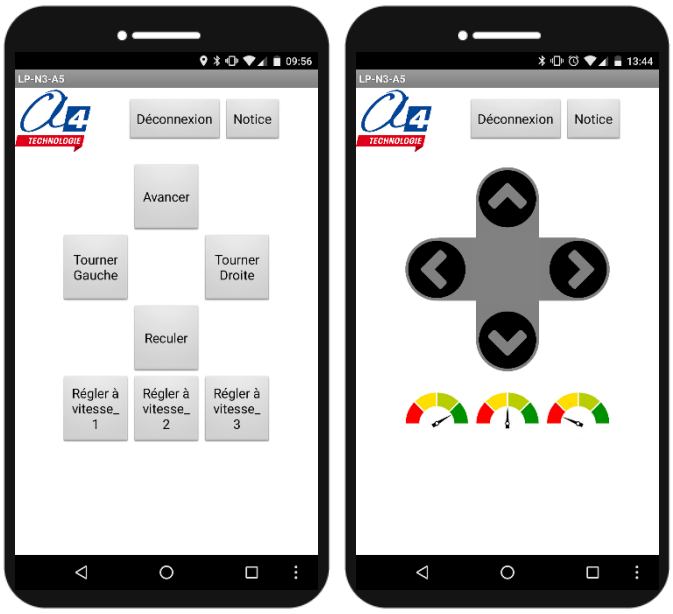
- « 1 » : avancer
- « 2 » : tourner à droite
- « 3 » : reculer
- « 4 » : tourner à gauche.

Ils renvoient tous la même consigne « 5 » lorsqu'ils sont relâchés pour signifier au robot d'arrêter son mouvement.

Les 3 boutons de vitesse envoient une consigne lors d'un clic (enfoncé + relâché) :

- « 6 » : vitesse faible
- « 7 » : vitesse modérée
- « 8 » : vitesse max.

Capture d'écran de deux versions de l'application :



Ex supplémentaire niv.3 - A6 - Commander Loupiot à la voix en Bluetooth

Fichier corrigé :

- Code du robot : LP_N3_A6.xml
- Application AppInventor de base : LP_N3_A6.aia / LP_N3_A6.apk

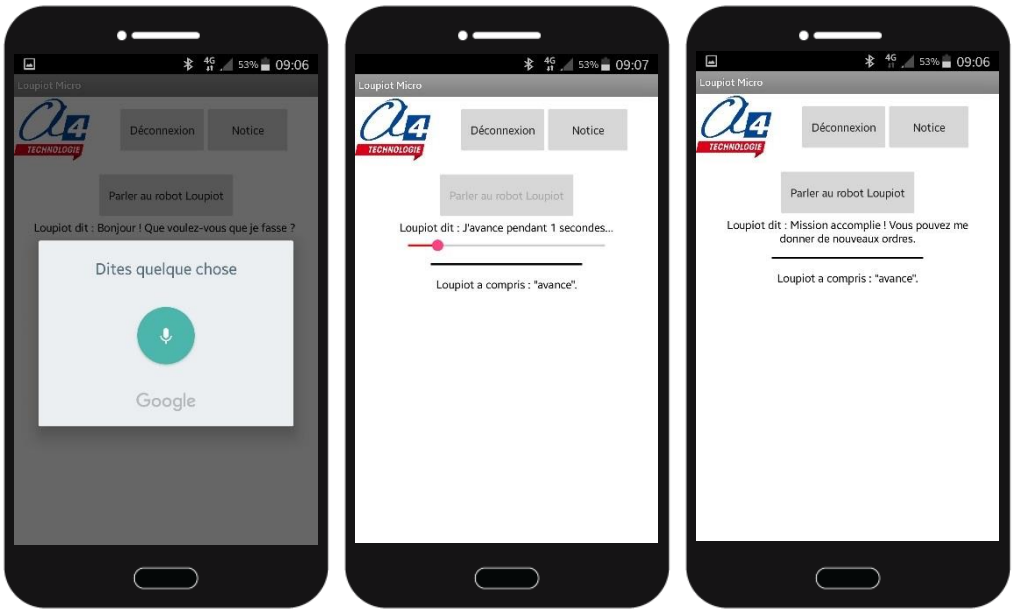
Objectif : Programmer par la voix une séquence de mouvements que le robot loupiot va exécuter (actions + temps d'exécution des actions).

Note : la reconnaissance vocale est réalisée sur les serveurs de Google et nécessite donc une connexion internet permanente lors de l'utilisation de l'application.

Fonctionnement de l'application AppInventor :

1. Cliquer sur le bouton « Parler au robot Loupiot ».
2. Une fenêtre de prise de voix s'ouvre, attendre que celle-ci vous invite à parler.
3. Énoncer la séquence de mouvements/instructions à effectuer.
4. Une fois que vous arrêtez de parler, la fenêtre se ferme et le robot exécute les instructions.
5. Une fois la séquence d'instructions terminée, le bouton « Parler au robot Loupiot » redevient disponible et vous pouvez de nouveau lui donner une séquence d'instructions.

Important : Après la prise de voix, le robot Loupiot écrit la phrase qu'il a entendue.
Cela permet parfois de comprendre pourquoi une instruction n'a pas été exécutée. En effet, la reconnaissance vocale de Google peut se tromper de mot ou l'orthographier d'une manière imprévue.



Lors de l'exécution des instructions, une barre de progression vous indique où en est le Loupiot.

Loupiot connait les instructions : **Avancer / Avance, Reculer / Recule, Tourner / Tourne droite ou gauche**.

Pour enchainer plusieurs instructions, il faut les séparer par le mot-clé « **puis** ».

Vous pouvez indiquer un temps d'exécution d'une instruction avec le mot-clé **« secondes »**

Par défaut, si aucune durée n'est spécifiée, le temps d'exécution de l'instruction est de 1 seconde.

Par défaut, si aucun sens de rotation n'est spécifié pour l'instruction « tourne », le robot tourne à gauche.

Si une instruction n'est pas comprise par le robot, elle n'est pas exécutée.

Si plusieurs instructions sont dites sans être séparées par le mot-clé « puis », seule la première est exécutée.

Exemple : pour la formulation « avance puis accélère puis recule », le robot avance puis recule pendant une seconde. En effet, le mot « accélère » est ignoré car il ne fait pas partie du vocabulaire du Loupiot.

Exemples non exhaustifs de formulations possibles pour la séquence suivante :
Le robot avance pendant 2 secondes puis tourne 3 secondes à gauche et enfin recule 1 seconde.

- « Avancer pendant 2 secondes puis tourner à gauche pendant 3 secondes puis recule pendant 1 seconde » ;
- « Pendant 2 secondes avance puis tourne pendant 3 secondes puis recule » ;
- « Avance 2 secondes puis tourne durant 3 secondes puis recule une seconde » ;
- « 2 secondes avance puis tourne à gauche 3 secondes puis recule ».

Ex supplémentaire niv.3 - A7 : Mesurer la vitesse de base du Loupiot en cm/s

Fichier corrigé :

- Code du robot : LP_N3_A7.xml
- Application AppInventor de base : LP_N3_A7_V1.aia / LP_N3_A7_V1.apk
- Application AppInventor avec graphismes améliorés : LP_N3_A7_V2.aia / LP_N3_A7_V2.apk

Objectif : Mesurer à partir de deux temps enregistrés sur l'application AppInventor la vitesse de base du robot Loupiot.
Quand le chrono est lancé, le robot reçoit la consigne et commence à avancer.
Le robot rencontre ensuite deux lignes espacées de X cm (placées au préalable sur son chemin) et envoie pour chaque ligne la consigne permettant d'enregistrer un temps intermédiaire.
L'appui sur le bouton « Stop » de l'application déclenche l'arrêt du robot.
En calculant le temps T que le robot a pris pour traverser les deux lignes, grâce au deux temps enregistrés dans l'application AppInventor, et l'espacement des deux lignes noires, on détermine la vitesse moyenne V du robot : V = X / T avec X en cm et T secondes.

Fonctionnement de l'application AppInventor :
L'application LP_N3_A7.apk est un chronomètre qui permet d'enregistrer des temps.
Il peut se contrôler manuellement à partir des boutons de l'application ou par Bluetooth à partir du robot en envoyant des consignes au smartphone.

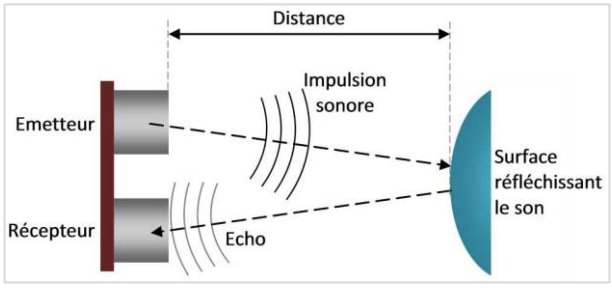
Note : quand un bouton est appuyé, il envoie la même consigne qu'on aurait envoyée au smartphone pour activer la fonction.
Ci-dessous, la liste des consignes à envoyer au smartphone ou envoyées par le smartphone pour chaque fonction de l'application :

- « 1 » : Lancer le chrono
- « 2 » : Mettre en pause le chrono
- « 3 » : Stopper le chrono (remise à zéro)
- « 4 » : Enregistrer un temps intermédiaire
- « 5 » : Vider la liste de temps enregistrés
- « 6 » : Enregistrer les temps dans un fichier texte

Capture d'écran de deux versions de l'application :

Niveau 3 B - Option télémètre à ultrasons

L'option télémètre 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.

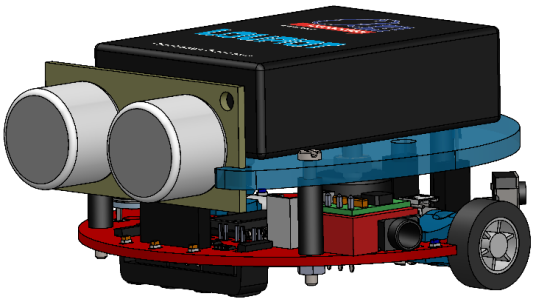


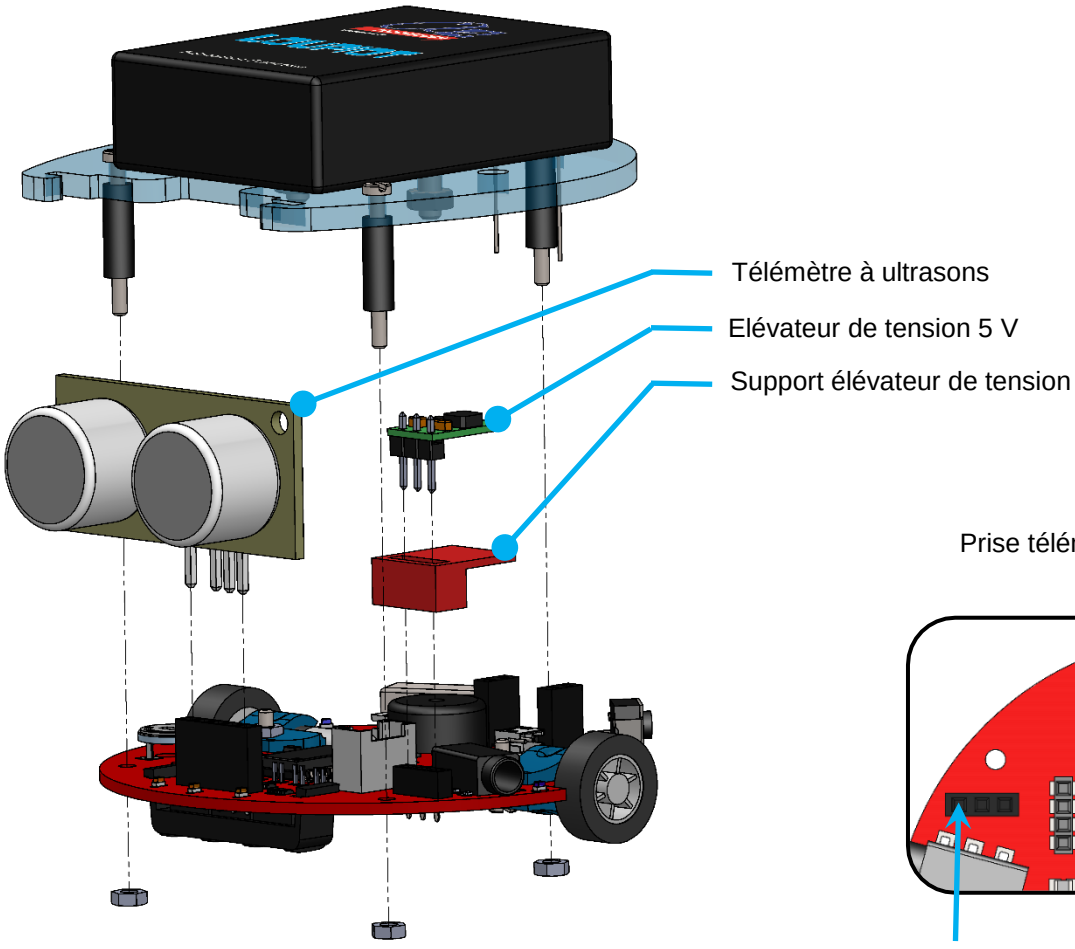
Mesure des distances de 2 à 255 cm.
Une LED rouge s'allume brièvement sur la face arrière du capteur pour indiquer que la mesure a bien été effectuée.
Si la LED est en permanence allumée cela veut dire que le capteur lance des mesures en permanence.

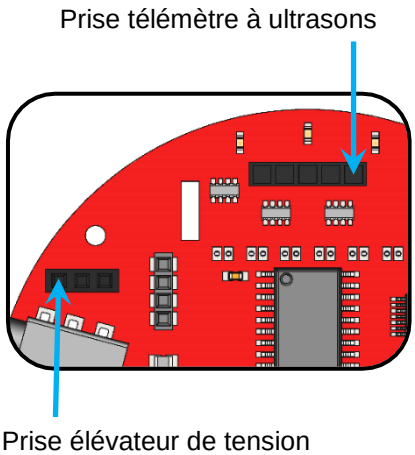
Montage option télémètre à ultrasons

Le kit contient deux composants : un télémètre à ultrasons permettant de mesurer la distance et un élévateur de tension 5 V qui alimente le télémètre à ultrasons.

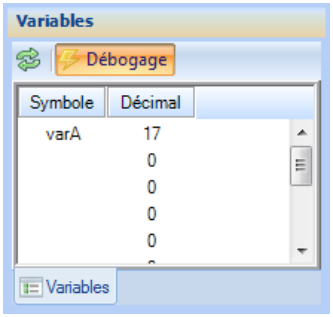
1) Démonter la coque du Loupiot.
2) Positionner le support de l'élévateur de tension sur le connecteur du circuit.
3) Brancher l'élévateur de tension sur le support.
4) Remonter la coque. (Vérifier que les picots du support de piles se sont bien insérés dans les connecteurs d'alimentation).
5) Brancher le télémètre à ultrasons à l'emplacement prévu sur le circuit à l'avant du robot.







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

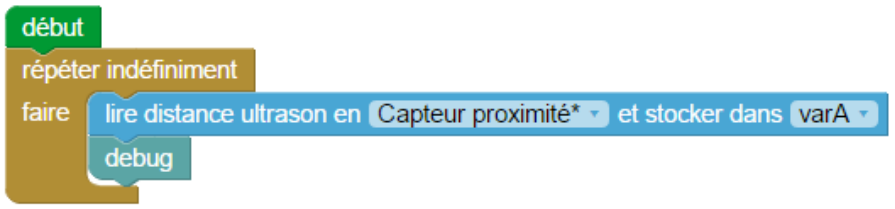


Fichier modèle : LP_N3_B.xml

Objectif : Afficher la valeur mesurée par le télémètre à ultrasons dans la fenêtre Variables 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 dans la fenêtre Variables.
- 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 continu car il devrait y avoir un grand nombre de mesures par secondes. Or, on remarque que la LED clignote plutôt rapidement mais pas en continu. 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 biper le buzzer avec une fréquence variant en fonction de la distance mesurée par le télémètre à ultrasons (similaire à un radar de recul pour une voiture).

Correction :

Blocs

Fichier Blockly : LP_N3_B2.xml

Remarque(s) :

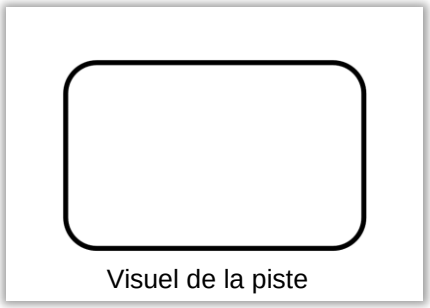
- Le bip a une durée fixe de 35 ms. On fait varier seulement la durée entre chaque bip.
- Le bloc « jouer son … pendant … sur … » n'accepte pas de variable dans la case durée.

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

Fichier modèle : LP_N3_B.xml

Objectif : A l'appui du bouton-poussoir, le robot suit une ligne fine et s'arrête quand un obstacle à moins de 6 cm est détecté.

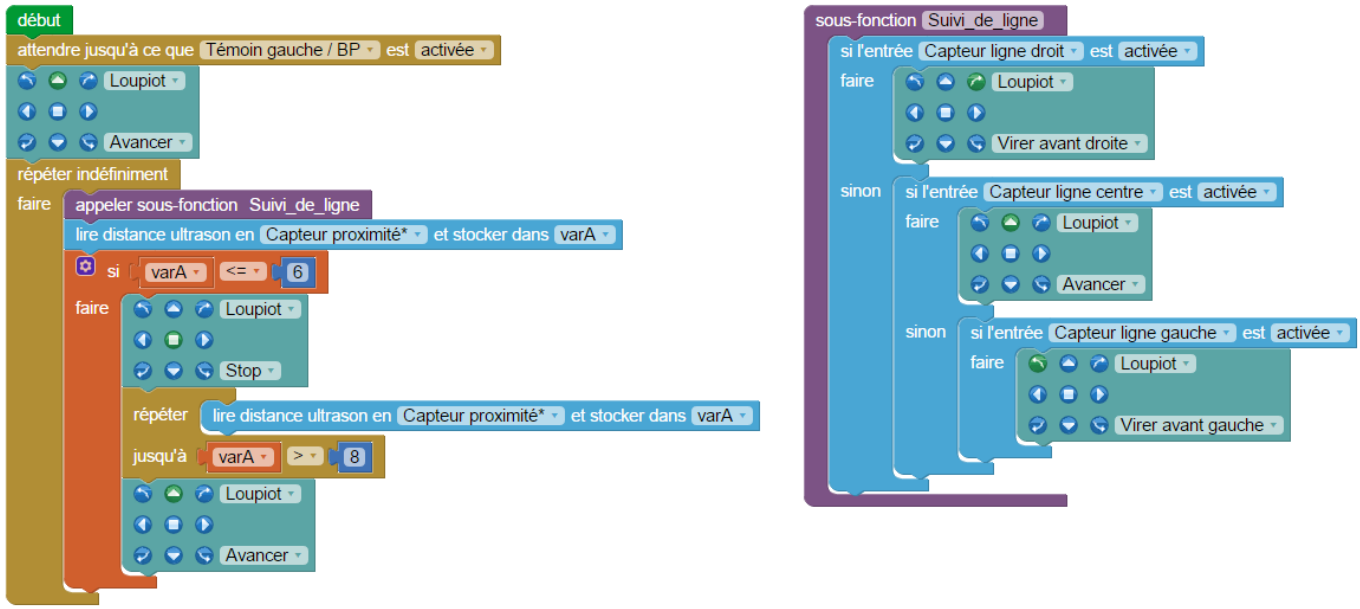
Note : Imprimer la piste "Piste_suivi_ligne_type1" pour tester ce code.



Visuel de la piste

Correction :

Blocs



Fichier Blockly : LP_N3_B3.xml

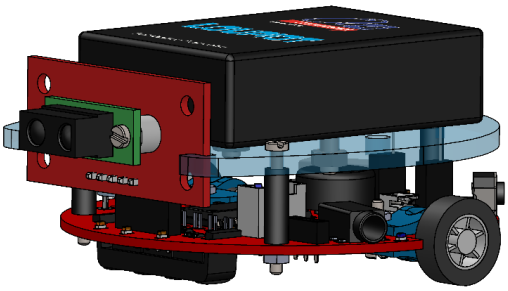
Remarque(s) :

- Le robot attend que le bouton-poussoir soit appuyé pour lancer le programme. Cela évite que le robot démarre directement après le transfert du programme et arrache le câble de programmation.
- 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.

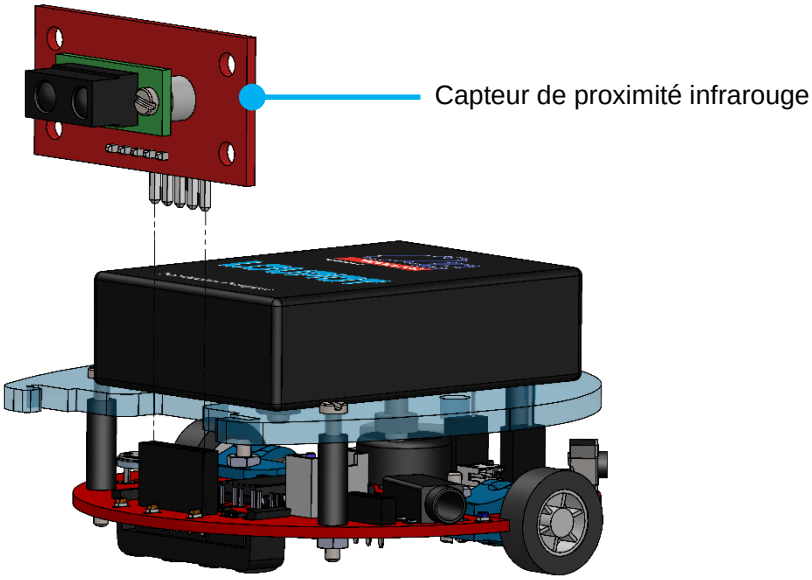
Niveau 3 C - Option détection d'obstacle

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.



Montage option détection d'obstacle

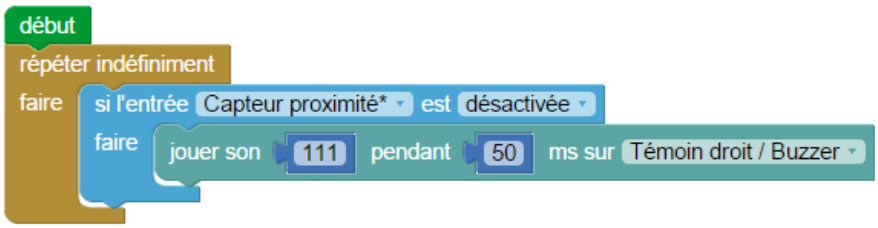


Exercice niveau 3 – C1 : Prévenir la présence d'un obstacle

Fichier modèle : LP_N3_C.xml

Objectif : Le buzzer bipe 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

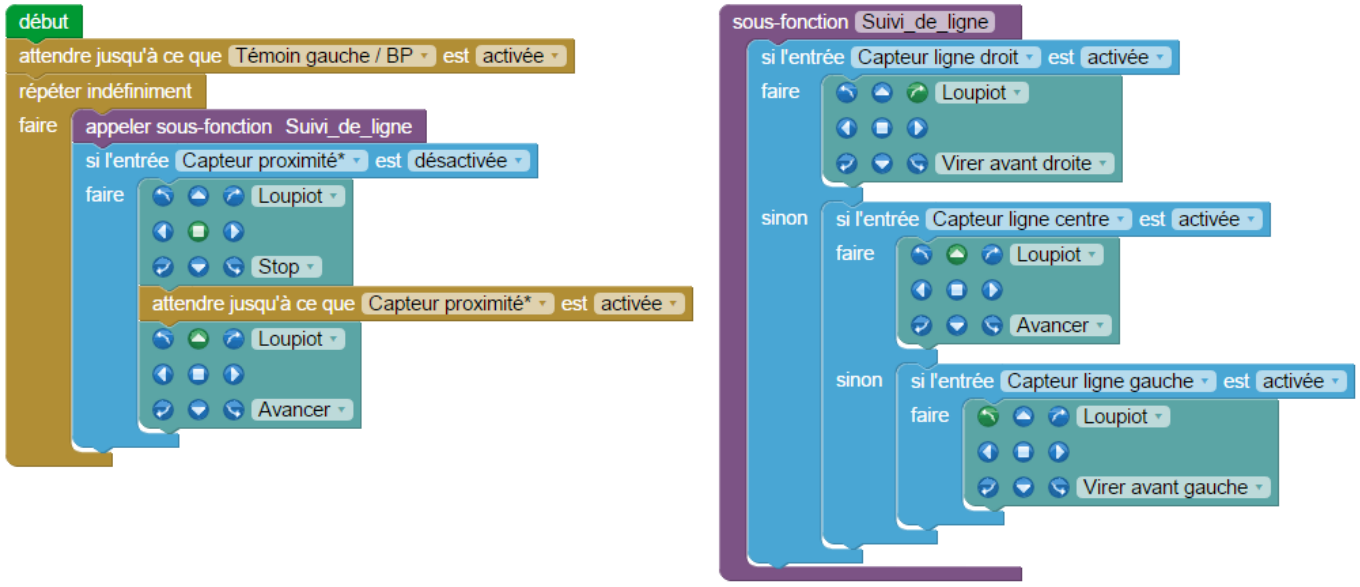
Objectif : A l'appui du bouton-poussoir, le robot suit une ligne fine et s'arrête quand un obstacle à moins de 5 cm est détecté.

Note : Imprimer la piste « Piste_suivi_ligne_type1 » pour ce programme.

Visuel de la piste

Correction :

Blocs



Fichier Blockly : LP_N3_C2.xml

Remarque(s) :

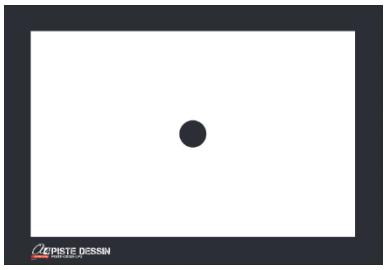
- Le robot attend que le bouton-poussoir soit appuyé pour lancer le programme.
- Cela évite que le robot démarre directement après le transfert du programme et arrache le câble de programmation.
- Une procédure a été placée pour rendre le code principal plus lisible.

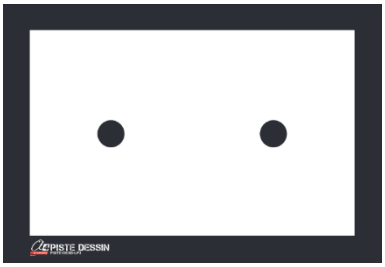
Niveau 3 D - Option porte-stylo

L'option porte-stylo permet de transformer le Loupiot en robot dessinateur.
La pièce clipsable à l'arrière du robot peut accueillir un stylo ou un marqueur de diamètre 9,5 mm.

En complément, nous vous proposons les éléments suivants :
Feutres effaçables à sec (réf. K-LP-FEUT).
Pack de 3 pistes effaçables + feutres (Réf. PISTE-PACK-DESS1).
Pack de 4 pistes robotique (Réf. PISTE-PACK-LP1).

Les 3 pistes dessin permettent de mener plusieurs activités, comme dessiner des formes géométriques : rond, soleil, etc.
Vous pouvez également réaliser des challenges de type « robot aspirateur » dans lesquels vous devez définir les algorithmes adaptés pour recouvrir le plus de surface possible (surface blanche ou avec obstacles matérialisés par des points noirs).





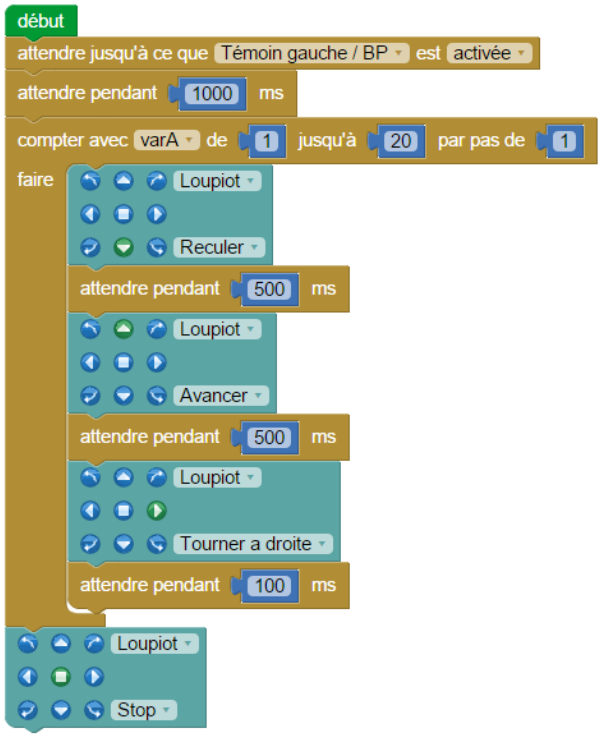
Exercice niveau 3 - D1 : Dessiner un motif géométrique

Fichier modèle : LP_N3_D.xml

Objectif : A l'appui du bouton-poussoir, dessiner un soleil en traçant un motif plusieurs fois.

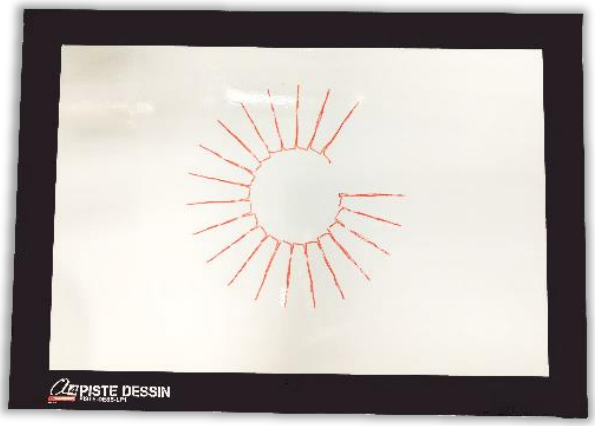
Correction :

Blocs



Fichier Blockly : LP_N3_D1.xml

Voici le résultat du motif dessiné par le robot.



Remarque(s) :

- Le robot attend que le bouton-poussoir soit appuyé pour lancer le programme. Cela évite que le robot ne démarre directement après le transfert du programme et n'arrache le câble de programmation.

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 de l'état des batteries du robot.

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

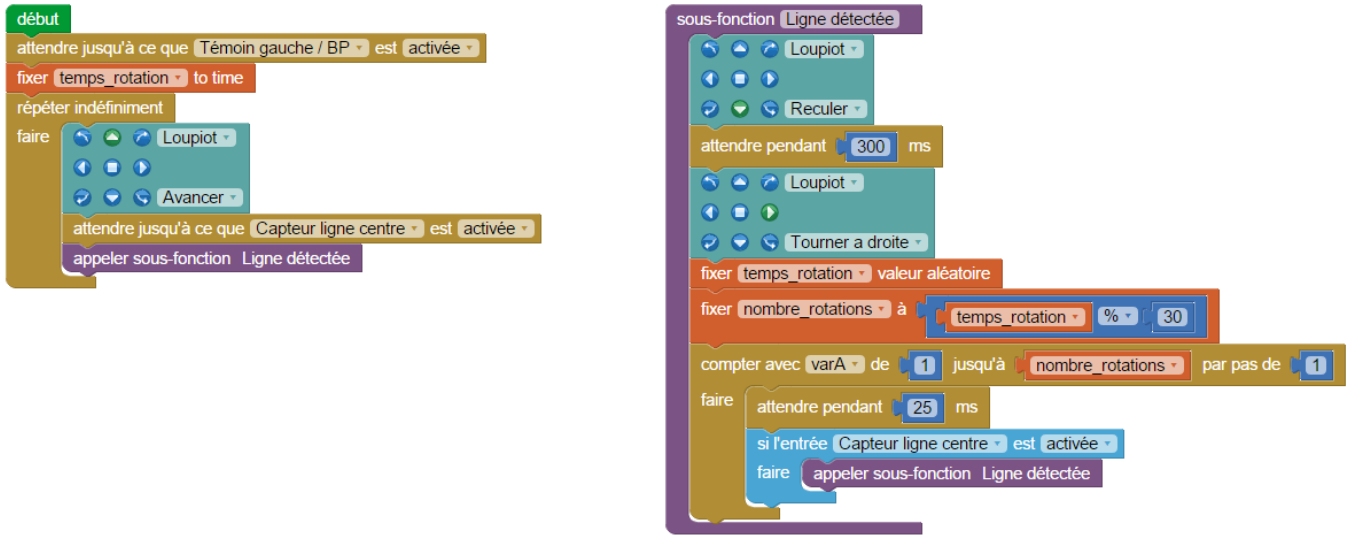
Fichier modèle : LP_N3_D.xml

Objectif : A l'appui du bouton-poussoir, remplir au maximum une zone délimitée par un contour noir, puis avec des obstacles.

Note : vous pouvez utiliser le programme « LP-N2-C3 » (Prison) comme base et l'améliorer pour optimiser la surface couverte par notre robot.

Correction :

Blocs



Fichier Blockly : LP_N3_D2.xml

Voici le résultat après 5 minutes.

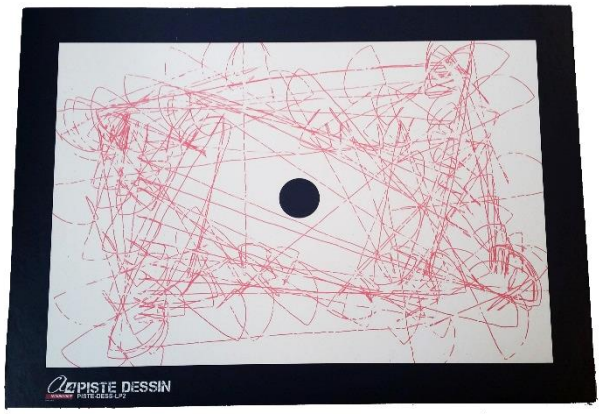
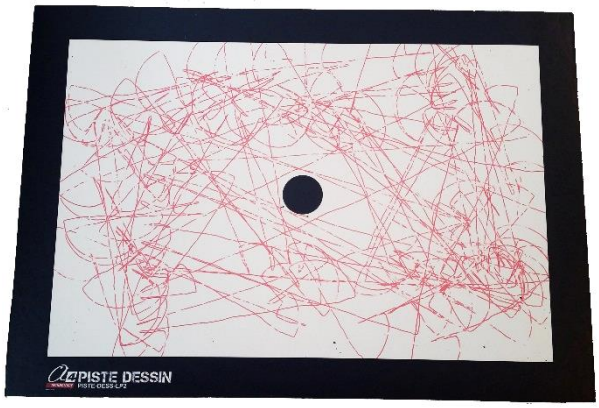
Avec le programme LP_N2_C3 | *Avec le programme LP_N3_D3*

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 … ».

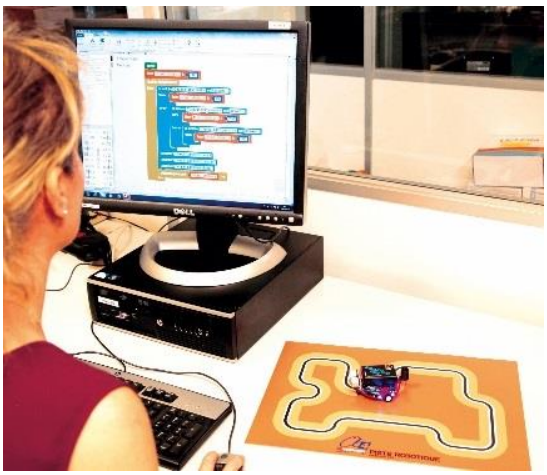
Remarque(s) :

- La variable « temps_rotations » doit être initialisée par une valeur différente à chaque fois pour avoir une vraie suite de nombres aléatoires. 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.
- 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.

On remarque que le programme pourrait encore être amélioré en retirant les valeurs aléatoires trop petites qui obligent le robot à tourner plusieurs fois de suite pour éviter une ligne noire ou des obstacles (matérialisés ici par les points noirs). Voir l'exemple LP-N3-D3.

	
Avec le programme LP_N3_D2	*Avec le programme LP_N3_D3*

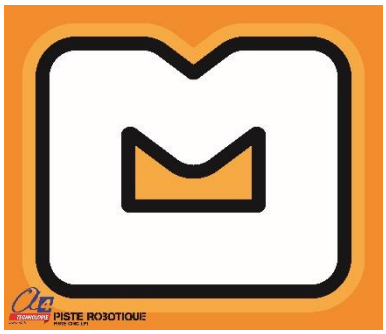
Niveau 3 E - Option pistes robotiques



Les pistes robotique permettent de mener des activités sur pistes : suivi de ligne, détection d'obstacles, …

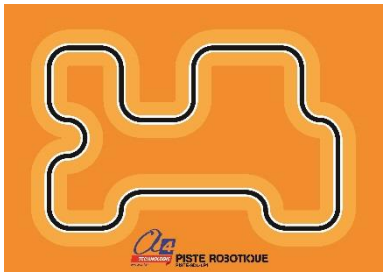
Imprimées sur bâche. Format A3.

- une piste circuit pour organiser des courses entre plusieurs Loupiots ;



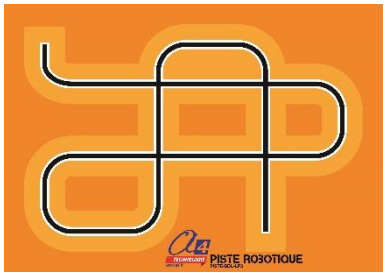
Réf. PISTE-CIRC-LP1

- 3 pistes suivi de ligne avec des parcours plus ou moins complexes pour des activités de programmation de difficulté croissante.



Réf. PISTE-SDL-LP1

Réf. PISTE-SDL-LP2



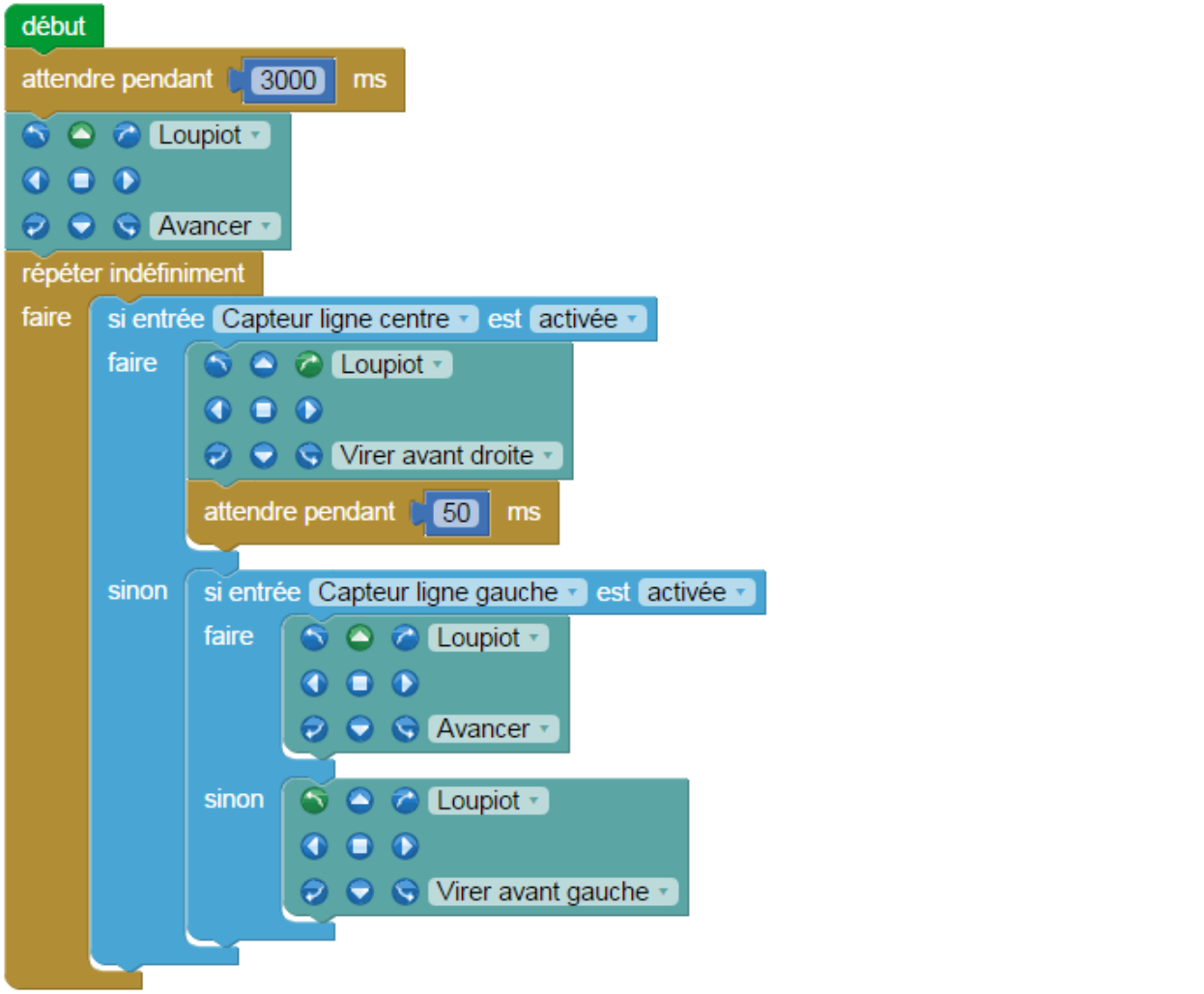
Réf. PISTE-SDL-LP3

Exercice niveau 3 - E-CIRC-LP1 : Evoluer sur la piste circuit

Fichier modèle : LP_N3_E.xml

Correction :

Blocs

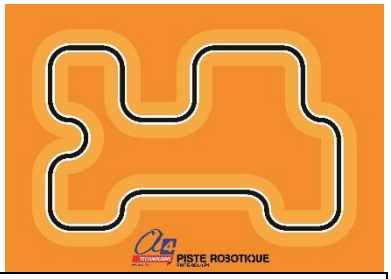



```
Tourner dans le sens des aiguilles d'une montre.
Attention: le robot doit être placé en conséquence au début du programme.
```

Fichier Blockly : LP_N3_E_CIRC_LP1.xml

Remarque(s) : Le robot doit être placé dans le bon sens au départ.

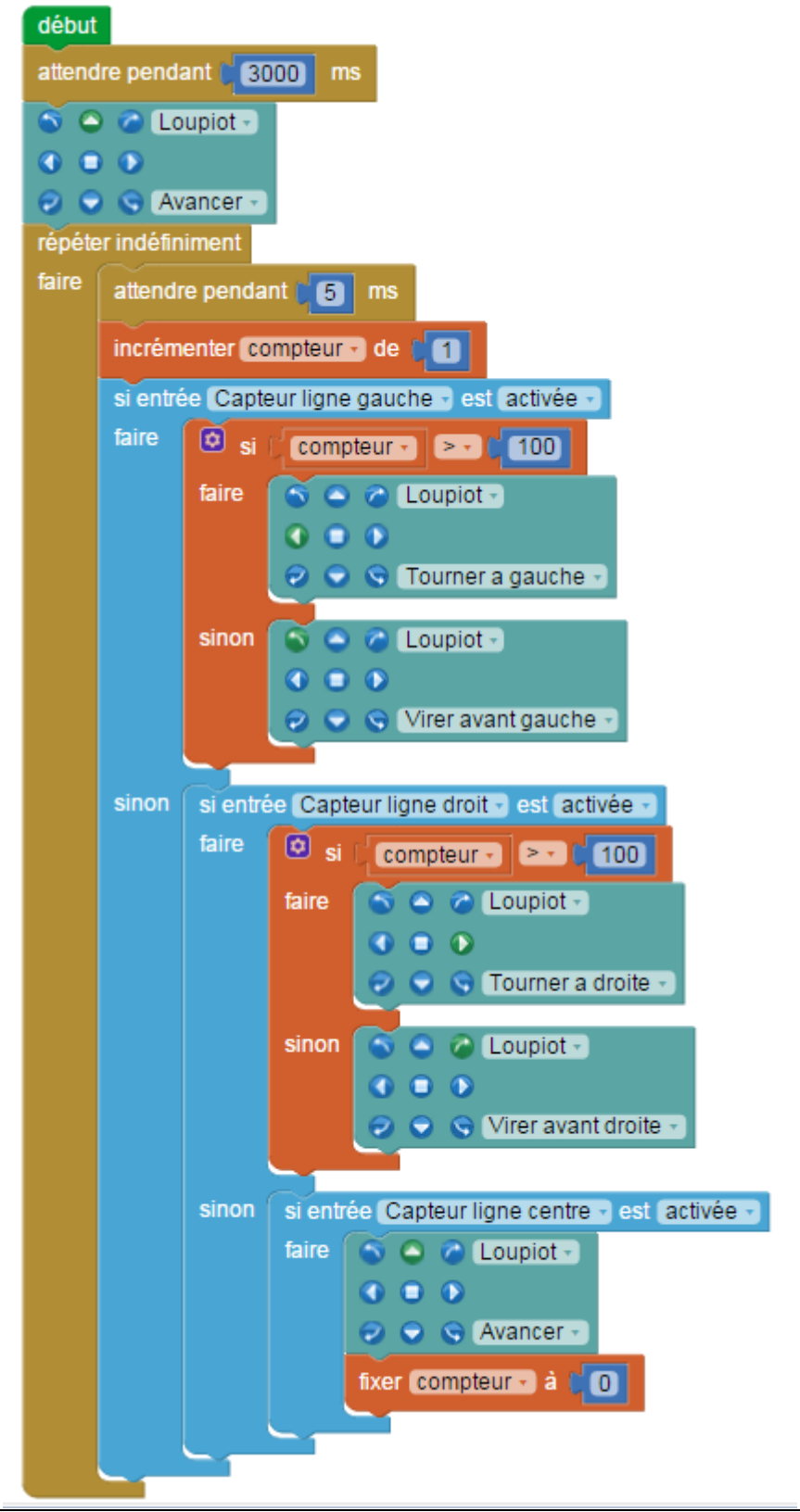
Exercice niveau 3 - E-SDL-LP1 : Suivre une ligne sur la piste SDL-LP1



Fichier modèle : LP_N3_E.xml

Correction :

Blocs



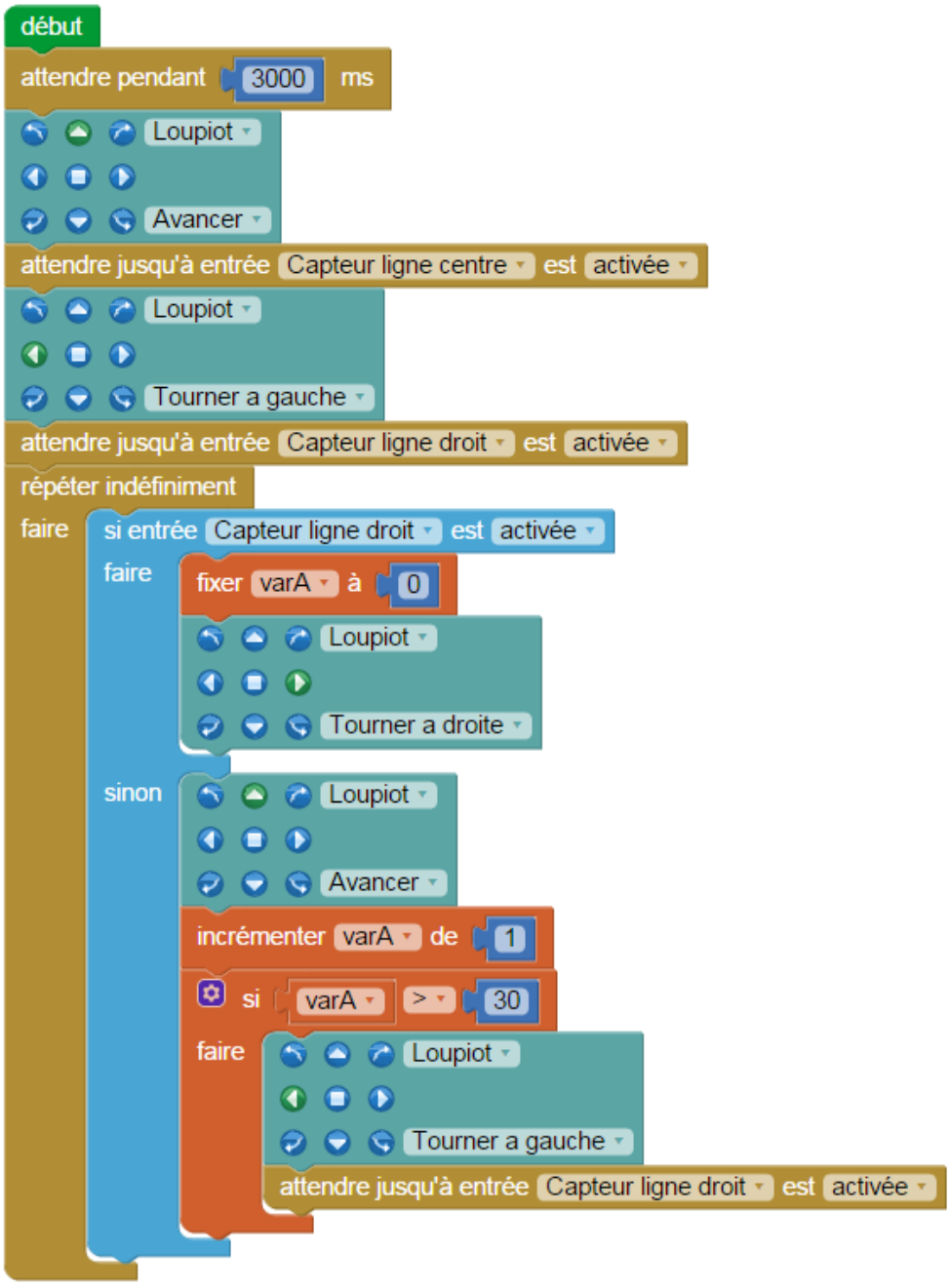
Fichier Blockly : LP_N3_E_SDL_LP1.xml

Exercice niveau 3 - E-SDL-LP2 : Suivre une ligne sur la piste SDL-LP2

Fichier modèle : LP_N3_E.xml

Correction :

Blocs



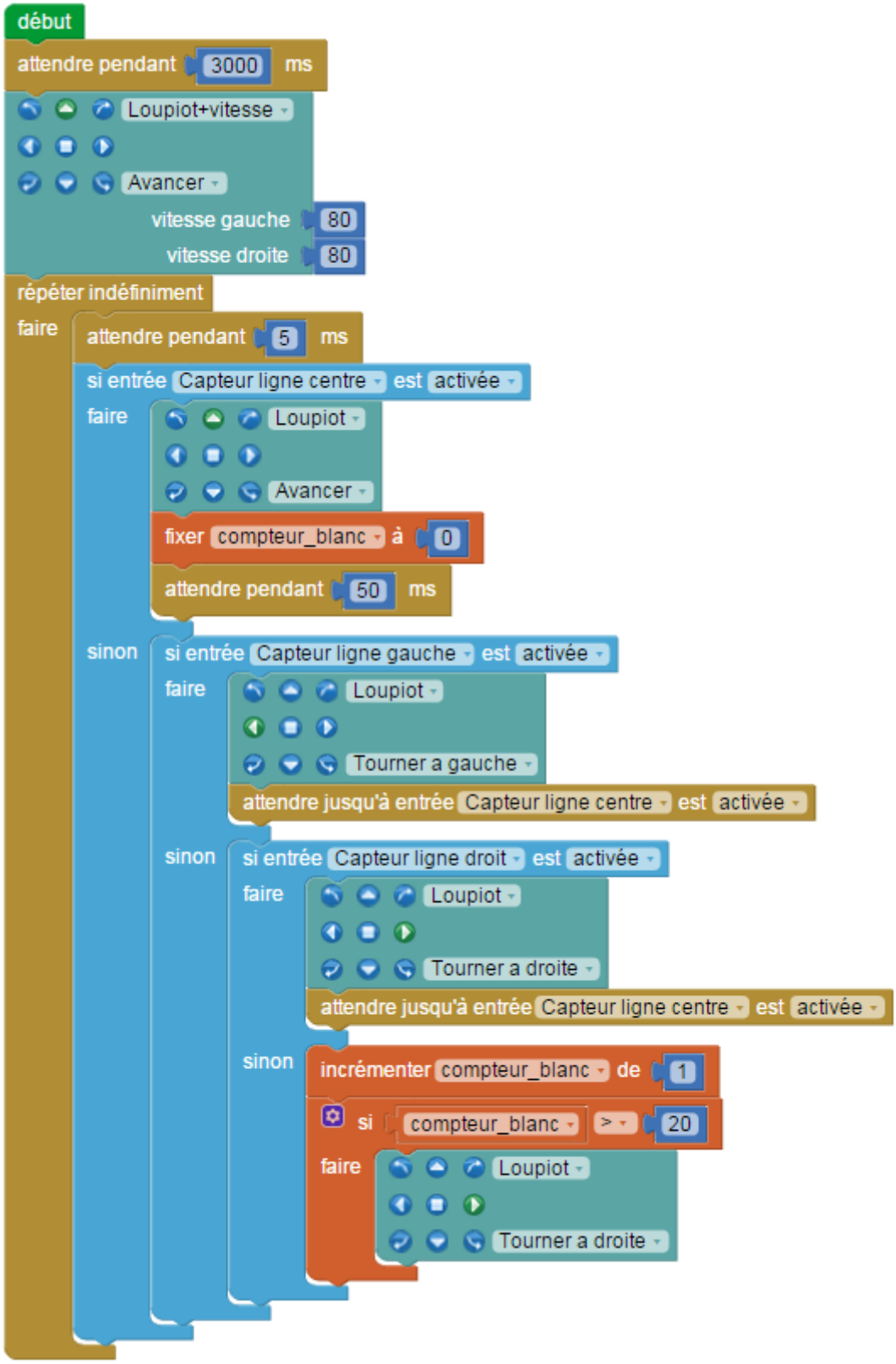
Fichier Blockly : LP_N3_E_SDL_LP2.xml

Exercice niveau 3 - E-SDL-LP3 : Suivre une ligne sur la piste SDL-LP3

Fichier modèle : LP_N3_E.xml

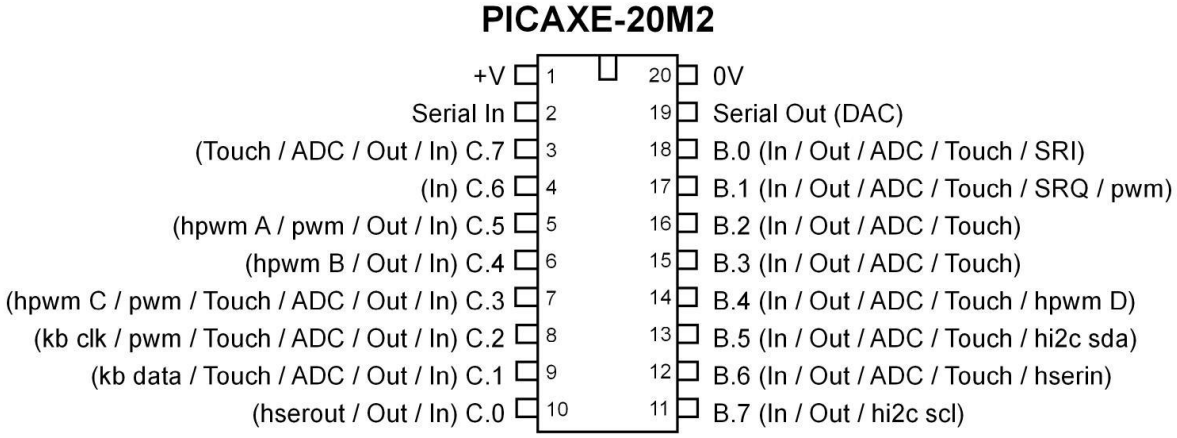
Correction :

Blocs



Fichier Blockly : LP_N3_E_SDL_LP3.xml

Schéma entrées / sorties microcontrôleur Picaxe 20M2



Etiquette Blockly	N° Broche	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 ultrasons / Détection d'obstacles à moins de 5 cm (OPTIONS)
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

** Options du Loupiot.*

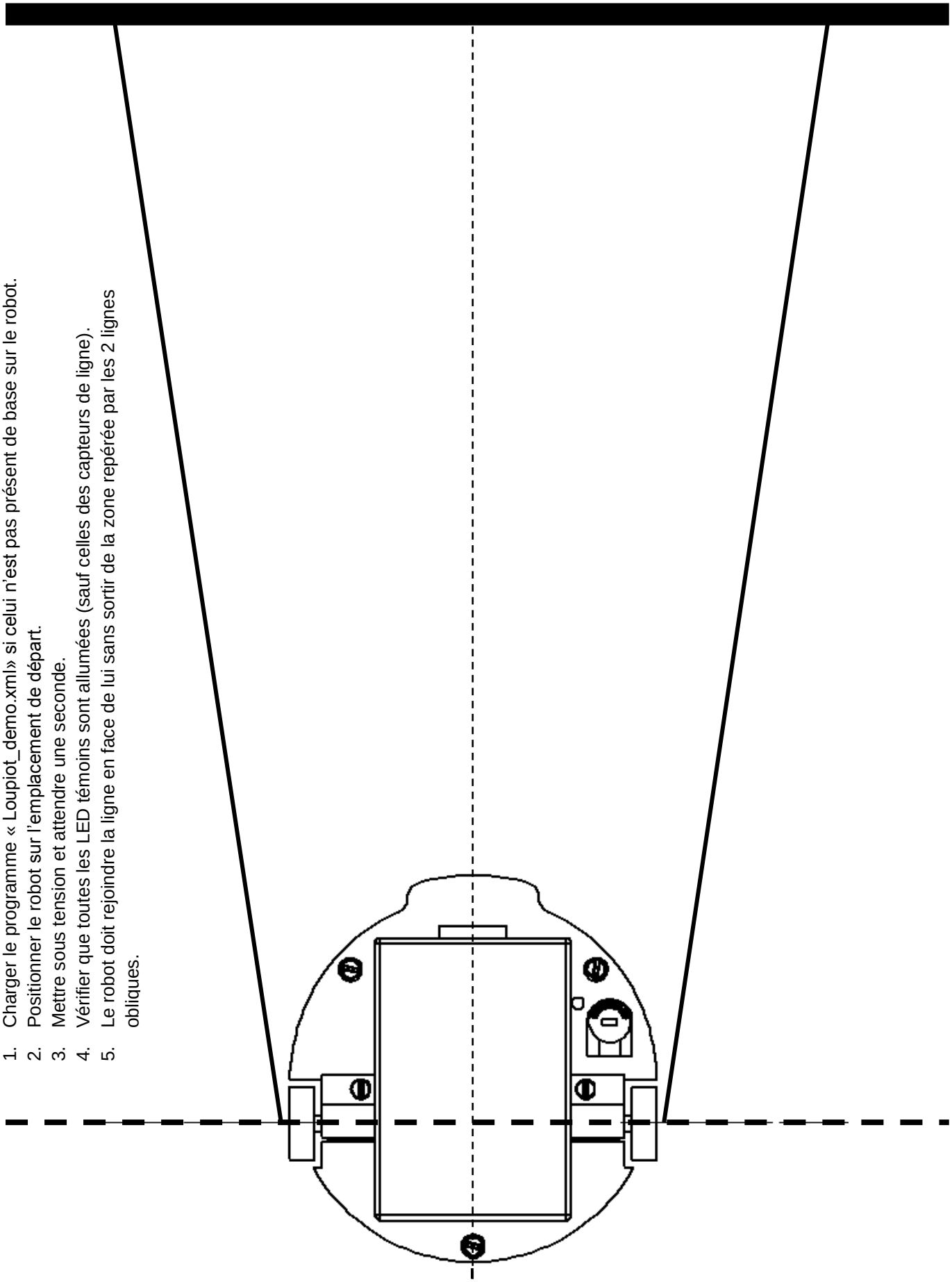
La table suivante donne le détail des sorties utilisées pour piloter les moteurs et gérer leur vitesse à l'aide d'instructions de base.

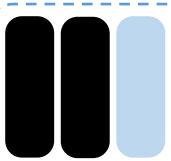
Table de vérité - Contrôle des moteurs						
	Moteur Gauche			**Moteur Droit**		
Broches Utilisées	B.2	B.3	B.1	C.1	C.2	C.3
Marche Avant	Activé	Désactivé	Vitesse PWM	Activé	Désactivé	Vitesse PWM
Marche Arrière	Désactivé	Activé		Désactivé	Activé	
Arrêt Moteur	Activé	Activé		Activé	Activé	

Piste de test

Piste de démonstration

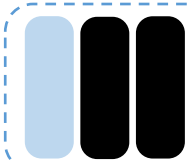
1. Charger le programme « Loupiot_demo.xml» si celui n'est pas présent de base sur le robot.
2. Positionner le robot sur l'emplacement de départ.
3. Mettre sous tension et attendre une seconde.
4. Vérifier que toutes les LED témoins sont allumées (sauf celles des capteurs de ligne).
5. Le robot doit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



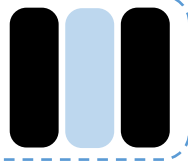
Animation lumineuse (version de base)
Séquence de clignotement des LED avec une musique

Piste de démonstration (6 programmes)

Pour robot Loupiot

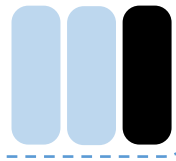
(Notice d'utilisation au verso)

Programmes disponibles sur http://a4.fr/wiki/index.php/Loupiot



Bluetooth (option)
Piloter le robot à partir d'un smartphone avec l'application **Télécommande.apk**
Téléchargeable sur www.a4.fr

Suivi de ligne avec télémètre à ultrasons (option)
Le robot suit la ligne jusqu'à la détection d'un obstacle



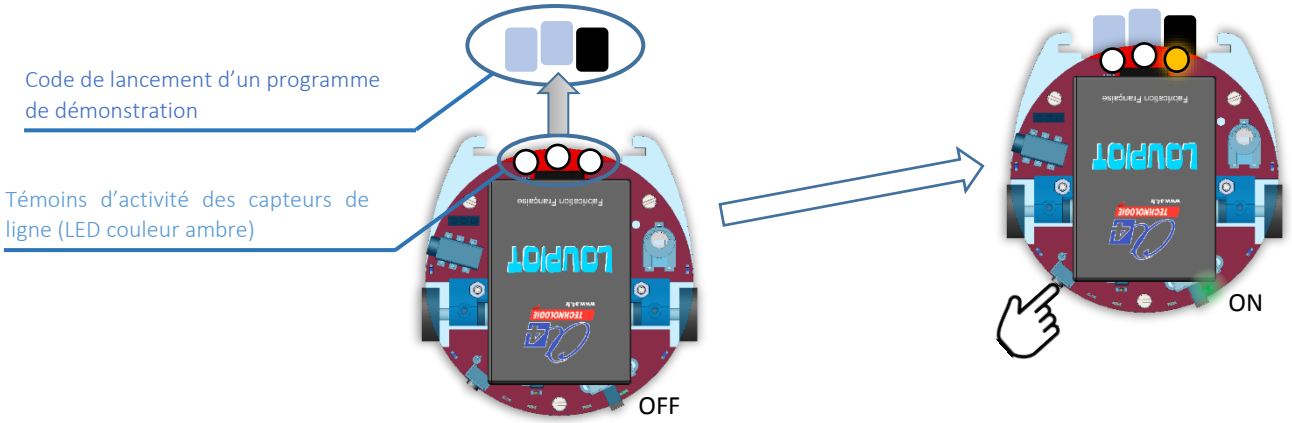
Suivi de ligne avec capteur de proximité infrarouge (option)
Le robot suit la ligne jusqu'à la détection d'un obstacle

Notice d'utilisation de la piste de démonstration

Le robot Loupiot est livré préprogrammé avec le fichier **Loupiot_demo.xml** et une piste de démonstration.
Plusieurs programmes de démonstration vous sont proposés : 3 pour la version de base du Loupiot et un pour chaque option : Bluetooth, télémètre à ultrasons, détection d'obstacles.

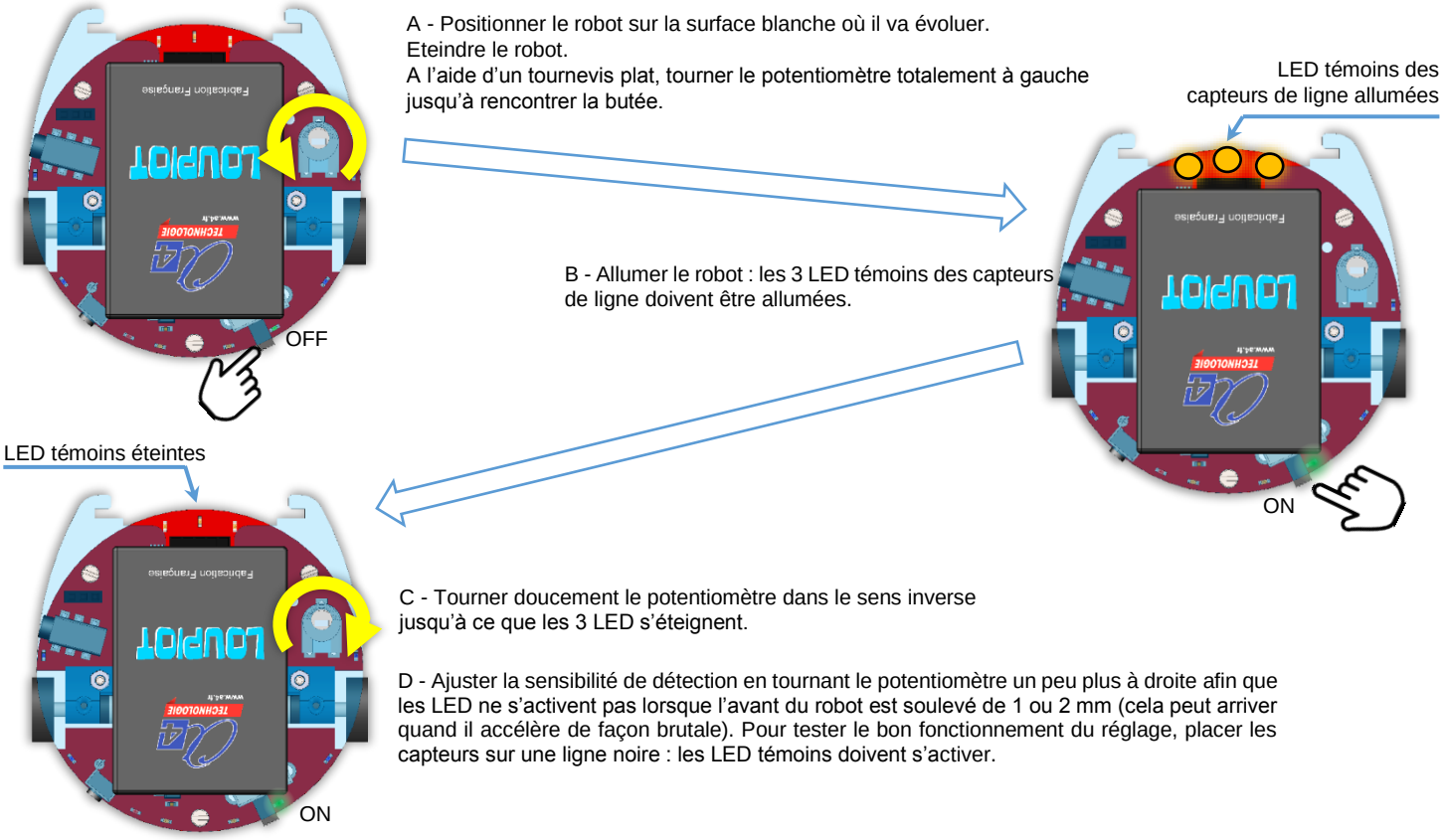
Pour lancer les programmes de démonstration des fonctionnalités du Loupiot :

1 - Mettre le robot sous tension (après avoir inséré les 3 piles AAA dans le logement prévu à cet effet).
2 - Positionner les capteurs de ligne sur un des codes.
= Les LED témoins des capteurs de ligne au-dessus de pastilles noires doivent s'allumer.
3 - Appuyer sur le bouton-poussoir pour lancer le programme de démonstration sélectionné.
4 - Le buzzer commence à sonner. Le programme de démonstration sélectionné se lance après 3 secondes.
Pour essayer un autre programme de démonstration, éteindre le Loupiot et recommencer la procédure.



Régler les capteurs de ligne

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 peut arriver que vous ayez à les re-régler. Pour cela, il suffit de suivre les étapes suivantes :



Logiciels, programmes, manuels utilisateurs téléchargeables gratuitement sur http://a4.fr/wiki/index.php/Loupiot

www.a4.fr

Concepteur et fabricant de matériels pédagogiques