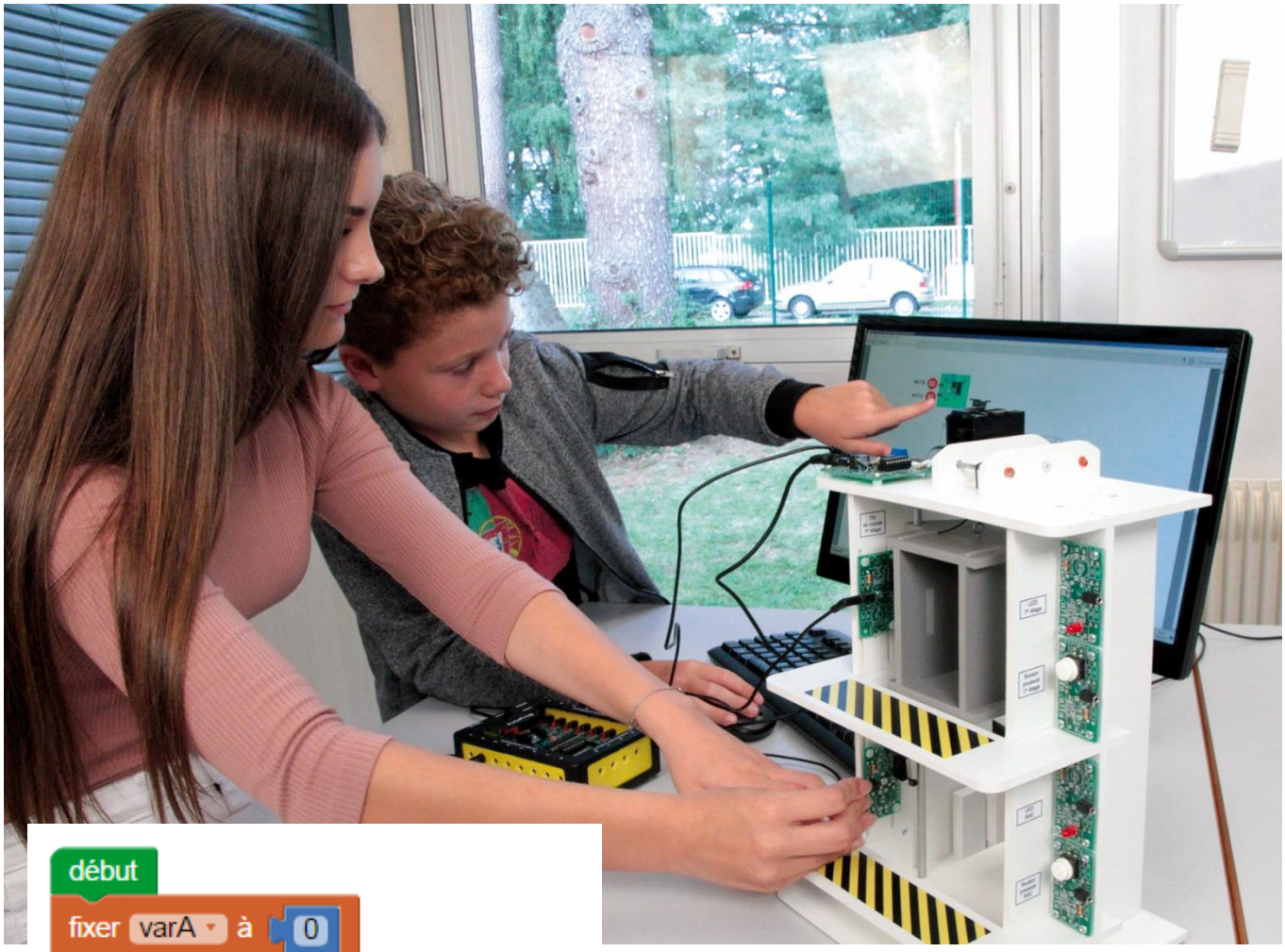
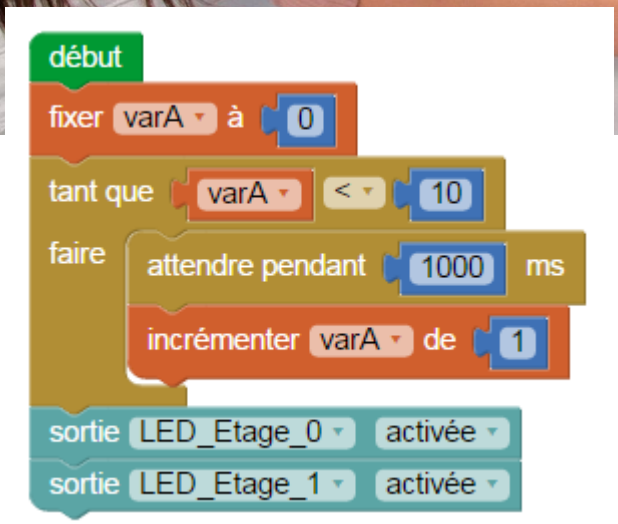
Monte-charge

Maquette programmable avec PICAXE Editor / Blockly pour PICAXE





TECHNOLOGIE

Ressources disponibles pour le projet Monte-charge

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

Monte-charge

- Fichiers **3D** (SolidWorks, Edrawings et Parasolid) de la maquette et de ses options.
- Dossier **technique** Monte-charge pour la mise en œuvre de la maquette ;
- Une notice d'utilisation de l'**option Bluetooth.**

Logiciels Picaxe Editor 6 / Blockly 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.

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

Les documents techniques et pédagogiques signés A4 Technologie sont diffusés librement sous licence Creative Commons BY-NC-SA :

– **BY** : Toujours citer A4 Technologie comme source (paternité).

– **NC** : Aucune utilisation commerciale ne peut être autorisée sans l'accord préalable de la société A4 Technologie.

– **SA** : La diffusion des documents éventuellement modifiés ou adaptés doit se faire sous le même régime.

Consulter le site ***http://creativecommons.fr/***

Note : la duplication de ce dossier est donc autorisée sans limite de quantité au sein des établissements scolaires, aux seules fins pédagogiques, à condition que soit cité le nom de l'éditeur A4 Technologie.

Logiciels, programmes, manuels utilisateurs téléchargeables gratuitement sur www.a4.fr

SOMMAIRE

Introduction

Monte-charge

La maquette Monte-charge (BE-MCHA) est une reproduction homothétique d'un monte-charge automatisé réel : plusieurs étages, capteurs fin de course, contrepoids, moteurs, etc.
Programmable et pilotée par les systèmes AutoProgX2 ou AutoProgUno, elle permet une activité de programmation complète par rapport aux attendus de fin de cycle collège : l'algorithmique en maths, l'étude de scénarios, la programmation et la mise en œuvre en Technologie.

Vous trouverez dans ce document tout le nécessaire pour démarrer des activités de programmation autour du monte-charge :

- La mise en œuvre de la maquette : câblage et configuration des modules.
- Différents scénarios de programmation, du plus simple au plus complexe, avec des exemples de programmes tout faits en langage par blocs.
- Des exercices complémentaires pour les différents modules en option : télécommande infrarouge, module Bluetooth

Les environnements de programmation graphique

Tous les programmes correspondant aux activités menées autour de la maquette AutoAlarme 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 la maquette AutoAlarme 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 de la maquette 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 de la maquette (version de base).

Niveau 3 :
Exemples d'utilisation des différentes options proposées : télécommande infrarouge et module Bluetooth.

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 :
- 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.).
- 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 :

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

Exemple : MC_N3_B2.xml
Correspond au niveau 3 avec le jeu d'instructions B, 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
- **Maquette** Monte-charge (Réf. BE-MCHA).
- **Câble de programmation** Picaxe USB (Réf : CABLE-USBPICAXE).
- **Interface programmable** AutoProgX1 ou X2 (Réf. K-APV2).
- 8 **cordons de liaison** jack compatibles AutoProg pour établir les liaisons entre l'interface programmable et la maquette.

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.

Caractéristiques techniques

Le guide de montage ainsi que les caractéristiques techniques des composants sont détaillés dans le dossier technique disponible sur le wiki.

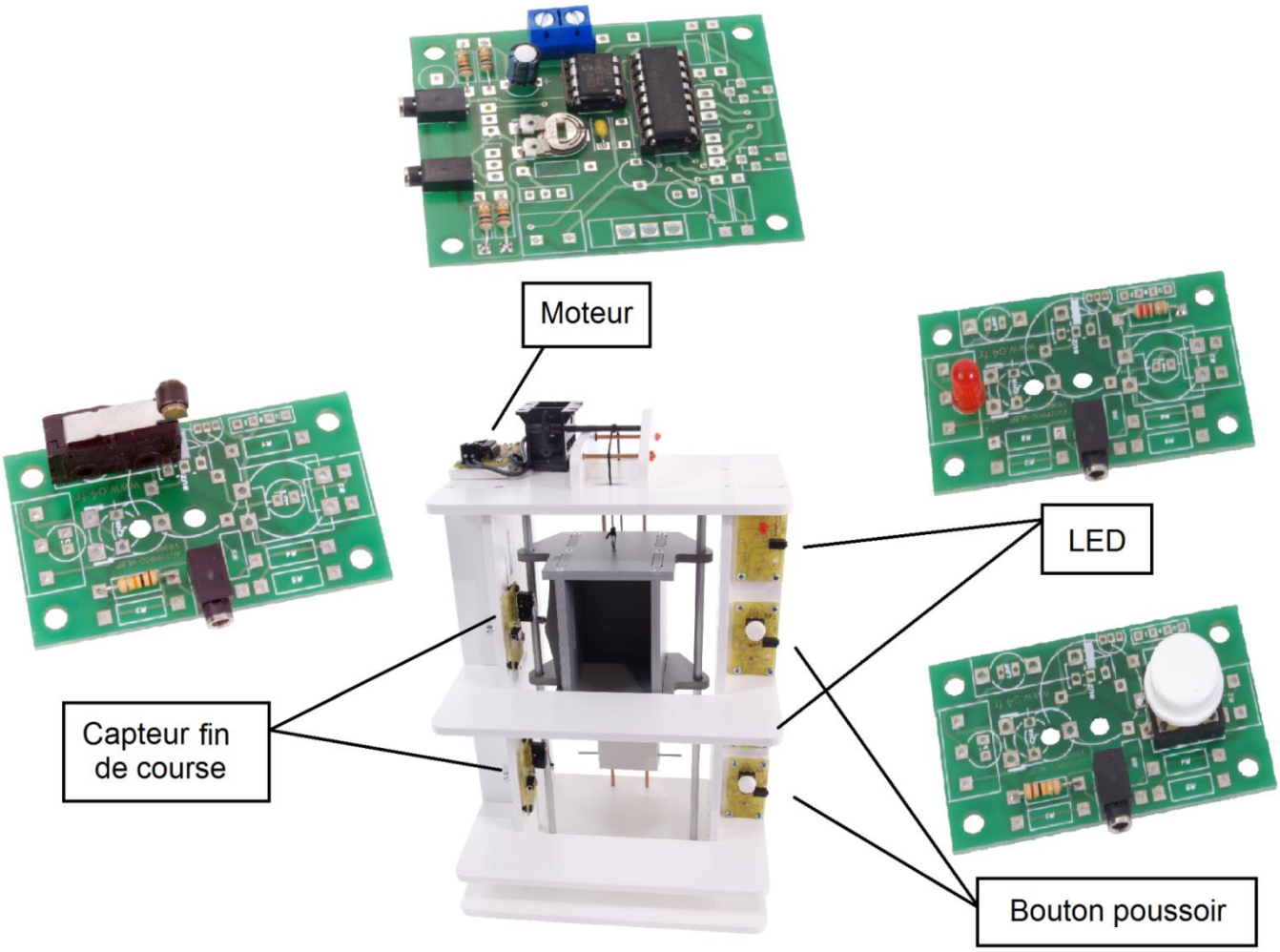
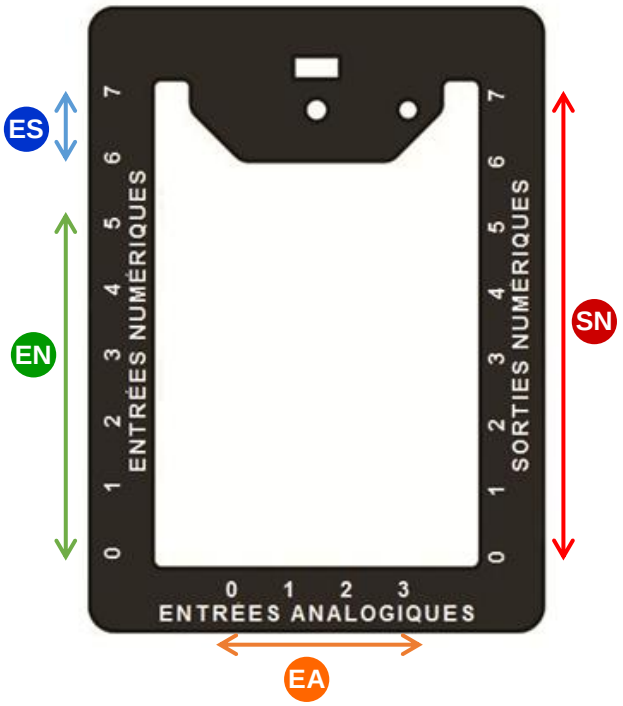


Tableau d'affectation des entrées et sorties

	Module de communication pour entrées / sorties numériques	Broche Blockly	Etiquette Blockly
7	Communication Bluetooth envoi de données	C.7	BLTH_TX
6	Communication Bluetooth réception de données	C.6	BLTH_RX
EN	**Modules capteurs pour entrées numériques**		
5	*(libre)*	C.5	
4	Capteur infrarouge pour télécommande (option)	*C.4*	Recepteur_IR*
3	Bouton poussoir 1er étage	C.3	BP_Etage_1
2	Capteur de fin de course de montée du monte-charge	C.2	FDC_Haut
1	Capteur de fin de course de descente du monte-charge	C.1	FDC_Bas
0	Bouton poussoir rez-de-chaussée	C.0	BP_Etage_0
EA	**Modules capteurs pour entrées analogiques**		
3	*(libre)*	*A.3*	
2	*(libre)*	A.2	
1	*(libre)*	A.1	
0	*(libre)*	A.0	
SN	**Modules actionneurs sorties numériques**		
7	Connecté à la broche MOTA-2 de la carte contrôle moteur	B.7	Moteur_A2
6	Connecté à la broche MOTA-1 de la carte contrôle moteur	B.6	Moteur_A1
5	*(libre)*	B.5	
4	*(libre)*	B.4	
3	*(libre)*	B.3	
2	*(libre)*	B.2	
1	Module signal LED rouge 1er étage	B.1	LED_Etage_1
0	Module signal LED rouge rez-de-chaussée	B.0	LED_Etage_0



Programmation version de base niveau 1

Objectifs :

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

Ce niveau permet de découvrir toutes les fonctionnalités de base du Monte-charge, en apprenant les structures de base de la programmation. Et en particulier celles demandées dans les nouveaux programmes : séquences, boucles, structures conditionnelles et enfin les variables.

Nous vous conseillons pour chaque exercice d'essayer d'écrire le programme vous-même, en partant du modèle de base (fourni avec les exercices), avant de regarder la correction et l'explication de chaque programme.
Par exemple, pour le programme « MC_N1_A1.xml », charger le programme modèle « MC_Base.xml ».

Dans chaque programme modèle du niveau 1, vous trouverez la liste de blocs nécessaires à la réalisation des exercices des sous niveaux A, B, C et D.
Au fur et à mesure de l'avancement dans les sous niveaux, la liste de blocs s'agrandit jusqu'à retrouver tous les blocs nécessaires pour piloter complétement la maquette.

Nom du fichier	Description	Objectif
Niveau 1 A **Fichier modèle :** MC_N1_A.xml		
MC_N1_A1	Allumer le voyant lumineux pendant 3 secondes puis l'éteindre.	Fonctionnalité matérielle abordée : -Allumage/extinction du voyant lumineux. Notions de programmation abordées : -séquence d'instructions -temps d'attente -boucle infinie
MC_N1_A2	Répéter cette même action deux fois.	
MC_N1_A3	Répéter cette action à l'infini.	
Niveau 1 B **Fichier modèle :** MC_N1_B.xml		
MC_N1_B1	Allumer un voyant lumineux lorsque la porte de devant est ouverte.	Fonctionnalité matérielle abordé : -Gestion du moteur -Utilisation de Bouton-poussoir Notions de programmation abordées : -boucle qui dépend d'une entrée
MC_N1_B2	Activer et désactiver un Buzzer toutes les 3 secondes pendant 1 seconde.	
Niveau 1 C **Fichier modèle :** MC_N1_C.xml		
MC_N1_C1	Allumer un voyant lumineux à l'appui d'un bouton-poussoir.	Fonctionnalité matérielle abordé : -Gestion des modules infra-rouge -Utilisation de Bouton-poussoir Notions de programmation abordées : -Le test d'une entrée (si/sinon)
MC_N1_C2	Activer le voyant lumineux lorsque le détecteur de fin de course est activé.	
MC_N1_C3	Contrôler le moteur avec les boutons poussoirs et allumer les LED sur le franchissement des capteurs de fin de course.	
Niveau 1 D **Fichier modèle :** MC_N1_D.xml		
MC_N1_D1	Incrémenter une variable au cours du temps et observer sa valeur à l'aide du PC (débogage).	Fonctionnalité matérielle abordé : Notions de programmation abordées : -Définition de variable -Incrémentation de variable -Test (si/sinon) de variable -Test (juste si) d'entrée -Débogage
MC_N1_D2	Incrémenter une variable au cours du temps faire un test sur celle-ci pour activer le voyant.	
MC_N1_D3	Incrémenter une variable à l'appui d'un bouton-poussoir, la décrémenter à l'appui de l'autre bouton poussoir.	
MC_N1_D4	Incrémenter une variable puis faire un test sur celle-ci pour contrôler l'état du voyant.	

Niveau 1 - A

Exercice niveau 1 - A.1 : Activer / désactiver un témoin lumineux

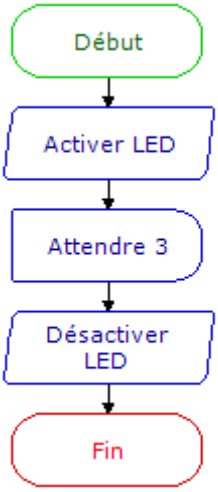
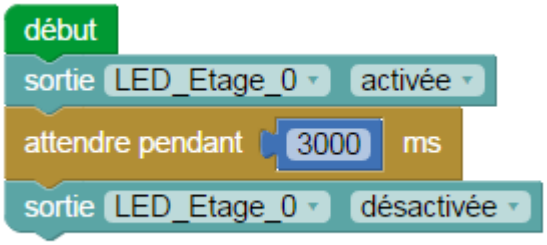
Fichier modèle : MC_N1_A.xml

Objectif : allumer une LED pendant 3 secondes puis l'éteindre.

Notions abordées : séquence d'instructions, activation / désactivation d'une sortie, temps d'attente.

Instructions utilisées :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_A1_Organigramme.plf	Fichier Blockly : MC_N1_A1.xml

Remarque : avec le langage de programmation par blocs la dernière instruction exécutée marque la fin du programme.

Exercice niveau 1 - A.2: Répéter une action deux fois

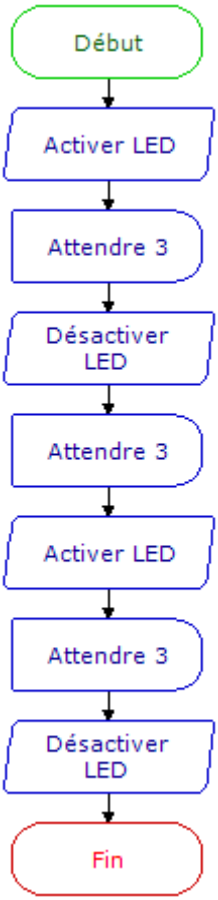
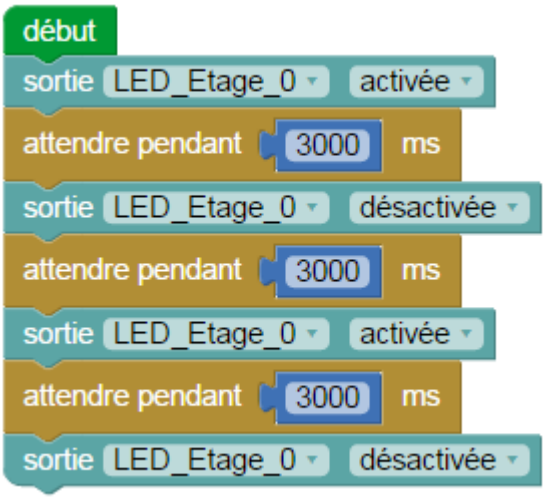
Fichier modèle : MC_N1_A.xml

Objectif : allumer le voyant lumineux pendant 3 secondes puis l'éteindre, recommencer.

Notions abordées : séquence d'instructions, activation / désactivation d'une sortie, temps d'attente.

Instructions utilisées :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_A2_Organigramme.plf	Fichier Blockly : MC_N1_A2.xml

Exercice niveau 1 - A.3 : Répéter une séquence indéfiniment

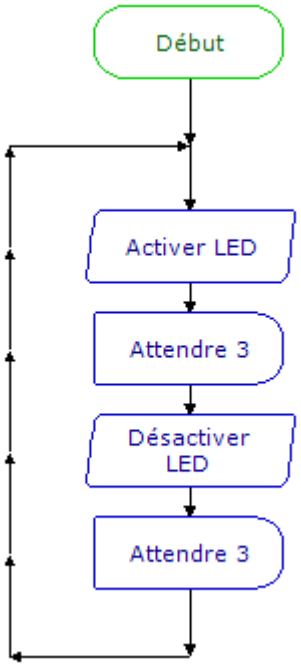
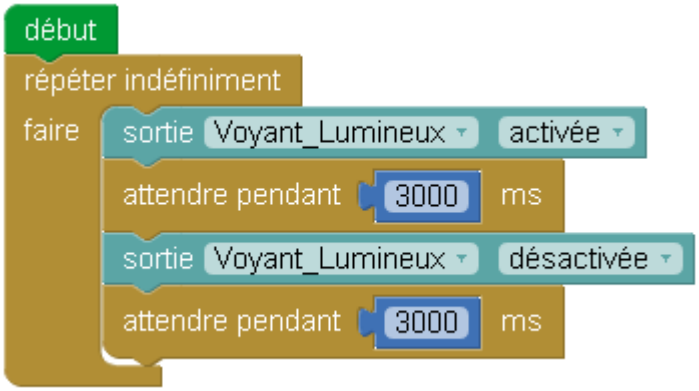
Fichier modèle : MC_N1_A.xml

Objectif : faire clignoter le voyant lumineux avec une période de 6 secondes indéfiniment.

Notion abordée : la boucle infinie.

Instructions utilisées :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_A3_Organigramme.plf	Fichier Blockly : MC_N1_A3.xml

Remarque : le programme ne peut s'arrêter lorsqu'il est dans une boucle infinie. Le seul moyen de sortir de la boucle est de faire un Reset ou d'éteindre et rallumer le boîtier AutoProg.

Sur organigramme, une boucle infinie se fait par un retour grâce aux flèches.

Niveau 1 - B

Exercice niveau 1 - B.1 : Maitriser la rotation du moteur

Fichier modèle : MC_N1_B.xml

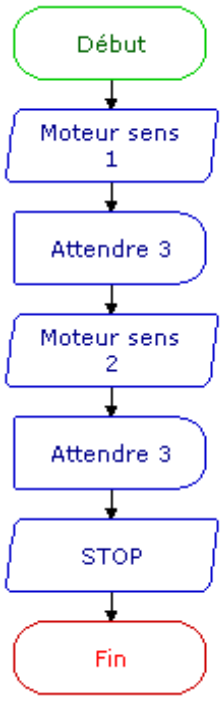
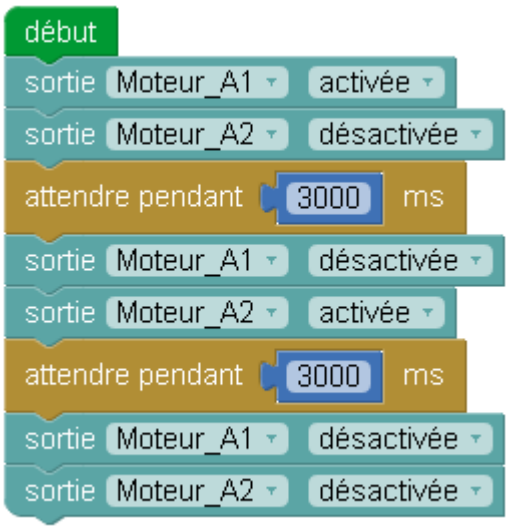
Objectif : activer un moteur dans un sens puis dans l'autre pour enfin s'arrêter.

Notion abordée : utilisation d'un moteur.

Remarque : Placer 2 charges sur les barres porte charges

Instructions utilisées :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_B1_Organigramme.plf	Fichier Blockly : MC_N1_B1.xml

ATTENTION : pour cet exercice, il est recommandé de placer le monte-charge à mi-hauteur pour éviter tout dommage.
Il faut également activer le moteur à l'aide de l'interrupteur (Une LED rouge indique si le moteur est allumé).

Pour l'organigramme utilisez la commande « Sorties » et non pas « Moteur ».

Exercice niveau 1 - B.2 : Utilisation d'une boucle tant que

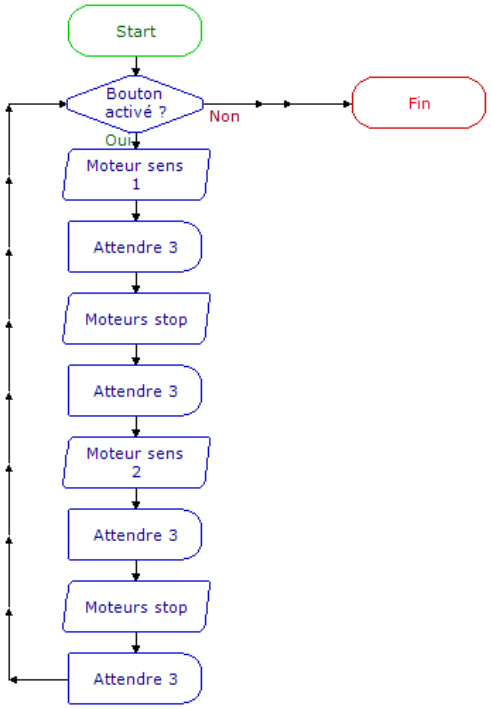
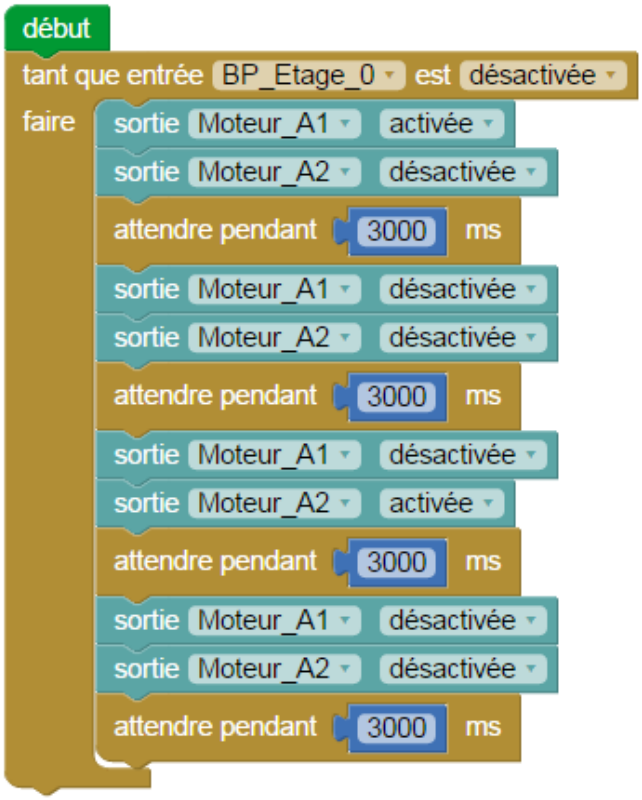
Fichier modèle : MC_N1_B.xml

Objectif : monter et descendre le monte-charge en continu jusqu'à l'appui d'un bouton-poussoir.

Notion abordée : exécuter une boucle qui dépend de l'état d'une entrée.

Instructions utilisées :

Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_B2_Organigramme.plf	Fichier Blockly : MC_N1_B2.xml

Remarque : Le programme ne peut sortir de la boucle qu'une fois le test sur le bouton-poussoir validé.
Le test sur le bouton-poussoir ne se fait qu'une seule fois en début de séquence, avant de commencer l'ouverture. Si un appui est effectué pendant la séquence, aucun effet n'aura lieu sur le programme. Afin de vérifier à tout moment le changement d'état d'une entrée dans une séquence, l'utilisation des interruptions est indispensable (voir ex sur interruption).
Pour l'organigramme, nous détournons la boucle tant que en créant un programme ayant les mêmes effets.

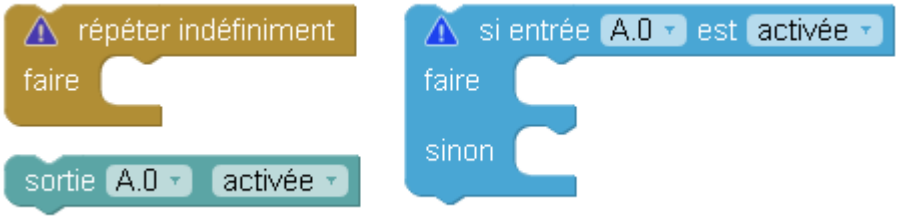
Exercice niveau 1 - C.1 : Instruction conditionnelle et bouton-poussoir

Fichier modèle : MC_N1_C.xml

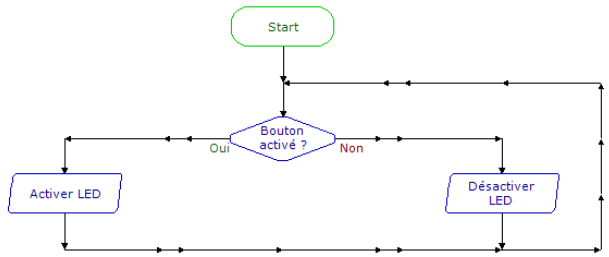
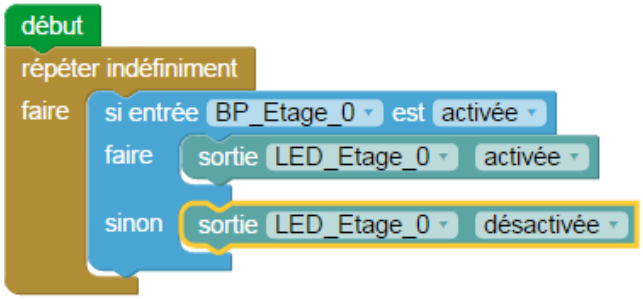
Objectif : allumer un voyant lumineux à l'appui d'un bouton poussoir.

Notion abordée : utilisation des commandes conditionnelles (si/sinon).

Instructions utilisées :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_C1_Organigramme.plf	Fichier Blockly : MC_N1_C1.xml

Remarque : les blocs de couleur bleu claires représentent des commandes concernant l'utilisation des entrées.

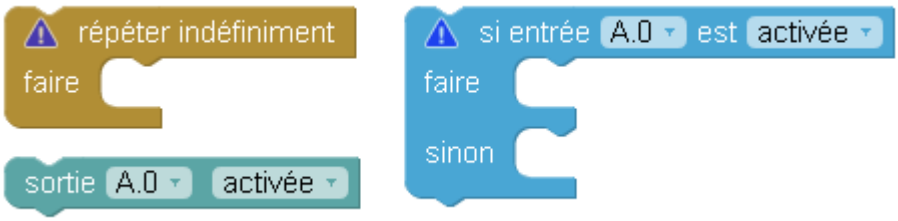
Exercice niveau 1 - C.2 : Instruction conditionnelle et capteur de fin de course

Fichier modèle : MC_N1_C.xml

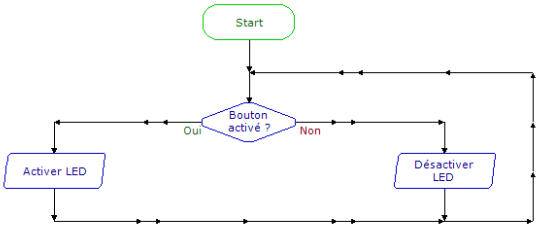
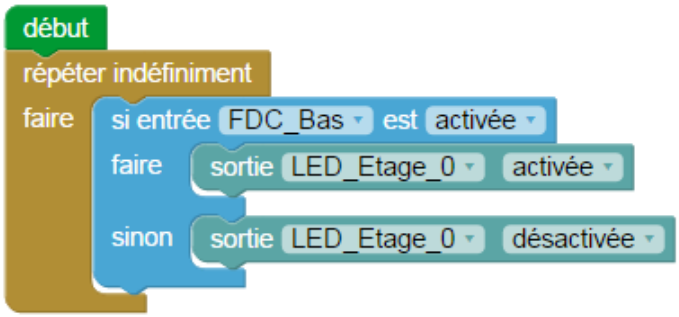
Objectif : activer le voyant lumineux lorsque le détecteur de fin de course est activé

Notions abordées : utilisation des commandes conditionnelles (si/sinon) / utilisation d'un capteur fin de course.

Instructions utilisées :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_C2_Organigramme.plf	Fichier Blockly : MC_N1_C2.xml

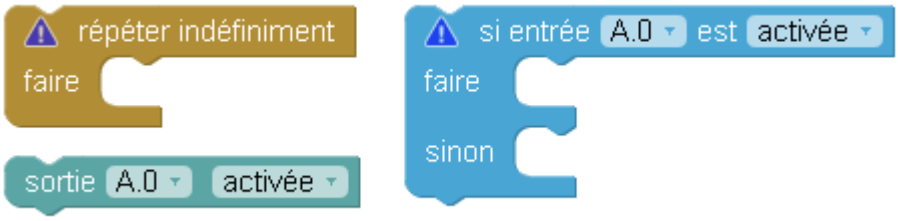
Exercice niveau 1 - C.3 : Contrôle moteur ET voyant lumineux

Fichier modèle : MC_N1_C.xml

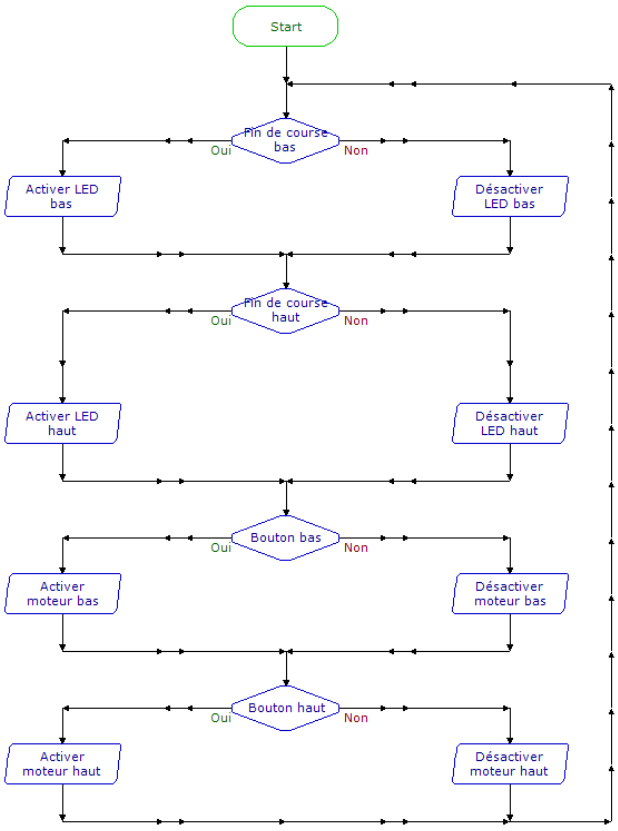
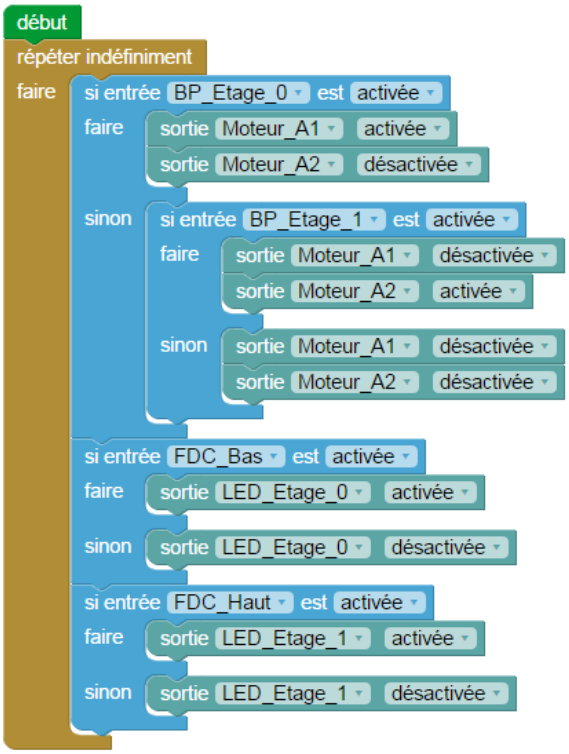
Objectif : contrôler le moteur avec les boutons poussoirs et allumer les LED sur le franchissement des capteurs de fin de course

Notion abordée : utilisation des commandes conditionnelles.

Instructions utilisées :



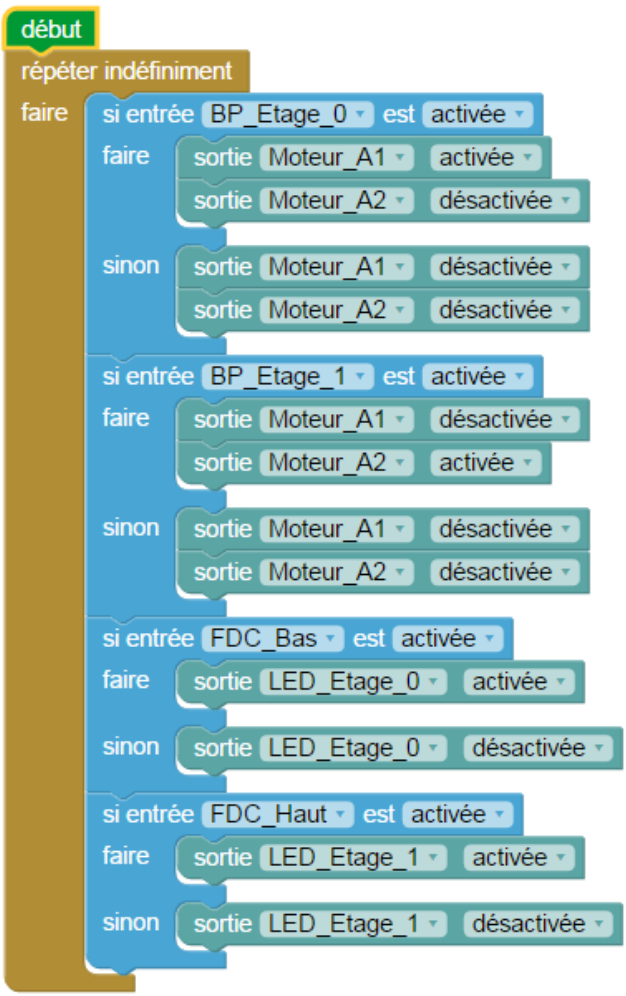
Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_C3_Organigramme.plf	Fichier Blockly : MC_N1_C3.xml

Remarque : ne pas surcharger le programme de conditions « si ».
Le programme cherchera à vérifier toutes les conditions une à une et une condition pourrait en annuler une autre.

Le programme ne permettra pas deux montées successives.

Exemple à ne pas faire :



Explication : Si on active un moteur avec ce programme, il sera automatiquement désactivé suite à une autre condition prenant en charge ce moteur.

Niveau 1 - D

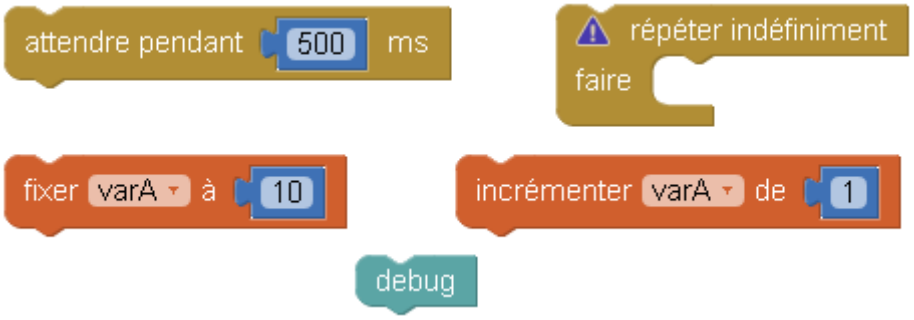
Exercice niveau 1 - D.1 : Utilisation des variables

Fichier modèle : MC_N1_D.xml

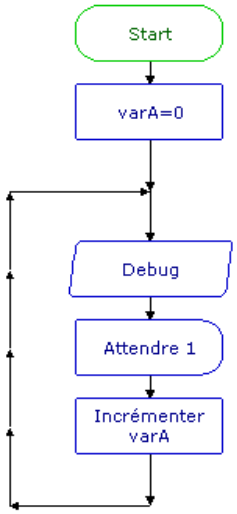
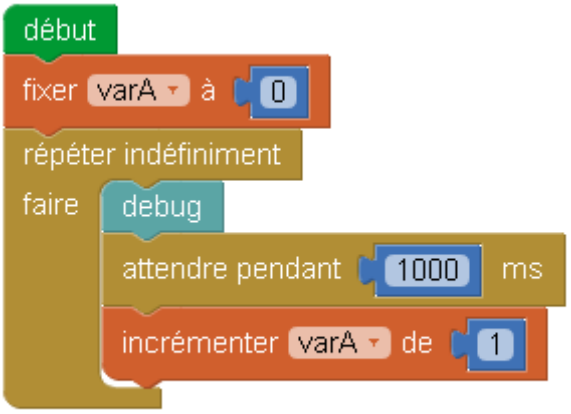
Objectif : incrémenter une variable au cours du temps et observer sa valeur à l'aide du PC (débogage).

Notions abordées : la variable : définition et incrémentation, debug.

Instructions utilisées :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_D1_Organigramme.plf	Fichier Blockly : MC_N1_D1.xml

Remarques : la commande « debug » est utilisée afin de retourner la valeur des variables à l'ordinateur. Il est donc indispensable de brancher le câble de programmation à l'ordinateur pour avoir un aperçu de leur valeur.

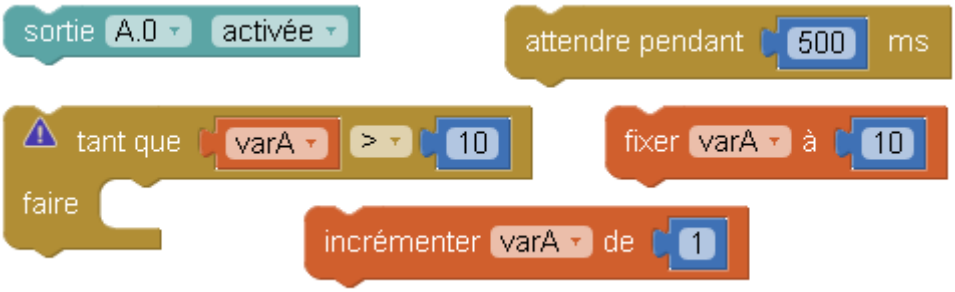
Exercice niveau 1 - D.2 : Utiliser et tester une variable

Fichier modèle : MC_N1_D.xml

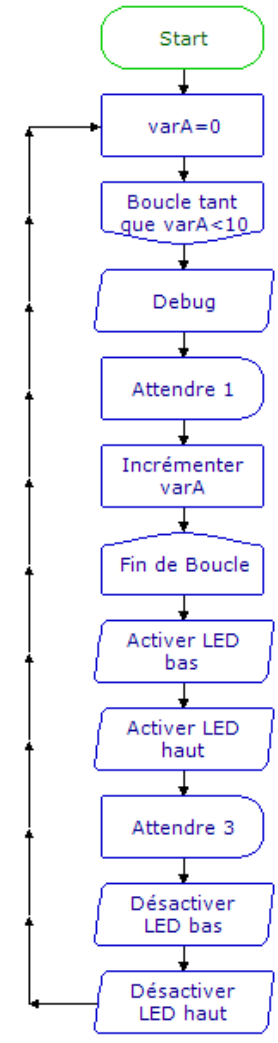
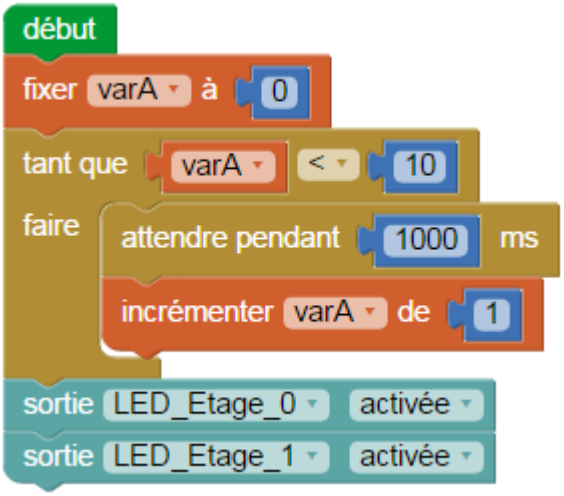
Objectif : incrémenter une variable au cours du temps. Lorsque la variable est supérieure à 10, activer les LED

Notion abordée : boucle « tant que » dépendant d'une variable.

Instructions utilisées :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_D2_Organigramme.plf	Fichier Blockly : MC_N1_D2.xml

Remarque : cet exercice peut être utilisé comme un minuteur.

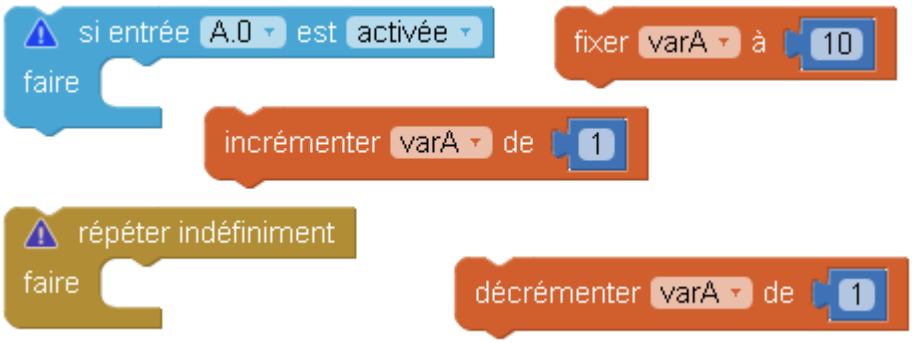
Exercice niveau 1 - D.3 : Contrôler la valeur d'une variable à l'aide des boutons-poussoirs

Fichier modèle : MC_N1_D.xml

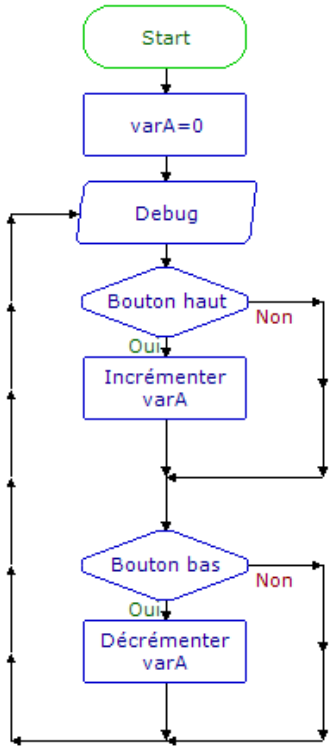
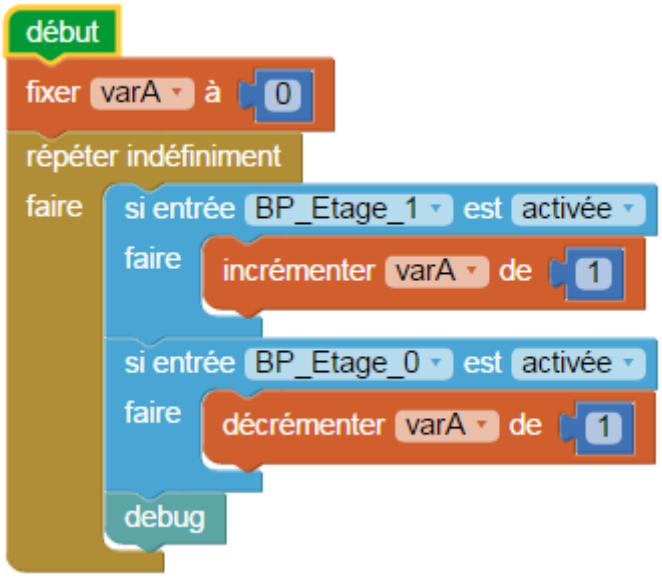
Objectif : incrémenter une variable à l'appui d'un bouton poussoir, décrémenter la même variable à l'appui de l'autre bouton poussoir.

Notions abordées : test sur entrées et incrémentation/décrémentation contrôlée d'une variable.

Instruction utilisée :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_D3_Organigramme.plf	Fichier Blockly : MC_N1_D3.xml

Remarques : deux tests sont insérés l'un après l'autre. La vitesse d'exécution du programme donne l'impression que les commandes sont exécutées en même temps.
La commande « debug » est utilisée afin de retourner la valeur des variables à l'ordinateur. Il est donc indispensable de brancher le câble de programmation à l'ordinateur pour avoir un aperçu de leur valeur.

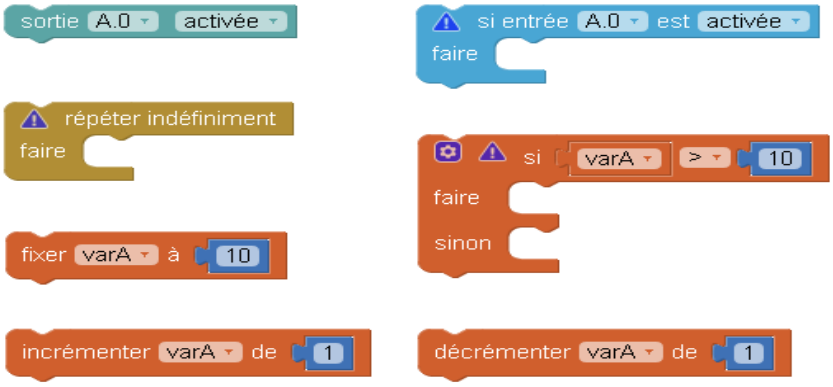
Exercice niveau 1 - D.4 : Tests /variables

Fichier modèle : MC_N1_D.xml

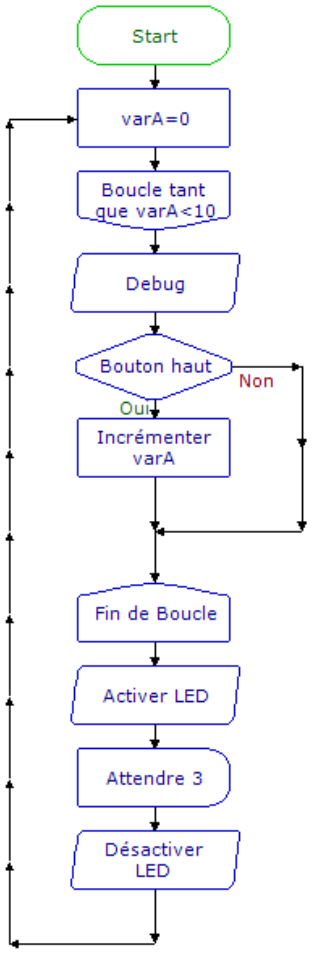
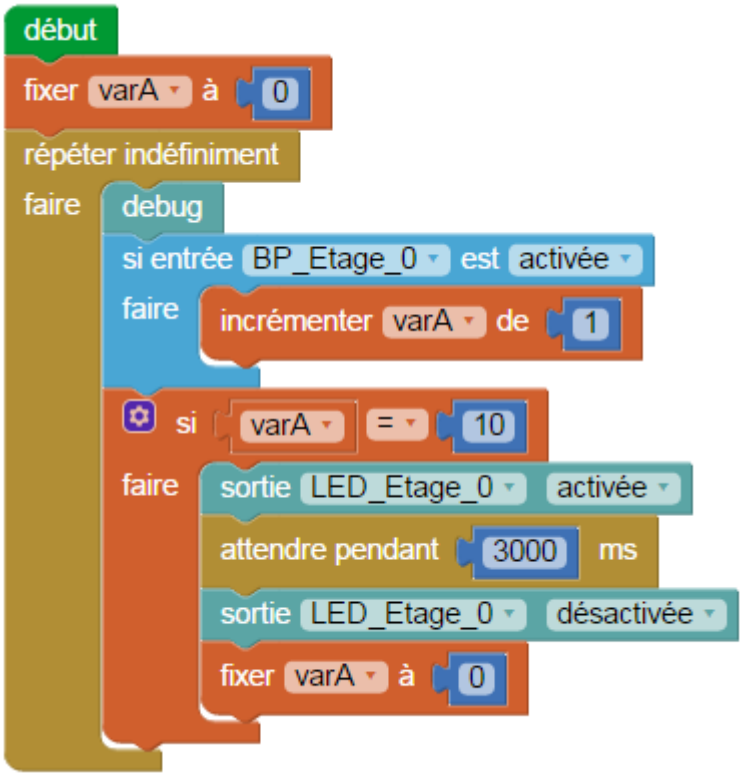
Objectif : incrémenter une variable à chaque appui d'un bouton-poussoir. Lorsque le compteur arrive à 10, activer une LED 3 secondes et remettre la variable à zéro

Notion abordée : test dépendant d'une variable

Instructions utilisées :



Correction :

Organigramme	Blocs
	
Fichier organigramme PE6 : MC_N1_D4_Organigramme.plf	Fichier Blockly : MC_N1_D4.xml

Programmation version de base niveau 2

Objectifs :

- Utilisation concrète du monte-charge
- Utilisation de tous les modules de la maquette
- Appréhension des différentes fonctionnalités du matériel ainsi que certaines notions de sécurité.

Ce niveau permet de mettre en œuvre le monte-charge, au fur et à mesures des exercices vous allez utiliser de plus en plus de modules et enrichir votre code pour obtenir à la fin du niveau un monte-charge qui marche parfaitement et qui respecte une logique de fonctionnement calquée sur le réel.

Nom du fichier	Description	Objectif
Niveau 2 A **Fichier modèle :** MC_N2_A.xml		
MC_N2_A1	Monter et descendre le monte-charge avec 2 secondes d'attente entre chaque mouvement. Utiliser les capteurs fins de course pour contrôler l'ouverture et la fermeture.	Fonctionnalité matérielle abordée : Utilisation des FDC.
MC_N2_A2	Montée du monte-charge à l'appui sur le bouton-poussoir d'en haut. Descente du monte-charge à l'appui sur bouton-poussoir d'en bas.	
MC_N2_A3	Monter et descendre le monte-charge à l'aide des BP sans distinction. Faire en sorte que les LED clignotent lors d'une manœuvre de la barrière.	
MC_N2_A4	Ouvrir et fermer le monte-charge à l'aide des BP sans distinction. Les LED doivent clignoter lors d'une manœuvre du monte-charge.	

Niveau 2 - A

Exercice niveau 2 - A.1 : ouverture/fermeture entre fins de course

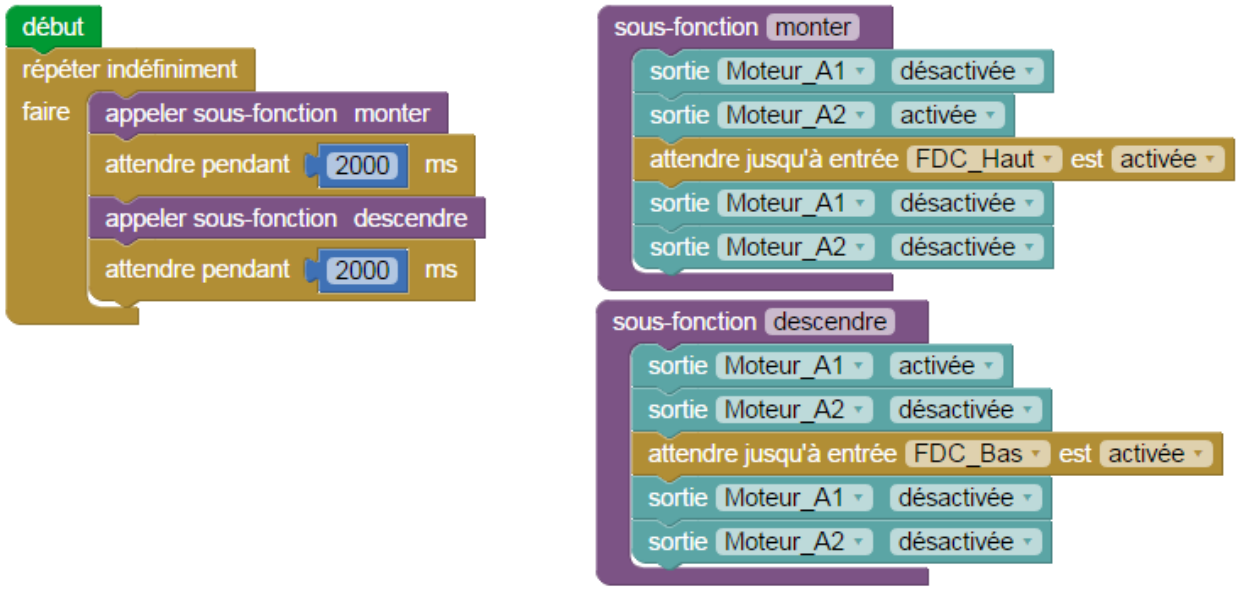
Objectif : monter et descendre le monte-charge avec 2 secondes d'attente entre chaque mouvement.
Utiliser les capteurs fins de course pour contrôler l'ouverture et la fermeture.

Notions abordées : utilisation des fins de course, procédures (sous-fonctions)

Réf. K-AP-MMR

Correction :

Blocs



Fichier Blockly : MC_N2_A1.xml

Remarque : l'utilisation des sous-fonctions « monter » et « descendre » facilite la lecture du programme.

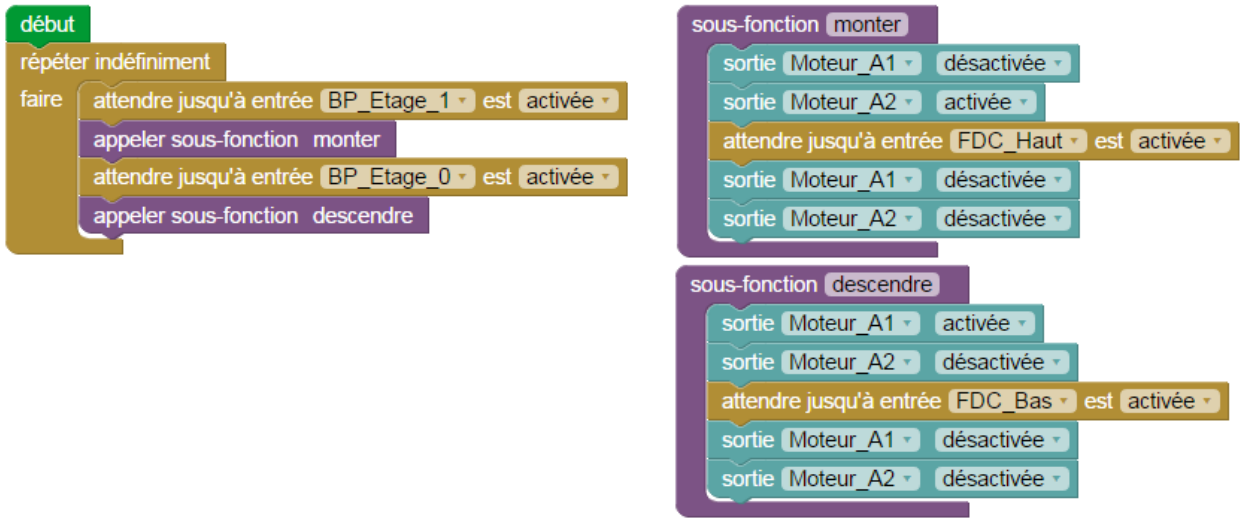
Exercice niveau 2 - A.2 : Contrôle de l'ouverture et de la fermeture

Objectif : montée du monte-charge à l'appui du bouton étage 1. Descente du monte-charge à l'appui du bouton étage 0.

Notions abordées : réutilisation des sous-fonctions créées pour un autre programme.

Correction :

Blocs



Fichier Blockly : MC_N2_A2.xml

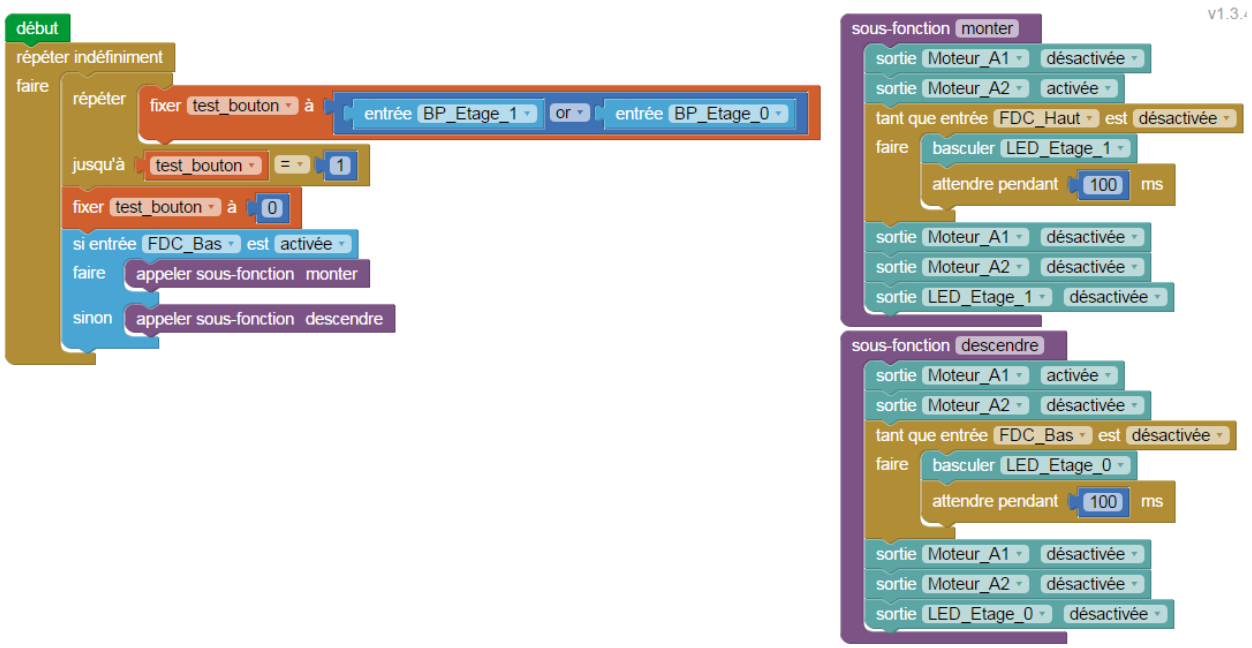
Exercice niveau 2 - A.3 : Contrôle ouverture/fermeture avec BP et signal d'arrivée

Objectif : Faire monter et descendre le monte-charge à l'aide des boutons-poussoirs sans distinction, faire en sorte qu'une LED clignote lors d'une manœuvre de la barrière.

Notions abordées : utilisation d'opérateur logique OU (+)

Correction :

Blocs



Fichier Blockly : MC_N2_A3.xml

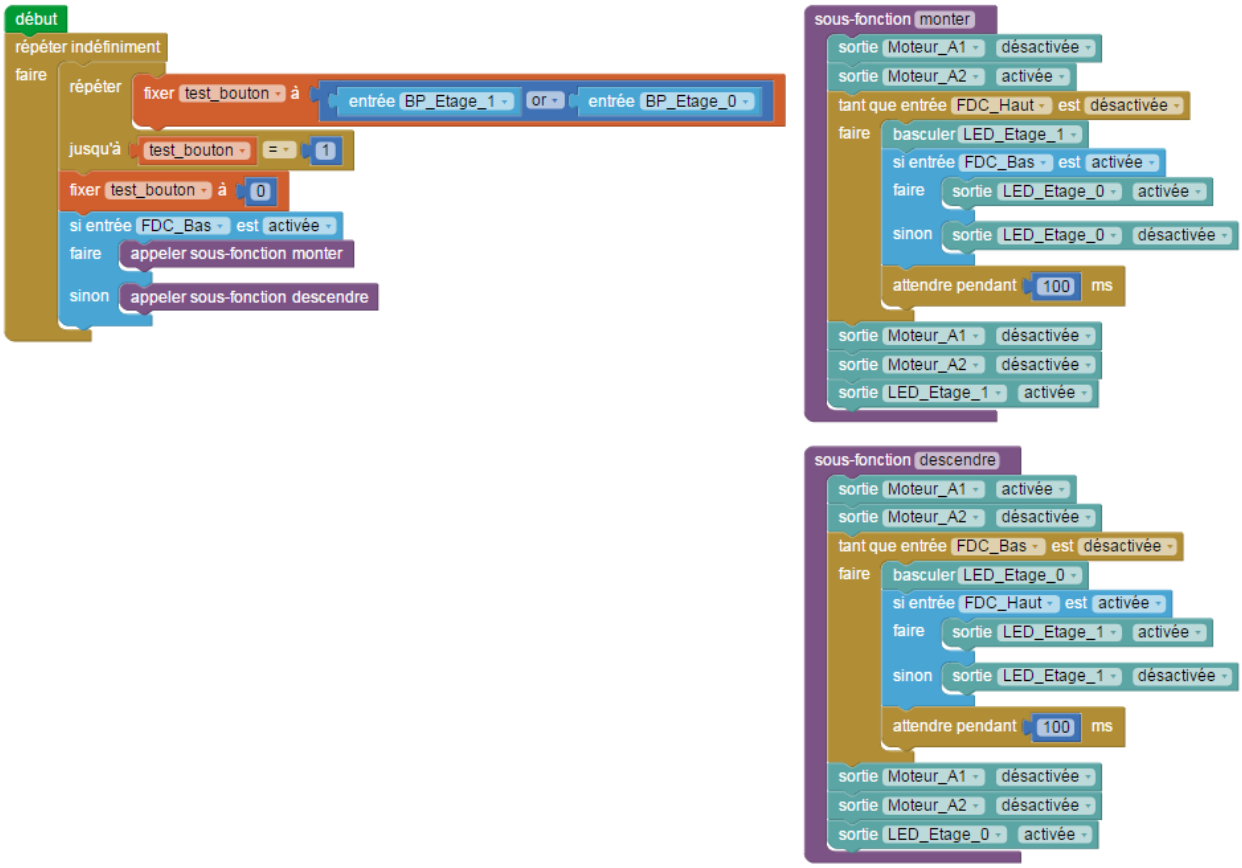
Exercice niveau 2 - A.4 : Contrôle des LED lors d'une entrée dans une nouvelle boucle

Objectif : Reprendre l'exercice précédent, une LED doit rester allumée à l'étage où se trouve le monte-charge

Remarque : Utiliser les capteurs fin de course et des conditions

Correction :

Blocs



Fichier Blockly : MC_N2_A4.xml

Programmation niveau 3

Objectif :

- Utiliser les modules plus complexes : pilotage à distance, en Bluetooth, par infrarouge…

Le niveau 3 n'intègre pas de nouvelles notions de programmation mais de nouveaux blocs permettant d'utiliser les modules options.

Nom du fichier	Description	Objectif
Niveau 3 A – Module Bluetooth **Fichier modèle :** MC_N3_A.xml		
MC_N3_A1	Contrôler la montée et la descente du monte-charge à l'aide de 2 boutons présent sur l'application Android.	Fonctionnalité matérielle abordé : - module Bluetooth Notions de programmation abordées : liaison série (hserin/hserout)
MC_N3_A2	Monter ou descendre à partir d'un seul bouton disponible sur l'application Android.	
MC_N3_A3	Jouer une sonnerie sur le Smartphone à partir de l'appui d'un BP du monte-charge.	
MC_N3_A4	Gérer la sonnette ainsi que le contrôle du monte-charge à distance à l'aide de l'application Android.	
Niveau 3 B – Télécommande infrarouge **Fichier modèle :** MC_N1_B.xml		
MC_N3_B1	Monter et descendre le monte-charge à l'aide de la télécommande IR.	Fonctionnalité matérielle abordé : Utilisation de la télécommande IR Notions de programmation abordées : Utilisation d'un block dédié à la communication IR.
MC_N3_B2	Mettre le monte-charge en mouvement lorsque le bon code est envoyé par une télécommande IR.	

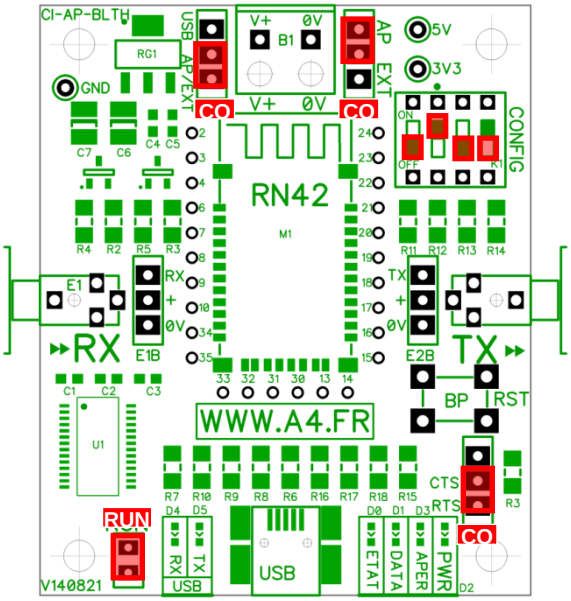
Option : Module Bluetooth

Le module Bluetooth développé par A4 Technologie permet de convertir le protocole Bluetooth en protocole de communication type Série qui est le mode de communication classique utilisé avec PICAXE ou Arduino. Ce module accepte différentes configurations.
En mode avancé, il peut être configuré au travers d'une liaison par connexion USB à un PC ou par l'envoi de commandes au travers de ses liaisons RX et TX.

La documentation technique du module Bluetooth décrit en détail les fonctionnalités du module. Elle est téléchargeable sur http://a4.fr/wiki/index.php/Module_Bluetooth_-_K-AP-MBLTH_/_S-113020008 .
Les informations seront envoyées via un smartphone ou une tablette possédant la technologie Bluetooth à l'aide d'une application développée sous AppInventor par l'équipe technique de A4.

Configuration

Positionner les cavaliers et interrupteurs comme indiqué par les positions repérées en rouge ci-dessous.



Le cavalier repéré **RUN** est utilisé lors de la mise au point de programmes avec **Arduino**. Il doit être ôté pour permettre le téléversement du programme puis doit être remis lors de l'utilisation.
La mise au point de programmes avec **PICAXE** ne nécessite pas d'ôter ce cavalier pour transférer le programme.

Les cavaliers **CO1** et **CO2** permettent de sélectionner le mode d'alimentation du module Bluetooth. Dans la configuration ci-dessus, son alimentation provient directement de l'interface AutoProg ou AutoProgUno au travers des cordons de liaison avec le module ; ils sont positionnés respectivement sur AP et sur AP/EXT.

Le cavalier **CO3** est utilisé en mode avancé pour relier ou dissocier les signaux CTS et RTS nécessaires au fonctionnement du module Bluetooth. Ici, il est positionné sur CTS/RTS.

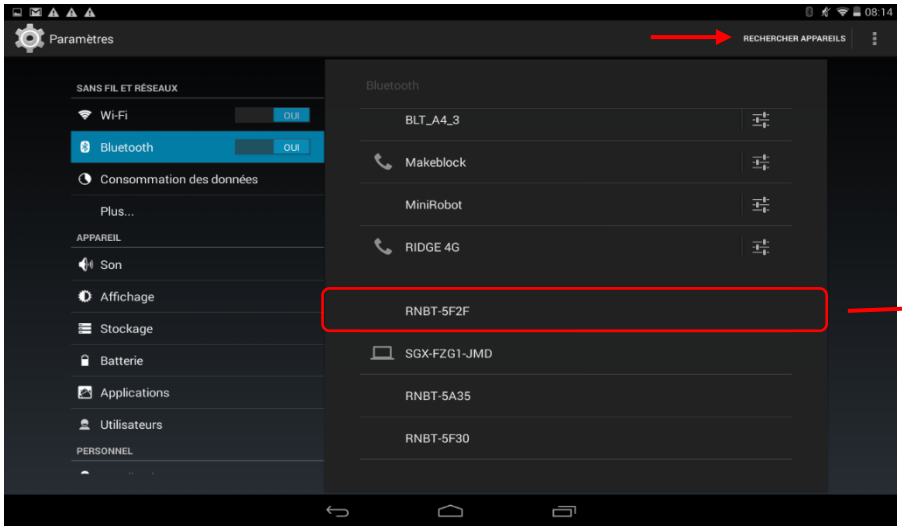
Les interrupteurs **CONFIG** permettent de paramétrer le mode de fonctionnement du module Bluetooth. Ici, l'interrupteur n°2 est positionné sur ON pour sélectionner une vitesse de transmission des données à 9600 bauds.

Témoins lumineux

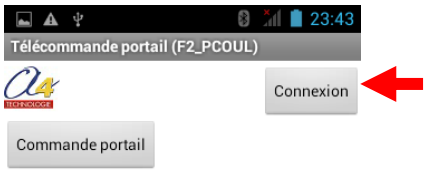
PWR indique que le module est sous tension.
APER indique que le module est associé avec un matériel Bluetooth.
DATA indique qu'il y a un flux de données entre le module et l'appareil avec lequel il est connecté.
ETAT indique que le module est opérationnel. L'affichage clignotant indique qu'il n'est pas opérationnel.
USB RX indique qu'il y a un flux de données sur la liaison USB du PC vers le module.
USB TX indique qu'il y a un flux de données sur la liaison USB du module vers le PC.

Mise en place des programmes et procédure de connexion

Avant de commencer à tester les programmes il faut d'abord appairer le smartphone ou la tablette au module bluetooth.
Pour cela rendez-vous dans les réglages bluetooth et lancer une recherche d'appareils (la maquette doit être allumée pour allimenter le module). Le nom de votre module s'appelle : RNBT + les 4 derniers chiffres de l'adresse mac du module notés sur le composant. Selectionnez le et un message proposant de vous connecter à lui deARait s'afficher.



Une fois cette étape passée vous pourrez vous connecter au module à partir du programme AppInventor à chaque fois.
Lorsque la connexion est réalisée, le bouton **Déconnexion** apparaît dans l'application.
Le témoin vert **DATA** s'allume sur le module dès qu'une donnée est émise ou reçue par le module Bluetooth.
L'appui sur le bouton d'envoi de données, dans cet exemple **Commande portail**, déclenche l'allumage fugitif de ce témoin.



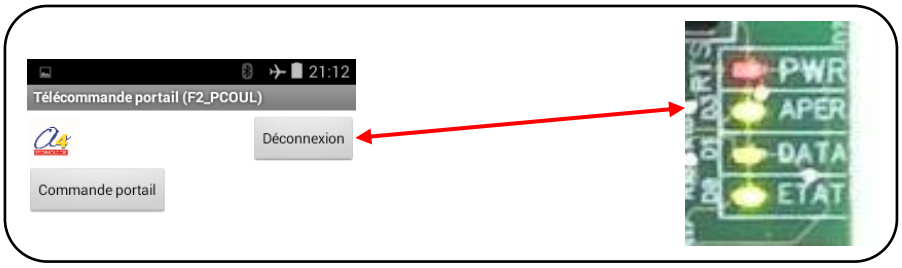
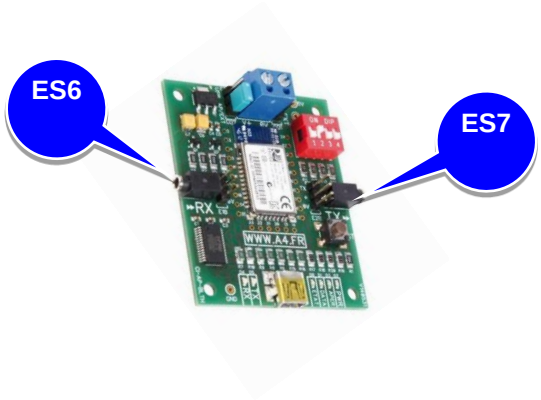


Tableau d'affectation des entrées et sorties en Bluetooth

	Modules de communication pour entrées / sorties numériques	Broche Blockly	Etiquette Blockly
7	Communication Bluetooth envoi de données	C.7	**BLTH_TX***
6	Communication Bluetooth réception de données	C.6	**BLTH_RX***
EN	**Modules capteurs pour entrées numériques**		
5	*(libre)*	C.5	
4	Récepteur infrarouge pour télécommande (option)	*C.4*	Recepteur_IR*
3	Bouton-poussoir 1er étage	C.3	BP_Etage_1
2	Capteur de fin de course de montée du monte-charge	C.2	FDC_Haut
1	Capteur de fin de course de descente du monte-charge	C.1	FDC_Bas
0	Bouton-poussoir rez-de-chaussée	C.0	BP_Etage_0
EA	**Modules capteurs pour entrées analogiques**		
3	*(libre)*	*A.3*	
2	*(libre)*	A.2	
1	*(libre)*	A.1	
0	*(libre)*	A.0	
SN	**Modules actionneurs sorties numériques**		
7	Connecté à la broche MOTA-2 de la carte contrôle moteur	B.7	Moteur_A2
6	Connecté à la broche MOTA-1 de la carte contrôle moteur	B.6	Moteur_A1
5	*(libre)*	B.5	
4	*(libre)*	B.4	
3	*(libre)*	B.3	
2	*(libre)*	B.2	
1	Module signal LED rouge 1er étage	B.1	LED_Etage_1
0	Module signal LED rouge rez-de-chaussée	B.0	LED_Etage_0

Câblage du module Bluetooth (K-AP-MBLTH)



Exercice niveau 3 - A.1 : Monter/descendre avec application Bluetooth

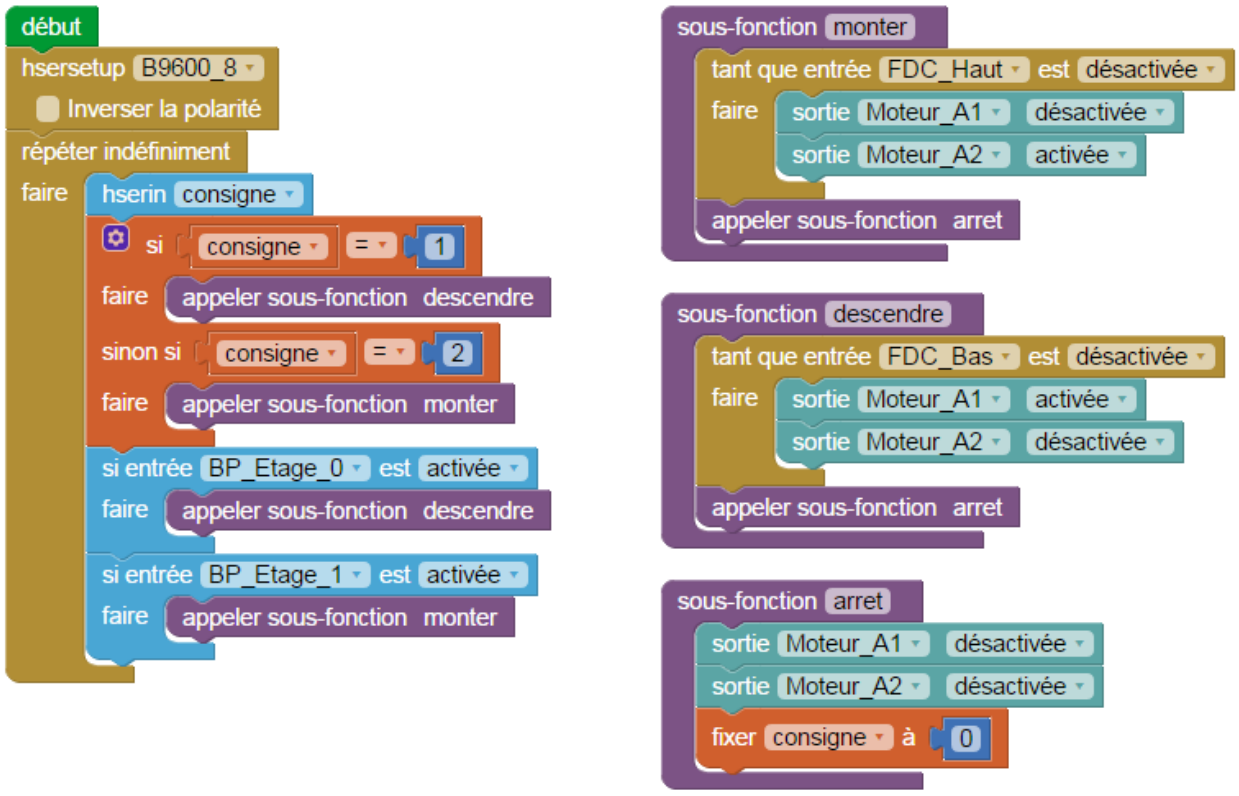
Objectif : Contrôler la descente et la montée du monte-charge à l'aide de 2 boutons présent sur l'application Android.

Notion abordée : réception de données Bluetooth envoyées par un Smartphone.

Application Android : MCharge_1.apk
Fichier App Inventor : MCharge _1.aia

Correction :

Blocs



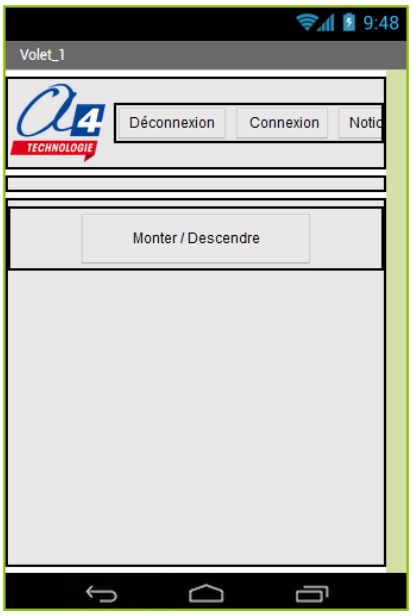
Fichier Blockly : MC_N3_A1.xml

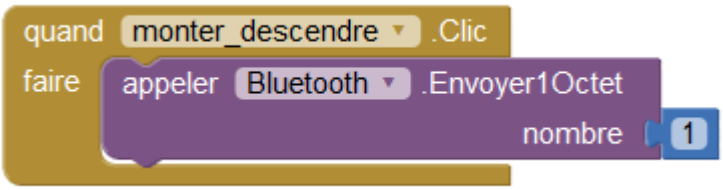
Exercice niveau 3 - A.2 : Contrôle du monte-charge par Smartphone

Objectif : Monter ou descendre le monte-charge à partir d'un seul bouton disponible sur l'application Android. La LED de destination du monte-charge doit être activée lors d'un déplacement.

Notion abordée : réception de données Bluetooth envoyées par un Smartphone.

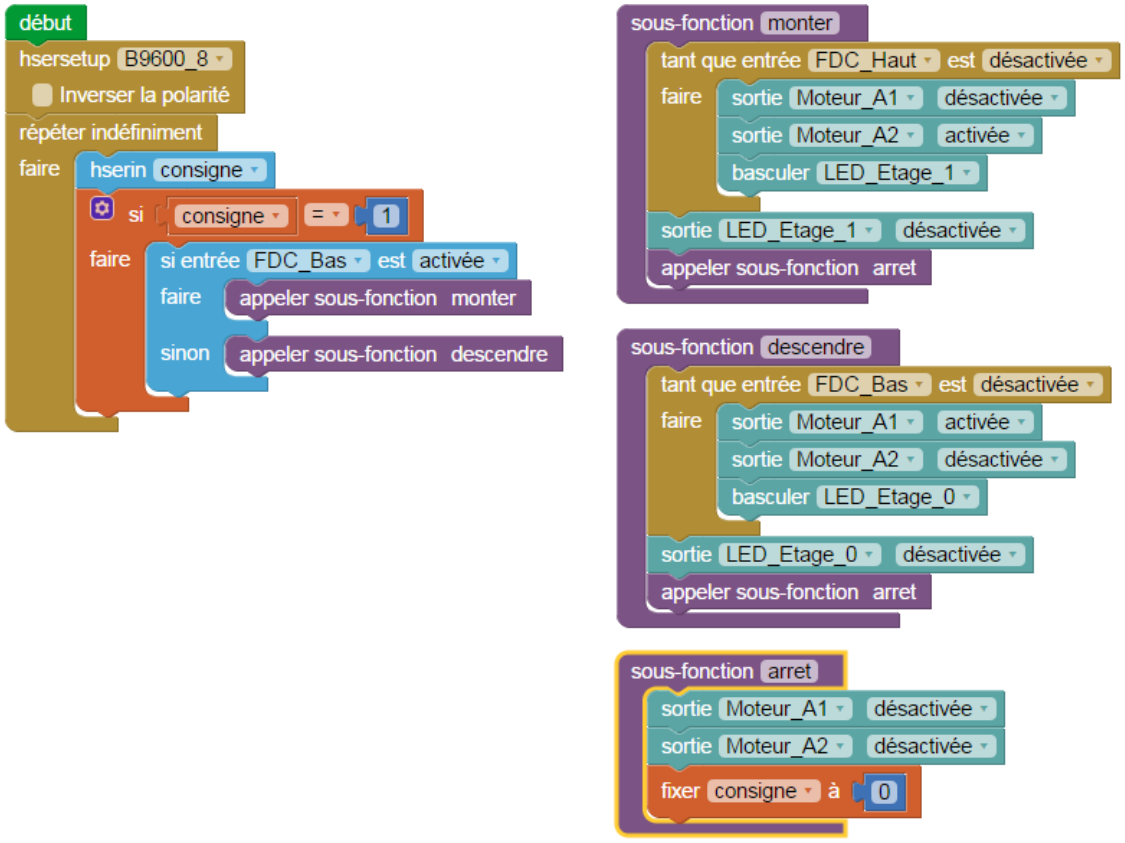
Application Android : MCharge _2.apk
Fichier App Inventor : MCharge _2.aia





Correction :

Blocs



Fichier Blockly : MC_N3_A2.xml

Exercice niveau 3 - A.3 : Envoyer des données vers un Smartphone

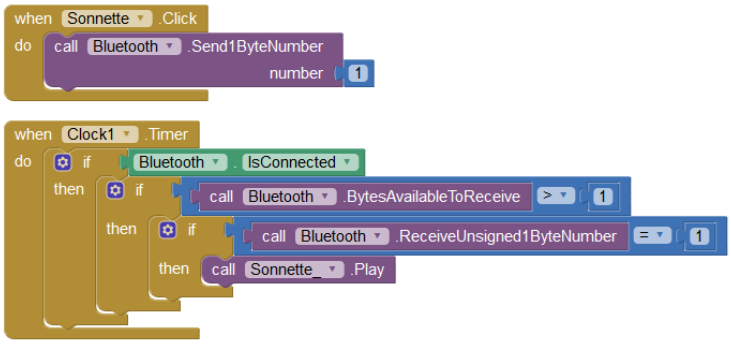
Objectif : jouer une sonnerie sur le Smartphone à partir de l'appui d'un BP du monte-charge ou sur un bouton présent sur l'application

Notion abordée : envoyer des informations à un Smartphone par Bluetooth.

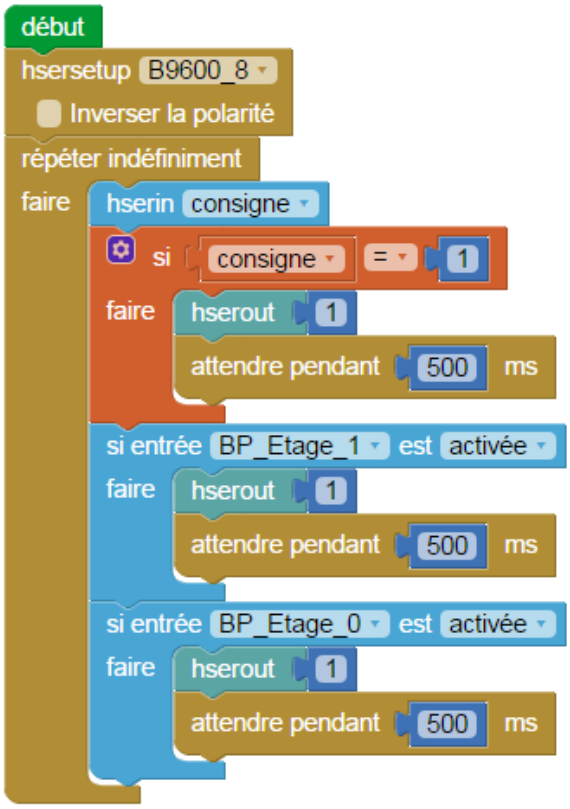
Application Android : MCharge_3.apk

Fichier App Inventor : MCharge_3.aia

Correction :

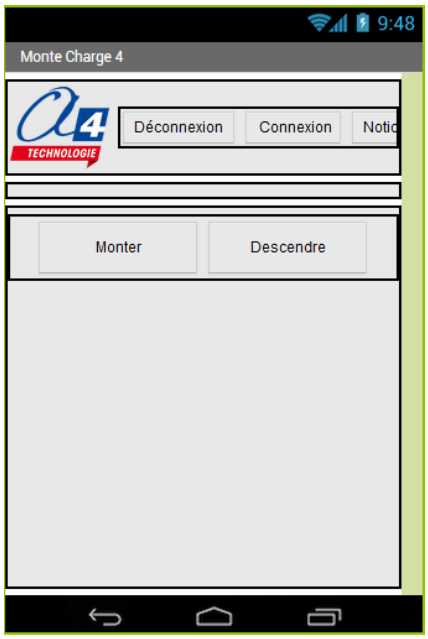


Blocs



Fichier Blockly : MC_N3_A3.xml

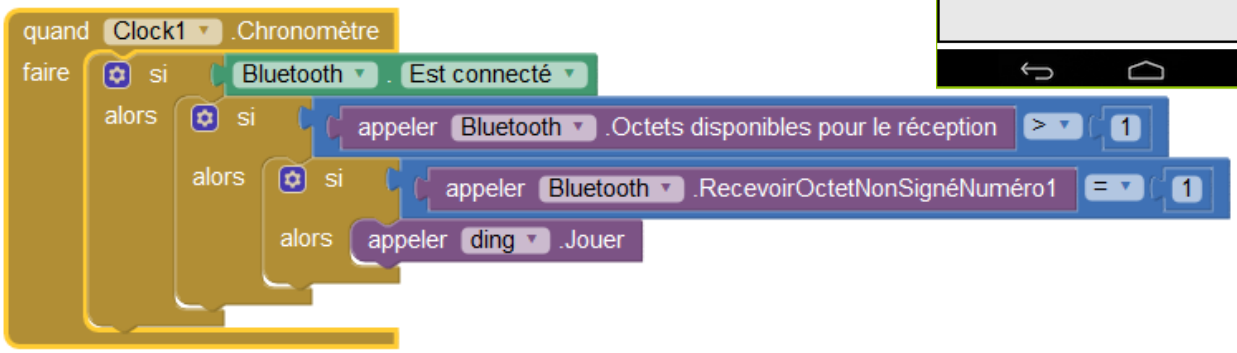
Exercice niveau 3 - A.4 : Envoyer et recevoir des données provenant d'un Smartphone

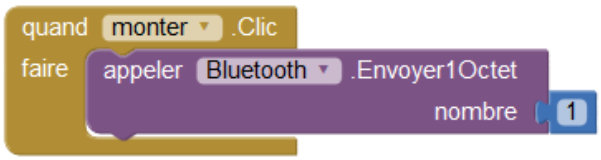


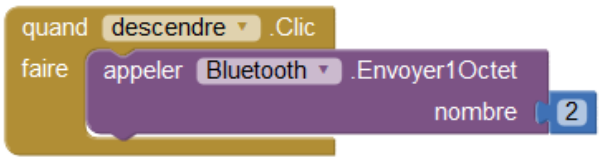
Objectif : Faire monter ou descendre le monte-charge à partir de boutons sur une application Bluetooth.
Jouer une sonnette lorsque le monte-charge s'arrête à un étage.
Allumer une LED à l'endroit où le monte-charge s'est arrêté. Eteindre cette LED lorsqu'il repart en mouvement.
Permettre également de faire monter ou descendre le monte-charge à l'aide des boutons poussoir.

Notion abordée : envoyer et recevoir des informations à l'aide du module Bluetooth à une application.

Application Android : MCharge_4.apk
App Inventor : MCharge_4.aia

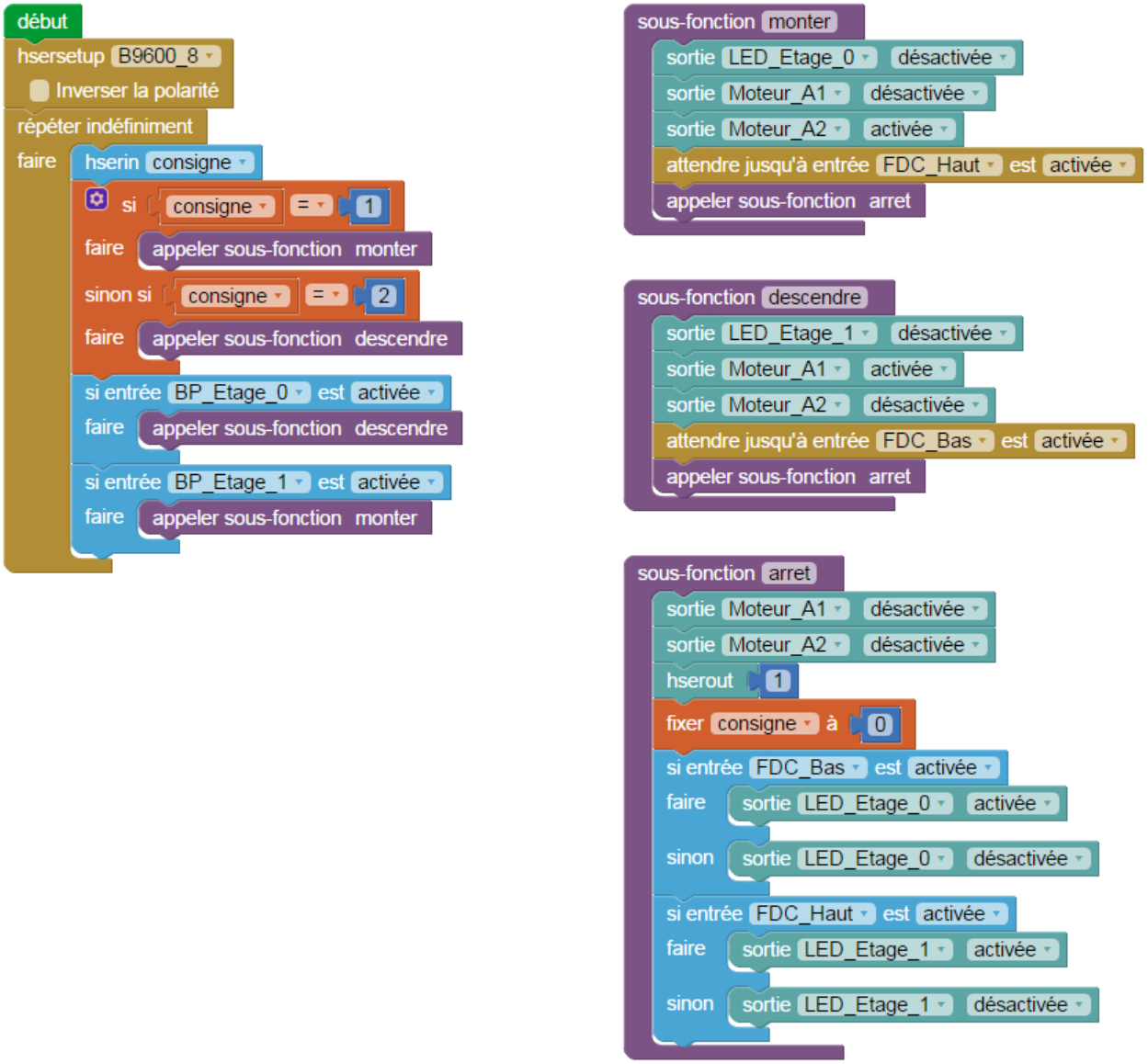






Correction :

Blocs



Fichier Blockly : MC_N3_A4.xml

Option : Module télécommande infrarouge

Télécommande RAX-TVR10

La télécommande universelle infrarouge associée à un capteur infrarouge approprié permet de piloter à distance une carte PICAXE.
Afin d'assurer la compatibilité de fonctionnement avec le système PICAXE il est nécessaire de l'initialiser avec le mode de fonctionnement au standard *« Sony TV ».*

Attention : L'émetteur infrarouge doit toujours être désactivé lors de l'utilisation de la télécommande infrarouge.

Procédure de mise en service pour PICAXE

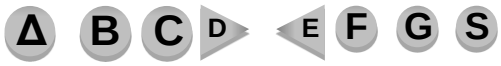
1. Insérer 2 piles AAA dans le logement au dos de la télécommande.

2. Appuyer simultanément sur S et B.
= La LED s'allume.

3. Taper le code 0 - 1 - 3
= La LED clignote brièvement à chaque appui des touches « 0 » et « 1 » puis s'éteint après l'appui sur la touche « 3 ».

4. Appuyer sur le bouton de mise en service. 
= La télécommande est opérationnelle.

Les touches suivantes risquent de déprogrammer :

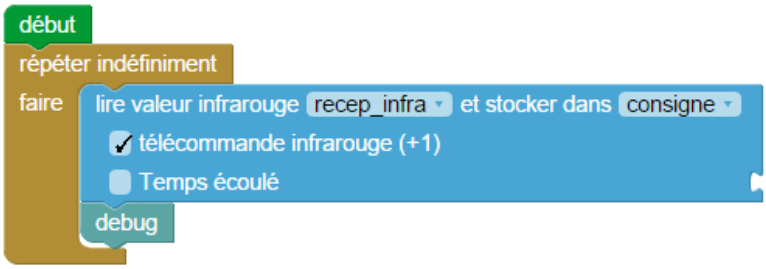


Conseil : Si la télécommande ne fonctionne plus, appuyer sur  pour revenir à la configuration compatible PICAXE

Remarque : Le guide d'utilisation complet de la télécommande est disponible ici :
http://www.a4telechargement.fr/RAX-TVR010/RAX-TRV10_Telecommande_InfraRouge.pdf

Tester la télécommande

Charger les programmes de test de la télécommande : « test_infra_bloc.xml » ou « test_infra_org.plf ».
Respecter le plan de câblage vu précédemment dans le dossier.



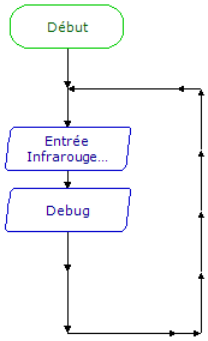


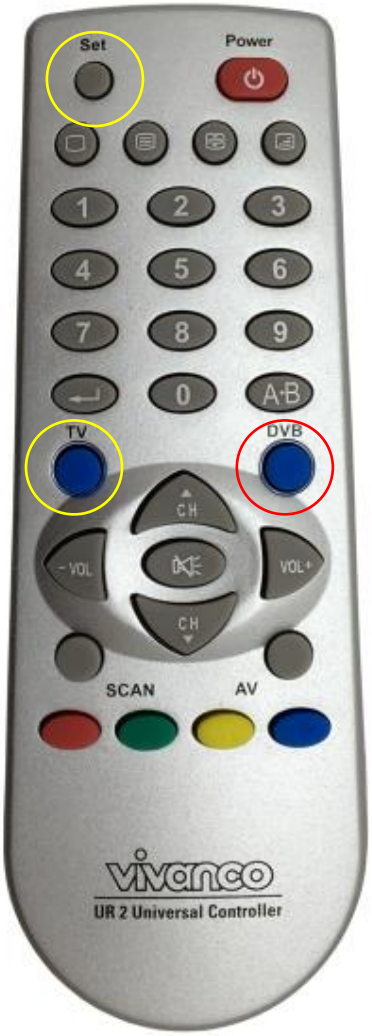
Tableau de correspondance des touches

Diriger la télécommande vers le récepteur infrarouge et vérifier dans la partie « Variables » de Picaxe Editor que les données reçues sont correctes.
Ci-dessous, le tableau des valeurs renvoyées par les différents boutons de la télécommande :

Touche	1	2	3	4	5	6	7	8	9	0	
Code émis	0	1	2	3	4	5	6	7	8	9	21
Code émis	16	17	19	18	96	54	37	20	98	11	

Télécommande TELEC-IR-UNIV

Procédure de mise en service pour PICAXE



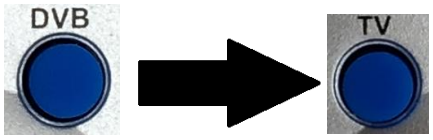
1. Insérer 2 piles AAA dans le logement au dos de la télécommande.

2. Appuyer simultanément sur les boutons **Set** et **TV**.
   Le bouton **Power** s'allume.

3. Taper le code 0-7-7.
   Le bouton **Power** clignote brièvement à chaque appui, puis s'éteint.

4. Appuyer sur le bouton **Power**.
   La télécommande est opérationnelle.

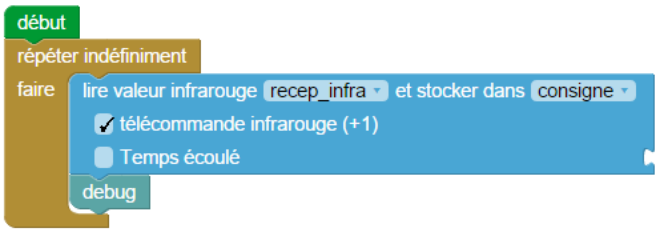
ATTENTION !
La touche **DVB** risque de changer le mode. Appuyer sur **TV** pour revenir dans le bon mode.



Conseil : Si la télécommande ne fonctionne plus, appuyer sur **TV** pour revenir à la configuration compatible PICAXE.

Tester la télécommande

Charger les programmes de test de la télécommande : « test_infra_bloc.xml » ou « test_infra_org.plf ».
Respecter le plan de câblage vu précédemment dans le dossier.



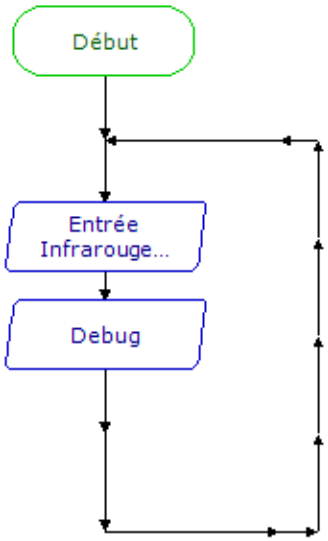
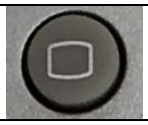
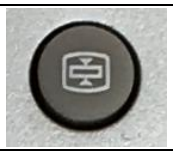
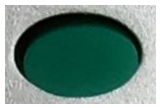
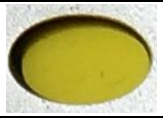


Tableau de correspondance des touches

L'appui sur une touche provoque l'émission d'un signal infrarouge qui véhicule un code correspondant à la touche. Par défaut : appui sur la touche 1 = envoi du code 0.

Pour simplifier l'utilisation de la télécommande infrarouge, il est possible d'activer la compatibilité entre le N° des touches et le code envoyé. Dans ce cas, appui sur la touche 1 = envoi du code 1.

Touche					
Code émis standard	**9**	**0**	**1**	**2**	**3**
Compatibilité activée	**10**	**1**	**2**	**3**	**4**
Touche					
Code émis standard	**4**	**5**	**6**	**7**	**8**
Compatibilité activée	**5**	**6**	**7**	**8**	**9**
Touche					
Code émis standard	**12**	**37**	**16**	**37**	**56**
Compatibilité activée	**13**	**38**	**17**	**38**	**57**
Touche					
Code émis standard	**63**	**74**	**29**	**21**	**76**
Compatibilité activée	**64**	**75**	**30**	**22**	**77**
Touche					
Code émis standard	**77**	**78**	**79**	**18**	**19**
Compatibilité activée	**78**	**79**	**80**	**19**	**20**
Touche					
Code émis standard	**20**	**16**	**17**		
Compatibilité activée	**21**	**17**	**18**		

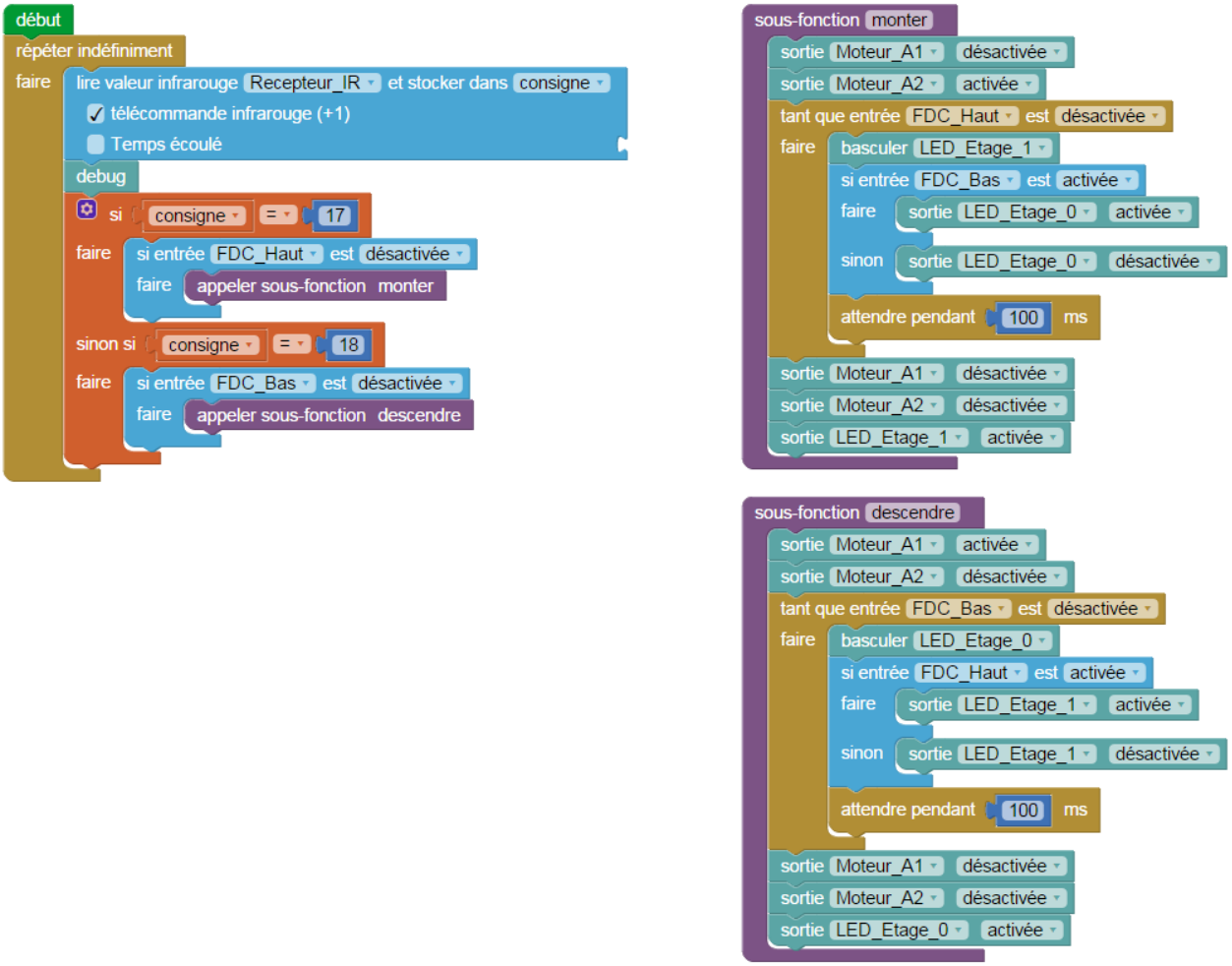
Exercice niveau 3 - B.1 : Contrôle la montée et la descente avec la télécommande IR

Objectif : Monter et descendre le monte-charge à l'aide de la télécommande IR.

Notion abordée : gestion d'une liaison infrarouge : télécommande/AutoProg à l'aide du bloc prévu à cet effet.

Correction :

Blocs



Fichier Blockly : MC_N3_A1.xml

Remarque : Cocher la case +1 permet d'avoir une concordance entre la touche pressée et la consigne reçue.

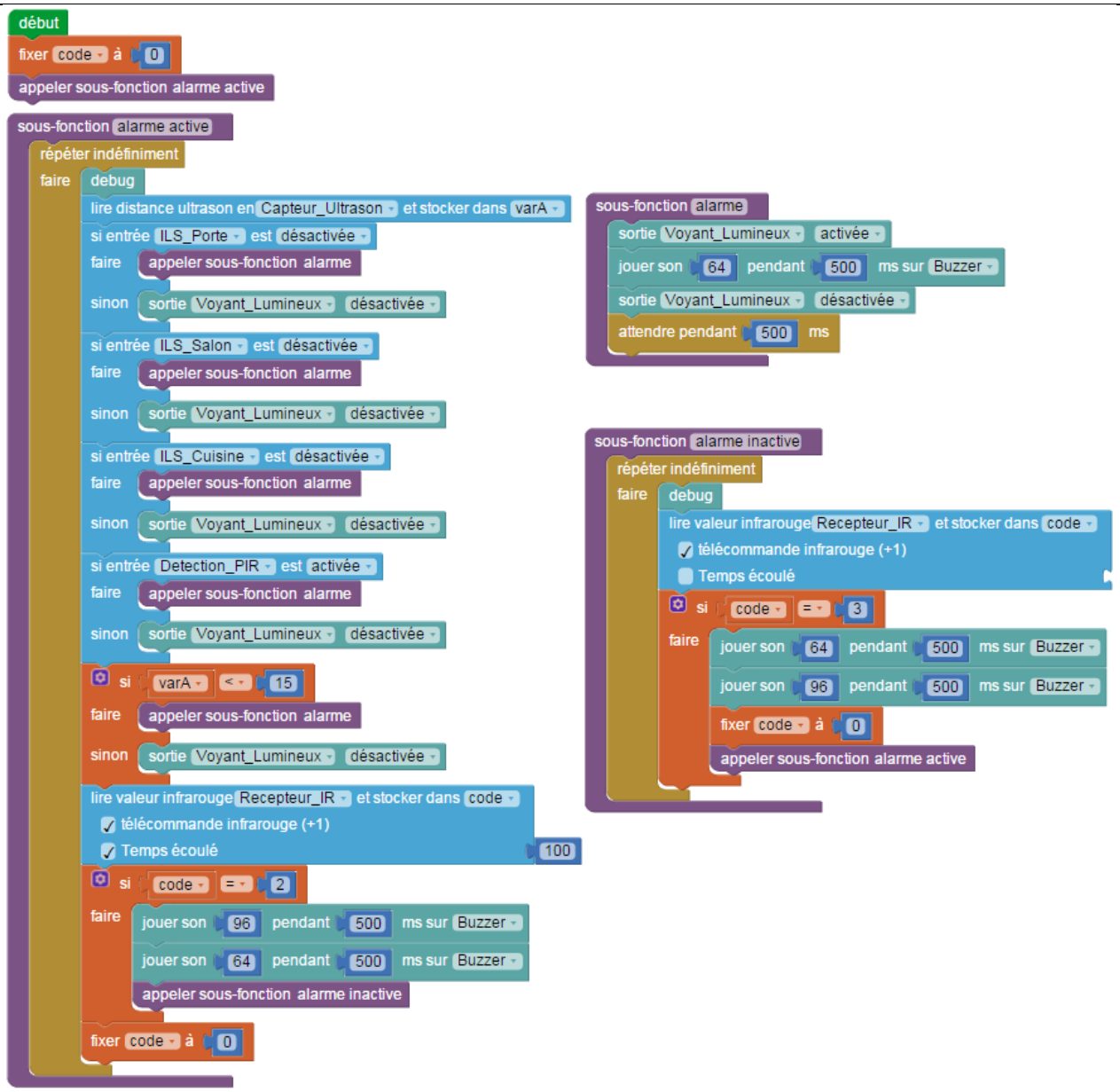
Exercice niveau 3 - B.2 : Activer / Désactiver l'alarme à l'aide de la télécommande IR

Objectif : Pouvoir activer ou désactiver l'alarme à partir de boutons sur la télécommande

Notion abordée : Utilisation du timeout sur la lecture infrarouge

Correction :

Blocs



Fichier Blockly : MC_N3_B2.xml

Remarque : Cocher la case timeout permet à la fonction de laisser place aux autres fonctions. Si on ne coche pas la case, la fonction va rester jusqu'à ce qu'on lui envoie une donnée et donc les autres fonctions ne seront pas atteignables.

Attention : Ne pas choisir un code envoyé à 1 avec un temps écoulé

CONCEPTEUR ET FABRICANT DE MATÉRIELS PÉDAGOGIQUES