

AutoLumi

Maquette programmable avec PICAXE Editor / Blockly pour PICAXE



```
début
sortie Lumiere_Salon_1 activée
attendre pendant 3000 ms
sortie Lumiere_Salon_1 désactivée
attendre pendant 3000 ms
sortie Lumiere_Salon_1 activée
attendre pendant 3000 ms
sortie Lumiere_Salon_1 désactivée
```



Ressources disponibles pour le projet AutoLumi

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

AutoLumi

- Fichiers **3D** (SolidWorks, Edrawings et Parasolid) de la maquette et de ses options.
- Dossier **technique** AutoLumi 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	1
AutoLumi.....	1
Les environnements de programmation graphique	1
Le dossier	1
Les fiches exercices	2
Prérequis	2
Caractéristiques techniques	2
Mise en œuvre d'AutoLumi.....	3
Tableau d'affectation des entrées et sorties.....	3
Programmation version de base niveau 1	4
Niveau 1 - A.....	5
Exercice niv. 1 - A.1 : Allumer / éteindre une lumière	5
Exercice niv. 1 - A.2: Allumer / éteindre une lumière deux fois.....	6
Exercice niv. 1 - A.3 : Allumer / éteindre une lumière indéfiniment.....	7
Niveau 1 - B.....	8
Exercice niv. 1 - B.1 : Allumer une lumière manuellement.....	8
Exercice niv. 1 - B.2 : Allumer/éteindre une lumière manuellement.....	9
Exercice niv. 1 - B.3 : Allumer une lumière tant que le bouton est actionné.....	10
Exercice niv. 1 - B.4 : Allumer une lumière si une présence est détectée	11
Niveau 1 - C.....	12
Exercice niv. 1 - C.1 : Allumer / éteindre toutes les lumières successivement	12
Exercice niv. 1 - C.2 : Allumer / éteindre toutes les lumières successivement avec un bouton	13
Niveau 1 - D.....	14
Exercice niv. 1 – D.1 : Afficher le niveau de lumière	14
Exercice niv. 1 – D.2 : Allumer/éteindre la lumière en fonction de la luminosité.....	15
Exercice niv. 1 – D.3 : Allumer la lumière si une présence est détectée et en fonction de la luminosité	16
Programmation version de base niveau 2	17
Niveau 2 - A.....	18
Exercice niv. 2 – A.1 : Utiliser une sous-fonction pour allumer la lumière	18
Exercice niv. 2 – A.2 : Utiliser une sous-fonction pour allumer la lumière lorsqu'une présence est détectée19	
Niveau 2 - B.....	20
Exercice niv. 2 – B.1 : Utilisation de variables.....	20
Exercice niv. 2 – B.2 : Utilisation de variables(2)	21
Exercice niv. 2 – C.1 : Automatiser l'éclairage de la maquette	22

Programmation niveau 3	23
Option : Module télécommande infrarouge	24
Télécommande RAX-TVR10.....	24
Télécommande TELEC-IR-UNIV.....	26
Exercice niv. 3 - A.1 : Contrôler une lumière avec la télécommande IR (deux boutons).....	28
Exercice niv. 3 - A.2 : Contrôler une lumière avec la télécommande IR (un seul bouton).....	29
Exercice niv. 3 - A.3 : Contrôler plusieurs lumières avec la télécommande IR.....	30
Exercice niv. 3 - A.4 : Contrôler toutes les lumières avec la télécommande IR.....	31
Option : Module Bluetooth	32
Configuration.....	32
Mise en place des programmes et procédure de connexion.....	33
Tableau d'affectation des entrées et sorties.....	34
Câblage du module Bluetooth (K-AP-MBLTH).....	34
Exercice niv. 3 - B.1 : Allumer/éteindre une lumière avec votre smartphone (deux boutons).....	35
Exercice niv. 3 - B.2 : Allumer/éteindre une lumière avec votre smartphone (un bouton).....	36
Exercice niv. 3 - B.3 : Envoi de données sur le smartphone.....	37
Exercice niv. 3 - B.4 : Envoyer et recevoir des données sur un smartphone.....	38

Introduction

AutoLumi

La maquette AutoLumi (BE-ALUMI) est une reproduction homothétique d'une maison éclairable automatisable : lumières, interrupteurs, capteur de lumière, capteurs de présence. 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 avec AutoLumi :

- 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, module de détection de mouvement PIR.

Les environnements de programmation graphique

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

- Gratuit
- Blocs et organigrammes (proche algorithme).
- Personnalisation des noms des entrées/sorties.
- Personnalisation du jeu d'instructions.
- Mode de simulation visuelle à l'écran pour mettre au point et déboguer 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 AutoLumi 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, module de détection de mouvement PIR, 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 :

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

Exemple : LU_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>
- **Maquette** AutoLumi (Réf. BE-ALUMI).
- **Câble de programmation** Picaxe USB (Réf : CABLE-USBPICAXE).
- **Interface programmable** AutoProgX1 ou X2 (Réf. K-APV2).
- 13 **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.

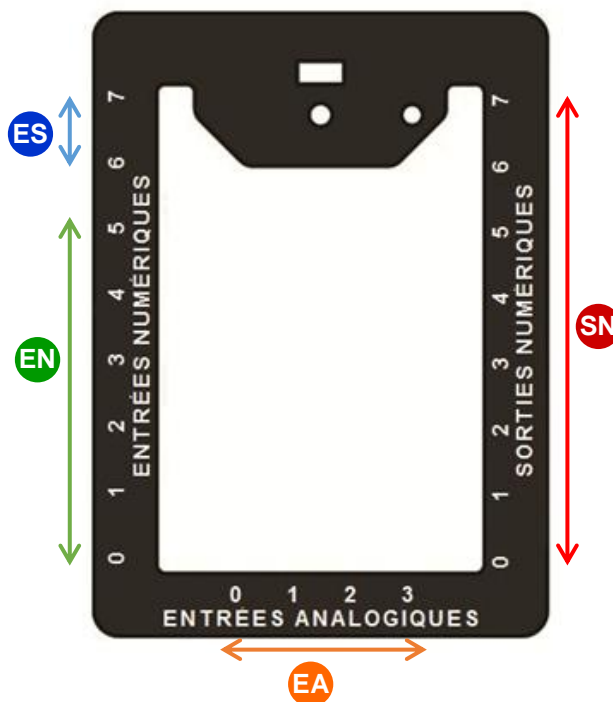
Caractéristiques techniques

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

Mise en œuvre d'AutoLumi

Tableau d'affectation des entrées et sorties

ES	MODULE DE COMMUNICATION POUR ENTRÉES / SORTIES NUMÉRIQUES	Broche Blockly	Etiquette Blockly
7	(libre)	C.7	
6	Récepteur infrarouge (option)	C.6	Recepteur_IR*
EN	MODULES CAPTEURS POUR ENTRÉES NUMÉRIQUES		
5	Capteur détection de présence	C.5	Detection_PIR
4	Bouton-poussoir entrée	C.4	BP_Entree
3	Bouton-poussoir 2 du salon	C.3	BP_Salon_2
2	Bouton-poussoir 1 du salon	C.2	BP_Salon_1
1	Bouton-poussoir des sanitaires	C.1	BP_Sanitaires
0	Bouton-poussoir de la cuisine	C.0	BP_Cuisine
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	(libre)	B.7	
6	(libre)	B.6	
5	Sortie lumière à l'entrée	B.5	Lumiere_Entree
4	Lumière au fond du salon	B.4	Lumiere_Salon_2
3	Lumière à l'entrée du salon	B.3	Lumiere_Salon_1
2	LED devant les sanitaires	B.2	LED_Sanitaires
1	Lumière dans les sanitaires	B.1	Lumiere_Sanitaires
0	Lumière dans la cuisine	B.0	Lumiere_Cuisine



Programmation version de base niveau 1

Objectifs :

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

Ce niveau permet de découvrir toutes les fonctionnalités de base d'AutoLumi, 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 « LU_N1_A1.xml », charger le programme modèle « LU_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 complètement la maquette.

Nom du fichier	Description	Objectif
Niveau 1 A Fichier modèle : LU_N1_A.xml		
LU_N1_A1	Allumer une lumière pendant 3 secondes puis l'éteindre.	Fonctionnalité matérielle abordée : -Allumage/extinction d'une lumière.
LU_N1_A2	Répéter cette action deux fois.	Notions de programmation abordées : -séquence d'instructions -temps d'attente -boucle infinie
LU_N1_A3	Répéter cette action à l'infini.	
Niveau 1 B Fichier modèle : LU_N1_B.xml		
LU_N1_B1	Activer une lumière sur l'appui d'un bouton-poussoir.	Fonctionnalité matérielle abordée : -utilisation de bouton-poussoir -utilisation d'un capteur tout ou rien
LU_N1_B2	Activer/désactiver une lampe à l'aide d'un bouton-poussoir.	
LU_N1_B3	Utiliser une boucle tant que.	Notions de programmation abordées : -boucle qui dépend d'une entrée
LU_N1_B4	Utilisation d'un capteur tout ou rien.	
Niveau 1 C Fichier modèle : LU_N1_C.xml		
LU_N1_C1	Activer une à une les lumières de la maison. Quand une lumière s'allume, une autre doit s'éteindre.	Fonctionnalité matérielle abordée : -Gestion de sorties
LU_N1_C2	Activer ou désactiver chaque lampe à l'aide de son bouton-poussoir.	Notions de programmation abordées : -Le test d'une entrée
Niveau 1 D Fichier modèle : LU_N1_D.xml		
LU_N1_D1	Lire une valeur sur un capteur analogique.	Fonctionnalité matérielle abordée :
LU_N1_D2	Activer une lumière lorsque la salle est éteinte.	Notions de programmation abordées : -Définition de variable -Test (si/sinon) de variable -Test (juste si) d'entrée -Débogage
LU_N1_D3	Lorsque la salle est éteinte et que le capteur PIR détecte une personne, allumer la lumière de l'entrée pendant 5 secondes.	

Niveau 1 - A

Exercice niv. 1 - A.1 : Allumer / éteindre une lumière

Fichier modèle : LU_BASE.xml

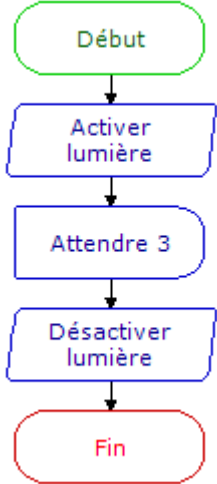
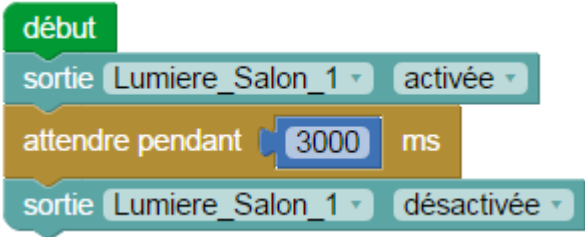
Objectif : allumer une lumière pendant 3 secondes puis l'éteindre.

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

Instructions utilisées :



Correction :

Organigramme	Blocs
 <pre>graph TD; A([Début]) --> B[Activer lumière]; B --> C[Attendre 3]; C --> D[Désactiver lumière]; D --> E([Fin]);</pre>	 <pre>graph TD; A[debut] --> B[sortie Lumiere_Salon_1 activée]; B --> C[attendre pendant 3000 ms]; C --> D[sortie Lumiere_Salon_1 désactivée];</pre>
Fichier organigramme PE6 : LU_N1_A1_Organigramme.pf	Fichier Blockly : LU_N1_A1.xml

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

Exercice niv. 1 - A.2: Allumer / éteindre une lumière deux fois

Objectif : allumer une lumière 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
<pre>graph TD; A([Début]) --> B[/Activer lumière/]; B --> C([Attendre 3]); C --> D[/Désactiver lumière/]; D --> E([Attendre 3]); E --> F[/Activer lumière/]; F --> G([Attendre 3]); G --> H[/Désactiver lumière/]; H --> I([Fin]);</pre>	
Fichier organigramme PE6 : LU_N1_A2_Organigramme.plf	Fichier Blockly : LU_N1_A2.xml

Exercice niv. 1 - A.3 : Allumer / éteindre une lumière indéfiniment

Objectif : faire clignoter une lumière toutes les 3 secondes indéfiniment.

Notion abordée : la boucle infinie.

Instructions utilisées :



Correction :

Organigramme	Blocs
<pre>graph TD; Start([Début]) --> On[Activer lumière]; On --> Wait3_1[Attendre 3]; Wait3_1 --> Off[Désactiver lumière]; Off --> Wait3_2[Attendre 3]; Wait3_2 --> On;</pre> <p>The flowchart starts with a 'Début' oval, followed by a sequence of four rectangular process boxes: 'Activer lumière', 'Attendre 3', 'Désactiver lumière', and 'Attendre 3'. A feedback loop arrow connects the end of the second 'Attendre 3' box back to the start of the 'Activer lumière' box, creating an infinite loop.</p>	<p>The Blockly code starts with a 'début' block, followed by a 'répéter indéfiniment' block. Inside the loop, there are four blocks: 'sortie Lumiere_Salon_1 activée', 'attendre pendant 3000 ms', 'sortie Lumiere_Salon_1 désactivée', and 'attendre pendant 3000 ms'.</p>
Fichier organigramme PE6 : LU_N1_A3_Organigramme.pf	Fichier Blockly : LU_N1_A3.xml

Remarque : le programme ne peut pas 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 l'interface AutoProg.

Niveau 1 - B

Exercice niv. 1 - B.1 : Allumer une lumière manuellement

Objectif : Activer une lumière en appuyant sur un bouton-poussoir.

Notion abordée : utilisation d'une entrée bouton-poussoir et d'une condition.

Instructions utilisées :



Correction :

Organigramme	Blocs
Fichier organigramme PE6 : LU_N1_B1_Organigramme.pf	Fichier Blockly : LU_N1_B1.xml

Exercice niv. 1 - B.2 : Allumer/éteindre une lumière manuellement

Objectif : Activer une lampe à l'aide d'un bouton-poussoir. Pour l'éteindre, il faut réappuyer sur le bouton.

Notion abordée : utilisation d'un moteur.

Instructions utilisées :



Correction :

Organigramme	Blocs
<pre>graph TD Start([Début]) --> Decision{BP activé ?} Decision -- Oui --> Process1[Basculer lumière] Process1 --> Process2[Attendre 0,5] Process2 --> Decision Decision -- Non --> Decision</pre>	<pre>graph TD Start([début]) --> Loop[répéter indéfiniment] Loop --> If[si entrée BP_Salon_1 est activée] If --> Do1[faire basculer Lumiere_Salon_1] Do1 --> Do2[attendre pendant 500 ms] Do2 --> Loop</pre>
Fichier organigramme PE6 : LU_N1_B2_Organigramme.plf	Fichier Blockly : LU_N1_B2.xml



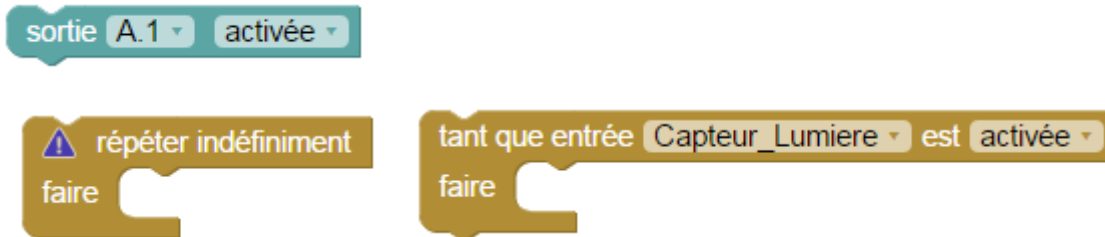
Remarque : La fonction `basculer Lumiere_Salon_1` permet de changer l'état d'une lumière. Le temps d'attente après celle-ci est dû au fait que lorsque le bouton est activé, la lumière va continuer à basculer et pourra rester éteinte ou allumée après un appui.

Exercice niv. 1 - B.3 : Allumer une lumière tant que le bouton est actionné

Objectif : Utiliser une nouvelle boucle

Notion abordée : utilisation d'une boucle tant que.

Instructions utilisées :



Correction :

Blocs

The script starts with a 'début' block, followed by a 'répéter indéfiniment' loop. Inside the loop, there is a 'faire' block containing a 'tant que' loop. The 'tant que' loop has 'BP_Salon_1' selected and 'est activée' as the condition. Inside this loop, there is a 'faire' block with 'sortie Lumiere_Salon_1' and 'activée'. Below the 'tant que' loop, there is a 'sortie Lumiere_Salon_1' block with 'désactivée' as the state.

Fichier Blockly : LU_N1_B3.xml

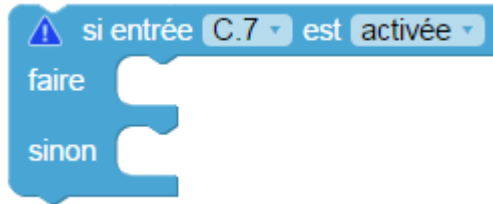
Notion complexe pour l'organigramme.

Exercice niv. 1 - B.4 : Allumer une lumière si une présence est détectée

Objectif : Activer une lumière lorsque le capteur PIR (Détecteur de présence) est activé.

Notion abordée : utilisation d'un capteur tout ou rien et d'une condition.

Instructions utilisées :



Correction :

Organigramme	Blocs
Fichier organigramme PE6 : LU_N1_B4_Organigramme.plf	Fichier Blockly : LU_N1_B4.xml

Niveau 1 - C

Exercice niv. 1 - C.1 : Allumer / éteindre toutes les lumières successivement

Objectif : Activer une à une les lumières de la maison. Quand une lumière s'allume, une autre doit s'éteindre.

Notion abordée : succession de consignes.

Instructions utilisées :



Correction :

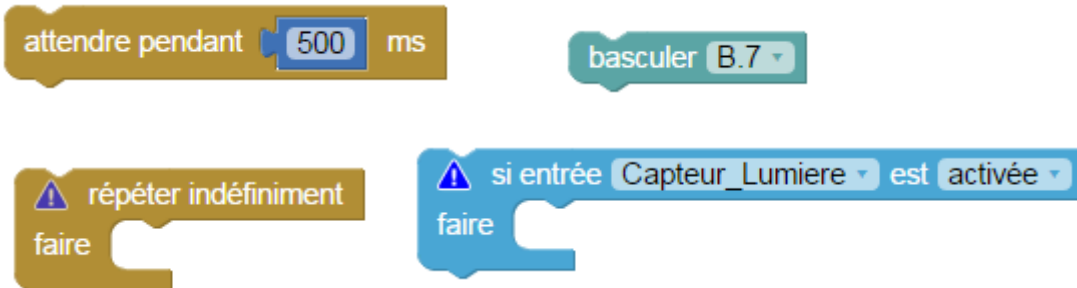
Organigramme	Blocs
<pre> graph TD Start([Début]) --> ActCuis[Activer cuisine] ActCuis --> Att05[Attendre 0,5] Att05 --> DesCuis[Désactiver cuisine] DesCuis --> ActSan[Activer sanitaires] ActSan --> Att05_2[Attendre 0,5] Att05_2 --> DesSan[Désactiver sanitaires] DesSan --> ActEnt[Activer entrée] ActEnt --> Att05_3[Attendre 0,5] Att05_3 --> DesEnt[Désactiver entrée] DesEnt --> ActSal1[Activer salon 1] ActSal1 --> Att05_4[Attendre 0,5] Att05_4 --> DesSal1[Désactiver salon 1] DesSal1 --> ActSal2[Activer salon 2] ActSal2 --> Att05_5[Attendre 0,5] Att05_5 --> DesSal2[Désactiver salon 2] DesSal2 --> ActCuis </pre>	<pre> début répéter indéfiniment faire sortie Lumiere_Salon_1 activée attendre pendant 500 ms sortie Lumiere_Salon_1 désactivée sortie Lumiere_Salon_2 activée attendre pendant 500 ms sortie Lumiere_Salon_2 désactivée sortie Lumiere_Cuisine activée attendre pendant 500 ms sortie Lumiere_Cuisine désactivée sortie Lumiere_Sanitaires activée attendre pendant 500 ms sortie Lumiere_Sanitaires désactivée sortie Lumiere_Entree activée attendre pendant 500 ms sortie Lumiere_Entree désactivée </pre>
<p>Fichier organigramme PE6 : LU_N1_C1_Organigramme.plf</p>	<p>Fichier Blockly : LU_N1_C1.xml</p>

Exercice niv. 1 - C.2 : Allumer / éteindre toutes les lumières successivement avec un bouton

Objectif : Activer ou désactiver chaque lampe à l'aide de son bouton-poussoir.
La LED des sanitaires doit s'activer en même temps que la lumière des sanitaires.

Notion abordée : boucles multiples.

Instructions utilisées :



Correction :

Organigramme	Blocs
<pre> graph TD Start([Début]) --> BP_Entrée{BP Entrée ?} BP_Entrée -- Oui --> Lumiere_Entrée[Lumière entrée] BP_Entrée -- Non --> BP_Salon_1{BP Salon 1} BP_Salon_1 -- Oui --> Lumiere_Salon_1[Lumière salon 1] BP_Salon_1 -- Non --> BP_Salon_2{BP Salon 2} BP_Salon_2 -- Oui --> Lumiere_Salon_2[Lumière salon 2] BP_Salon_2 -- Non --> BP_Cuisine{BP cuisine ?} BP_Cuisine -- Oui --> Lumiere_Cuisine[Lumière cuisine] BP_Cuisine -- Non --> Decision{Décision} Decision -- Oui --> Lumiere_Sanitaires[Lumière sanitaires] Decision -- Non --> Delay[Attendre 0,5] Lumiere_Entrée --> Delay Lumiere_Salon_1 --> Delay Lumiere_Salon_2 --> Delay Lumiere_Cuisine --> Delay Lumiere_Sanitaires --> Delay Delay --> BP_Entrée </pre>	
Fichier organigramme PE6 : LU_N1_C2_Organigramme.pf	Fichier Blockly : LU_N1_C2.xml

Niveau 1 - D

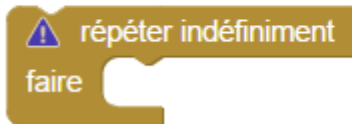
Exercice niv. 1 – D.1 : Afficher le niveau de lumière

Objectif : Récupérer la valeur analogique envoyée par un capteur de lumière.

Notion abordée : valeurs analogiques.

Instructions utilisées :

lire valeur analogique en Capteur_Lumiere et stocker dans varA



Correction :

Organigramme	Blocs
Fichier organigramme PE6 : LU_N1_D1_Organigramme.plf	Fichier Blockly : LU_N1_D1.xml

Remarque : un capteur analogique envoie une valeur entre 0 et 255. On considère que le capteur donne 0 dans un environnement totalement plongé dans le noir. La valeur augmente avec le niveau de lumière. Noter les valeurs de varA lorsque la salle est allumée et lorsqu'elle est éteinte.



La fonction  permet de rafraichir la valeur de varA à l'écran afin d'être lisible par l'utilisateur.

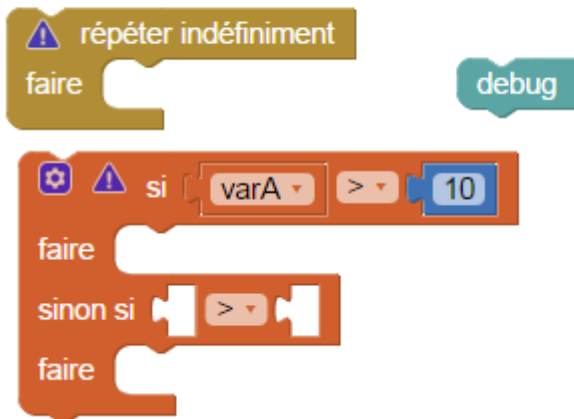
Exercice niv. 1 – D.2 : Allumer/éteindre la lumière en fonction de la luminosité

Objectif : Activer une lumière lorsque la salle est éteinte.

Notion abordée : conditions sur des valeurs analogiques.

Instructions utilisées :

lire valeur analogique en **Capteur_Lumiere** et stocker dans **varA**



Correction :

Organigramme	Blocs
<p>Fichier organigramme PE6 : LU_N1_D2_Organigramme.plf</p>	<p>Fichier Blockly : LU_N1_D2.xml</p>

Exercice niv. 1 – D.3 : Allumer la lumière si une présence est détectée et en fonction de la luminosité

Objectif : Lorsque la salle est éteinte et que le capteur PIR détecte une personne, allumer la lumière de l'entrée pendant 5 secondes. La lumière ne doit s'allumer que s'il fait sombre.

Notion abordée : conditions sur des valeurs analogiques.

Correction :

Organigramme	Blocs
<pre> graph TD Start([Début]) --> Read[Lire capteur de lumière] Read --> Debug[Debug] Debug --> VarA{varA < 200} VarA -- Non --> Deact[Désactiver lumière] Deact --> Start VarA -- Oui --> PIR{Capteur de présence PIR} PIR -- Oui --> Act[Activer lumière] Act --> Wait([Attendre 5]) Wait --> Deact PIR -- Non --> Deact </pre>	<pre> début répéter indéfiniment faire debug lire valeur analogique en Capteur_Lumiere et stocker dans varA si entrée Detection_PIR est activée faire si varA > 200 faire sortie Lumiere_Entree désactivée sinon si varA < 200 faire sortie Lumiere_Entree activée attendre pendant 5000 ms sinon sortie Lumiere_Entree désactivée </pre>
<p>Fichier organigramme PE6 : LU_N1_D3_Organigramme.plf</p>	<p>Fichier Blockly : LU_N1_D3.xml</p>

Programmation version de base niveau 2

Objectifs :

- Utilisation concrète d'AutoLumi
- 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 la maquette, au fur et à mesure des exercices vous allez utiliser de plus en plus de modules et enrichir votre code pour obtenir à la fin du niveau une maquette qui marche parfaitement et qui respecte une logique de fonctionnement calquée sur le réel.

Nom du fichier	Description	Objectif
Niveau 2 A Fichier modèle : LU_N2_A.xml		
LU_N2_A1	Créer une sous-fonction permettant d'allumer une lumière à l'appui d'un bouton-poussoir.	Notions de programmation abordées : -Utilisation des sous-fonctions
LU_N2_A2	Créer une sous-fonction qui allume la lumière pendant 5 secondes lorsque le capteur de présence détecte quelqu'un.	
Niveau 2 B Fichier modèle : LU_N2_B.xml		
LU_N2_B1	Créer une boucle qui augmente la valeur d'une variable toutes les secondes.	Notions de programmation abordées : -Manipulation de variables
LU_N2_B2	Créer une boucle qui augmente la valeur d'une variable toutes les secondes et à 10 secondes allume une lumière 3 secondes puis remet la variable à 0.	
Niveau 2 C Fichier modèle : LU_N2_C.xml		
LU_N2_C1	Créer un programme chargé de rendre possible tout l'éclairage de la maison.	Automatisation de la maquette

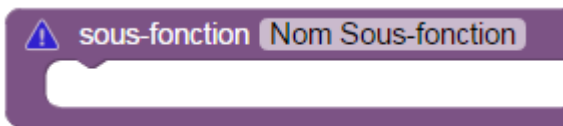
Niveau 2 - A

Exercice niv. 2 – A.1 : Utiliser une sous-fonction pour allumer la lumière

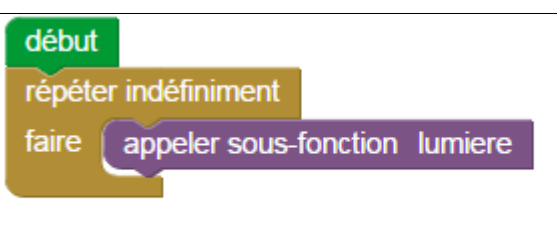
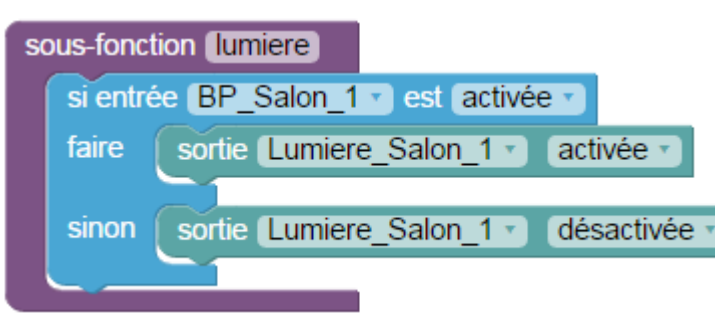
Objectif : Créer une sous-fonction permettant d'allumer une lumière à l'appui d'un bouton-poussoir.

Notion abordée : sous-fonction.

Instructions utilisées :



Correction :

Blocs



Fichier Blockly : LU_N2_A1.xml

Remarque : Le programme principal (sous le bloc début) ne doit contenir que la boucle de répétition et l'appel de la sous-fonction.

Exercice niv. 2 – A.2 : Utiliser une sous-fonction pour allumer la lumière lorsqu'une présence est détectée

Objectif : Créer une sous-fonction qui allume la lumière pendant 5 secondes lorsque le capteur de présence détecte quelqu'un. La lumière doit aussi pouvoir s'activer et se désactiver grâce à un bouton-poussoir.

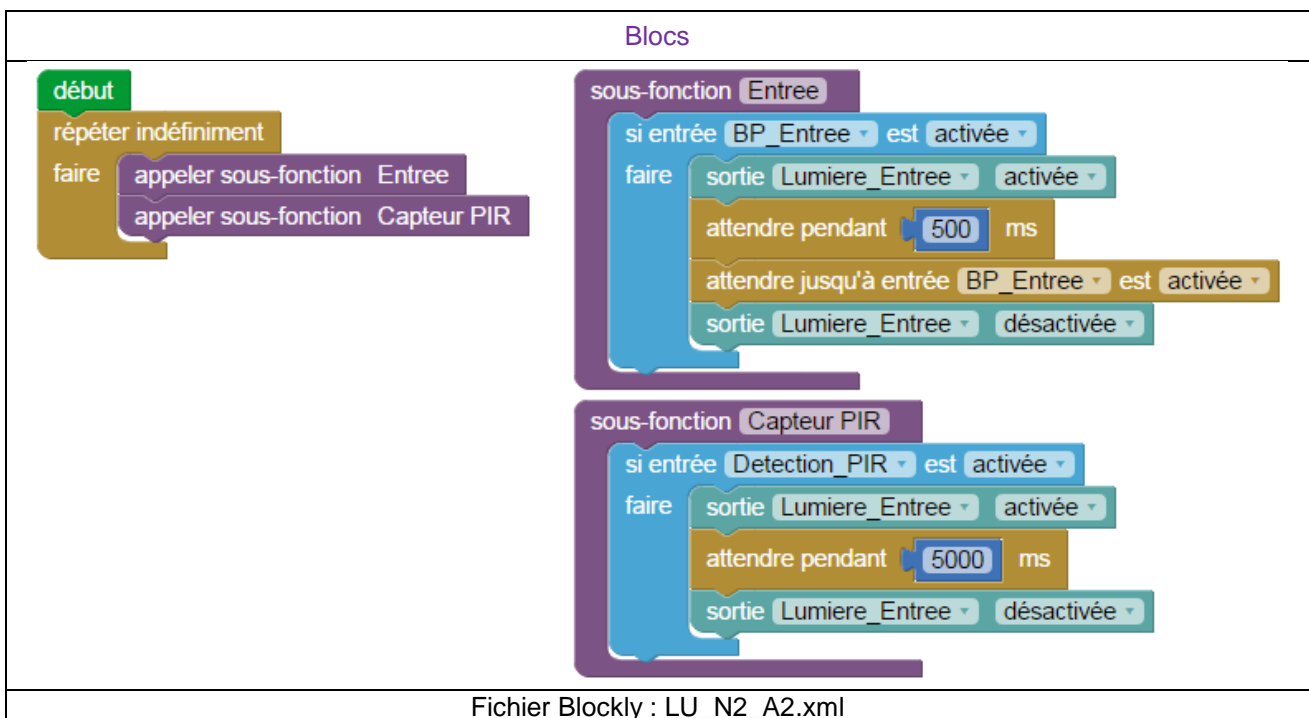
Notion abordée : Gestion de sous-fonctions.

Instructions utilisées :



Correction :

Blocs



Fichier Blockly : LU_N2_A2.xml

Niveau 2 - B

Exercice niv. 2 – B.1 : Utilisation de variables

Objectif : Créer une boucle qui augmente la valeur d'une variable toutes les secondes.

Notion abordée : gestion de variables.

Instructions utilisées :



Correction :

Blocs

```
graph TD
  Start[début] --> Set[fixer varA à 0]
  Set --> Loop[répéter indéfiniment]
  Loop --> Debug[debug]
  Loop --> Inc[incrémenter varA de 1]
  Loop --> Wait[attendre pendant 1000 ms]
  Loop --> Loop
```

Fichier Blockly : LU_N2_B1.xml

Exercice niv. 2 – B.2 : Utilisation de variables(2)

Objectif : Créer une boucle qui augmente la valeur d'une variable toutes les secondes et à 10 secondes allume une lumière pendant 3 secondes puis remet la variable à 0.

Notion abordée : gestion de variables.

Instructions utilisées :



Correction :

Blocs

```
graph TD
  Start[début] --> Set0[fixer varA à 0]
  Set0 --> Loop[répéter indéfiniment]
  Loop --> Do[faire]
  Do --> Debug[debug]
  Do --> Inc1[incrémenter varA de 1]
  Do --> Wait1[attendre pendant 1000 ms]
  Do --> If[si varA = 10]
  If --> DoInner[faire]
  DoInner --> Out1[sortie Lumiere_Entree activée]
  DoInner --> Wait2[attendre pendant 3000 ms]
  DoInner --> Out2[sortie Lumiere_Entree désactivée]
  DoInner --> Set0
  Do --> Loop
```

Fichier Blockly : LU_N2_B2.xml

Exercice niv. 2 – C.1 : Automatiser l'éclairage de la maquette

Objectif : Créer un programme chargé de rendre possible tout l'éclairage de la maison.

- L'appui sur un bouton allume ou éteint la lumière correspondante (comme un vrai interrupteur).
- Un passage sur le capteur de présence PIR allume la lumière puis l'éteint après quelques secondes.
- Si la lumière extérieure est trop forte, toutes les lumières s'éteignent ou doivent rester éteintes sauf les sanitaires.

Indice : utiliser des sous-fonctions et une variable pour le capteur de présence.

Correction :

Blocs

```
graph TD
    subgraph Main_Script
        Start[début] --> SetPresence[fixer presence à 0]
        SetPresence --> Loop[ répéter indéfiniment ]
        Loop --> Debug[faire debug]
        Loop --> ReadSensor[lire valeur analogique en Capteur_Lumiere et stocker dans varA]
        Loop --> CallBoutons[appeler sous-fonction boutons]
        Loop --> CallPresence[appeler sous-fonction presence]
        Loop --> CheckSanitaires[si entrée BP_Sanitaires est activée]
        CheckSanitaires --> ToggleLED[faire basculer LED_Sanitaires]
        ToggleLED --> ToggleLight[basculer Lumiere_Sanitaires]
        ToggleLight --> Delay500[attendre pendant 500 ms]
    end

    subgraph SubFunction_boutons
        direction TB
        S1[si varA < 200] --> F1[faire]
        F1 --> S1_1[si entrée BP_Cuisine est activée]
        S1_1 --> F1_1[faire]
        F1_1 --> T1_1[basculer Lumiere_Cuisine]
        T1_1 --> D1_1[attendre pendant 500 ms]
        F1 --> S1_2[si entrée BP_Salon_1 est activée]
        S1_2 --> F1_2[faire]
        F1_2 --> T1_2[basculer Lumiere_Salon_1]
        T1_2 --> D1_2[attendre pendant 500 ms]
        F1 --> S1_3[si entrée BP_Salon_2 est activée]
        S1_3 --> F1_3[faire]
        F1_3 --> T1_3[basculer Lumiere_Salon_2]
        T1_3 --> D1_3[attendre pendant 500 ms]
        F1 --> S1_4[si entrée BP_Entree est activée]
        S1_4 --> F1_4[faire]
        F1_4 --> T1_4[basculer Lumiere_Entree]
        T1_4 --> D1_4[attendre pendant 500 ms]
        S1 --> S2[sinon si varA > 200]
        S2 --> F2[faire]
        F2 --> O1[sortie Lumiere_Salon_1 désactivée]
        F2 --> O2[sortie Lumiere_Salon_2 désactivée]
        F2 --> O3[sortie Lumiere_Cuisine désactivée]
        F2 --> O4[sortie Lumiere_Entree désactivée]
    end

    subgraph SubFunction_presence
        direction TB
        S3[si Detection_PIR est activée] --> F3[faire]
        F3 --> I1[incrémenter presence de 1]
        F3 --> S4[si entrée Lumiere_Entree est désactivée]
        S4 --> F4[faire]
        F4 --> S5[si varA < 200]
        S5 --> F5[faire]
        F5 --> SetPresence2[fixer presence à 0]
        SetPresence2 --> O5[sortie Lumiere_Entree activée]
        S3 --> S6[si presence = 20]
        S6 --> F6[faire]
        F6 --> O6[sortie Lumiere_Entree désactivée]
    end
```

Fichier Blockly : LU_N2_C1.xml

Programmation niveau 3



Option : Module télécommande infrarouge

Télécommande RAX-TRV10

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 **B** pour revenir à la configuration compatible PICAXE

Remarque : Le guide d'utilisation complet de la télécommande est disponible ici : http://www.a4telechargement.fr/RAX-TRV10/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.

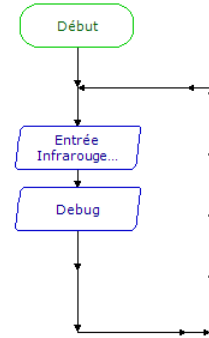
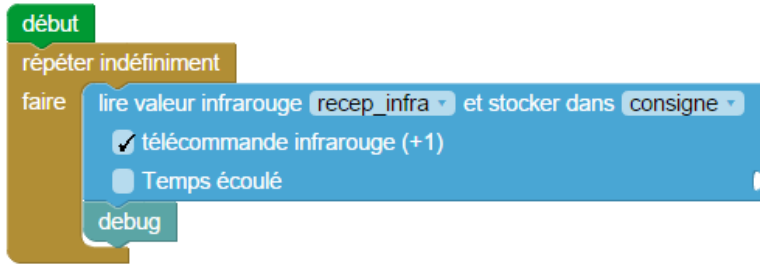













Tableau de correspondance des touches

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.

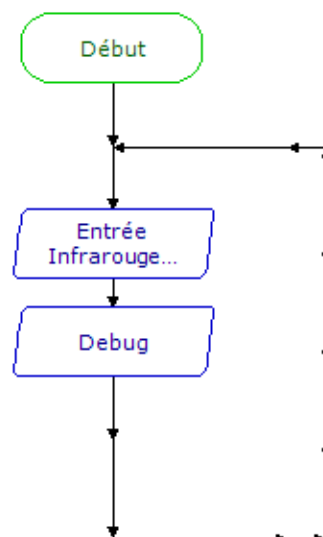
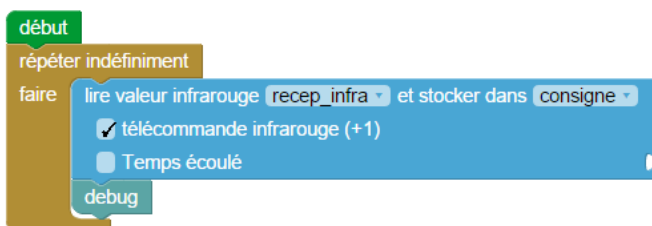










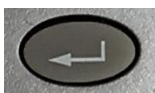




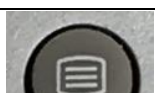


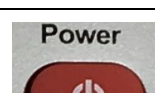











Tableau de correspondance des touches

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 niv. 3 - A.1 : Contrôler une lumière avec la télécommande IR (deux boutons)

Objectif : Allumer une lumière en appuyant sur 1, l'éteindre en appuyant sur 2.

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

Instructions utilisées :

```
lire valeur infrarouge Recepteur_IR et stocker dans consigne
 télécommande infrarouge (+1)
 Temps écoulé
```

Correction :

Blocs

```
début
répéter indéfiniment
faire
  lire valeur infrarouge Recepteur_IR et stocker dans consigne
   télécommande infrarouge (+1)
   Temps écoulé
  si consigne = 1
    faire sortie Lumiere_Entree activée
  sinon si consigne = 2
    faire sortie Lumiere_Entree désactivée
```

Fichier Blockly : LU_N3_A1.xml

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

Exercice niv. 3 - A.2 : Contrôler une lumière avec la télécommande IR (un seul bouton)

Objectif : Allumer ou éteindre une lumière avec un seul bouton 1.

Correction :

Blocs

```
graph TD
  Start[début] --> Loop[répéter indéfiniment]
  Loop --> Read[lire valeur infrarouge Recepteur_IR et stocker dans consigne]
  Read --> Check[si consigne = 1]
  Check --> Toggle[faire basculer Lumiere_Entree]
  Toggle --> Wait[attendre pendant 500 ms]
  Wait --> Loop
```

Fichier Blockly : LU_N3_A2.xml

Exercice niv. 3 - A.3 : Contrôler plusieurs lumières avec la télécommande IR

Objectif : Allumer ou éteindre des lumières à l'aide de plusieurs boutons.

Correction :

Blocs

```
graph TD
  Start[début] --> Loop[répéter indéfiniment]
  Loop --> Read[faire lire valeur infrarouge Recepteur_IR et stocker dans consigne]
  Read --> Check1[si consigne = 1]
  Check1 --> Toggle1[faire basculer Lumiere_Salon_1]
  Check1 --> Check2[si consigne = 2]
  Check2 --> Toggle2[faire basculer Lumiere_Salon_2]
  Toggle1 --> Wait[attendre pendant 500 ms]
  Toggle2 --> Wait
  Wait --> Loop
```

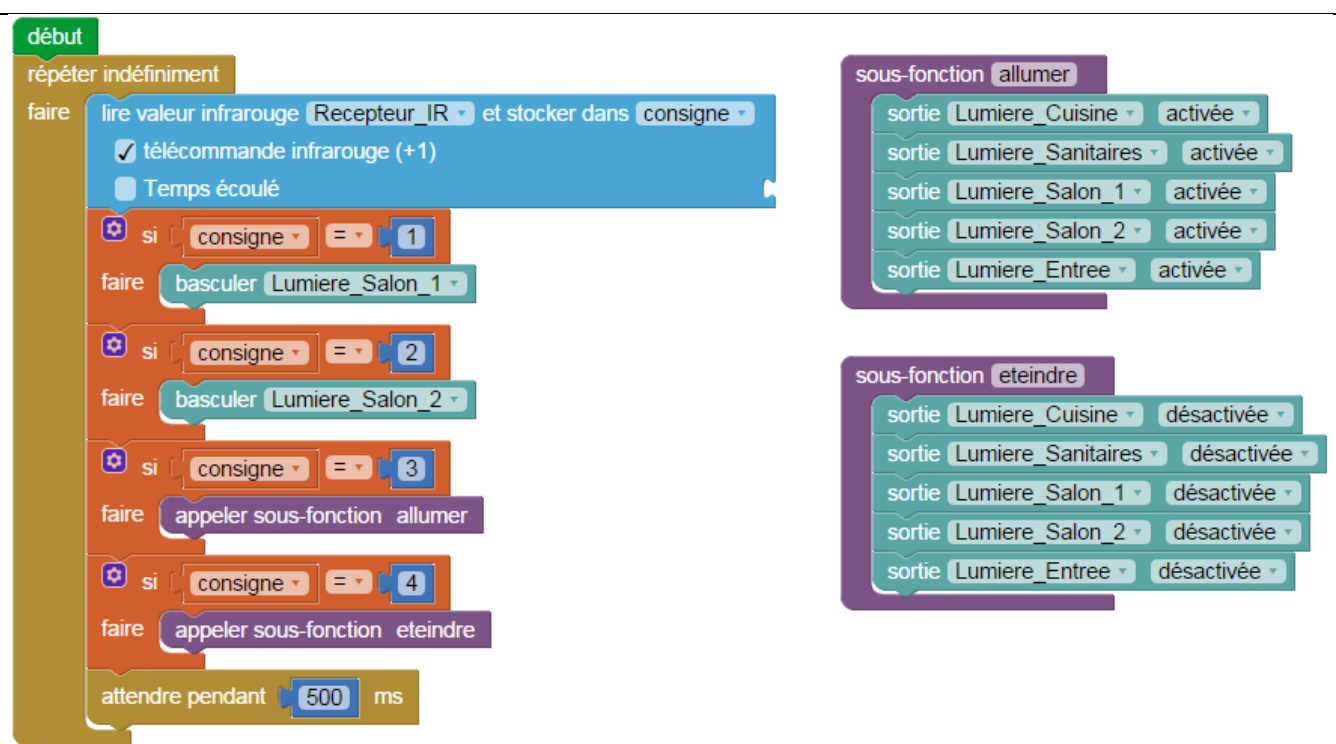
Fichier Blockly : LU_N3_A3.xml

Exercice niv. 3 - A.4 : Contrôler toutes les lumières avec la télécommande IR

Objectif : Reprendre le programme précédent. Rajouter une commande qui allume toute la maison et une autre qui éteint toute la maison. Utiliser les sous-fonctions pour ces deux objectifs.

Correction :

Blocs



```
graph TD
    Start([début]) --> Loop[ répéter indéfiniment ]
    Loop --> Read[ lire valeur infrarouge Recepteur_IR et stocker dans consigne ]
    Read --> Check1[ si consigne = 1 ]
    Check1 --> Toggle1[ faire basculer Lumiere_Salon_1 ]
    Check1 --> Check2[ si consigne = 2 ]
    Check2 --> Toggle2[ faire basculer Lumiere_Salon_2 ]
    Check2 --> Check3[ si consigne = 3 ]
    Check3 --> CallAllumer[ faire appeler sous-fonction allumer ]
    Check3 --> Check4[ si consigne = 4 ]
    Check4 --> CallEteindre[ faire appeler sous-fonction eteindre ]
    CallAllumer --> Delay[ attendre pendant 500 ms ]
    CallEteindre --> Delay
    Delay --> Loop
```

Fichier Blockly : LU_N3_A4.xml

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.

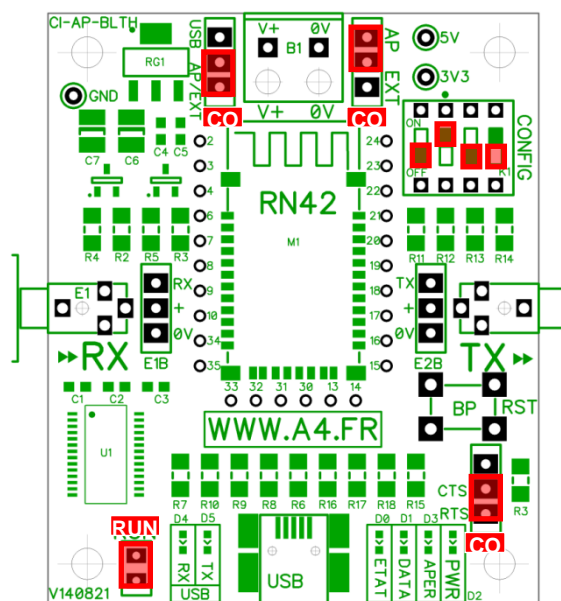
La documentation technique du module Bluetooth décrit en détail les fonctionnalités du module.

Elle est téléchargeable sur [http://a4.fr/wiki/index.php/Module Bluetooth - K-AP-MBLTH / S-113020008](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 Applinventor 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.

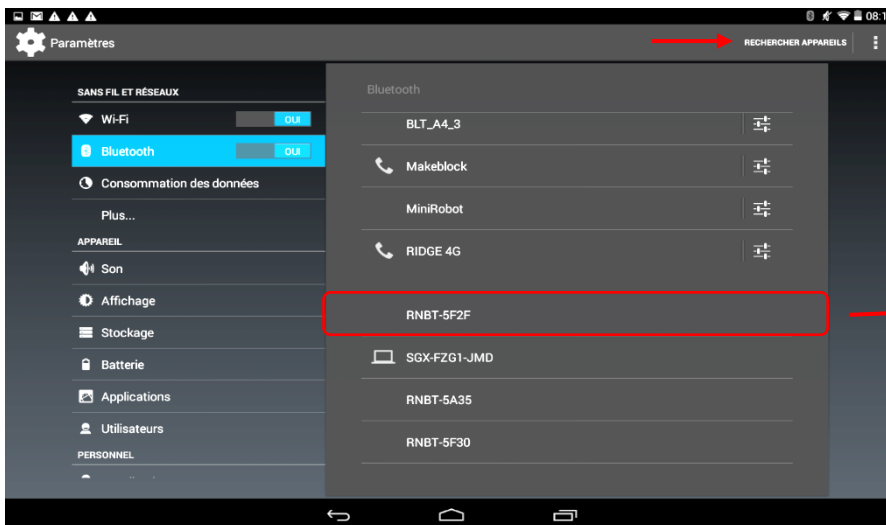
USB RX indique qu'il y a un flux de données sur la liaison USB du PC vers le module.

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 alimenter le module). Le nom de votre module s'appelle : RNBT + les 4 derniers chiffres de l'adresse mac du module notés sur le composant. Sélectionnez le et un message proposant de vous connecter à lui devrait s'afficher.



Une fois cette étape passée vous pourrez vous connecter au module à partir du programme Applinventor 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.

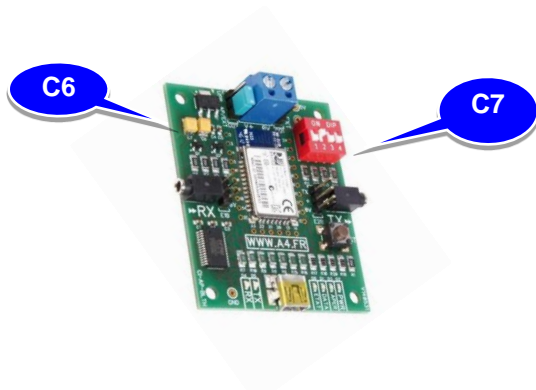


Tableau d'affectation des entrées et sorties

ES	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	Capteur détection de présence	C.5	Detection_PIR
4	Bouton-poussoir entrée	C.4	BP_Entree
3	Bouton-poussoir 2 du salon	C.3	BP_Salon_2
2	Bouton-poussoir 1 du salon	C.2	BP_Salon_1
1	Bouton-poussoir des sanitaires	C.1	BP_Sanitaires
0	Bouton-poussoir de la cuisine	C.0	BP_Cuisine
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	(libre)	B.7	
6	(libre)	B.6	
5	Sortie lumière à l'entrée	B.5	Lumiere_Entree
4	Lumière au fond du salon	B.4	Lumiere_Salon_2
3	Lumière à l'entrée du salon	B.3	Lumiere_Salon_1
2	LED devant les sanitaires	B.2	LED_Sanitaires
1	Lumière dans les sanitaires	B.1	Lumiere_Sanitaires
0	Lumière dans la cuisine	B.0	Lumiere_Cuisine

* Pour ces exercices, vous devez débrancher le récepteur infrarouge afin de pouvoir brancher le module Bluetooth.

Câblage du module Bluetooth (K-AP-MBLTH)



Exercice niv. 3 - B.1 : Allumer/éteindre une lumière avec votre smartphone (deux boutons)

Objectif : contrôler une lumière à l'aide de 2 boutons présent sur l'application Android **Lumi_B1** développée sous AppInventor.

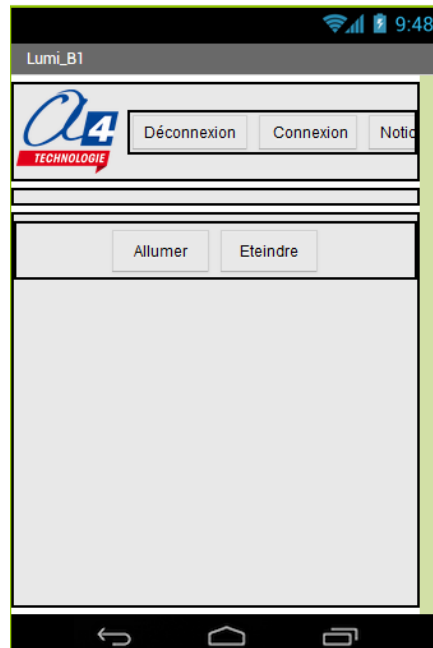
Notion abordée : réception de données Bluetooth envoyées par un Smartphone.

Application Android : Lumi_B1.apk

Fichier App Inventor : Lumi_1.aia

```
quand Allumer .Clic
faire appeler Bluetooth .Envoyer1Octet
      nombre 1

quand Eteindre .Clic
faire appeler Bluetooth .Envoyer1Octet
      nombre 2
```



Correction :

```
Blocs
début
  hsersetup B9600_8
  Inverser la polarité
  répéter indéfiniment
  faire
    debug
    hserin consigne
    si consigne = 1
    faire sortie Lumiere_Entree activée
    sinon si consigne = 2
    faire sortie Lumiere_Entree désactivée
```

Fichier Blockly : LU_N3_B1.xml

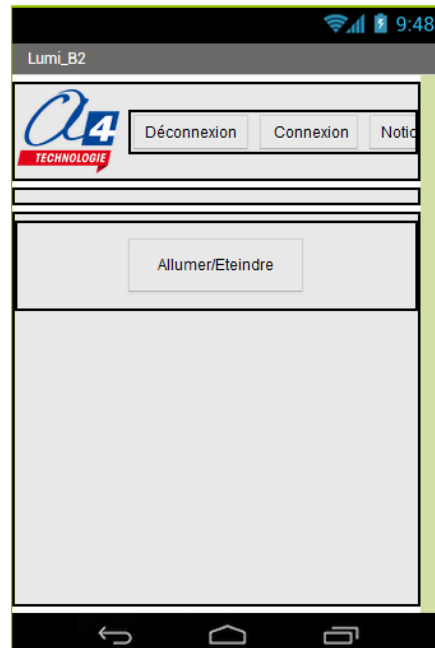
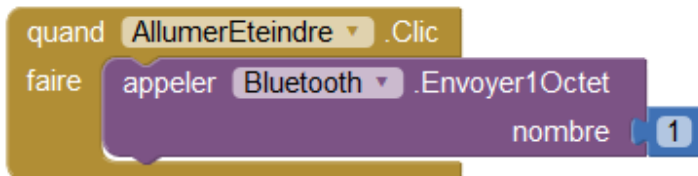
Exercice niv. 3 - B.2 : Allumer/éteindre une lumière avec votre smartphone (un bouton)

Objectif : contrôler une lumière à l'aide de 1 bouton présent sur l'application Android **Lumi_B2** développée sous AppInventor.

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

Application Android : Lumi_B2.apk

Fichier App Inventor : Lumi_2.aia



Correction :

Blocs

Code blocks for the 'début' event:

- début
- hersetup B9600_8
 - Inverser la polarité
- répéter indéfiniment
 - faire
 - debug
 - hserin consigne
 - si consigne = 1
 - faire
 - si entrée Lumiere_Entree est activée
 - faire sortie Lumiere_Entree désactivée
 - sinon sortie Lumiere_Entree activée
 - fixer consigne à 0

Fichier Blockly : LU_N3_B2.xml

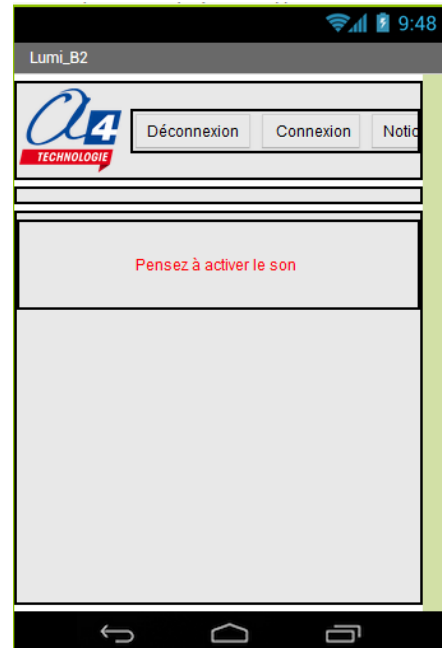
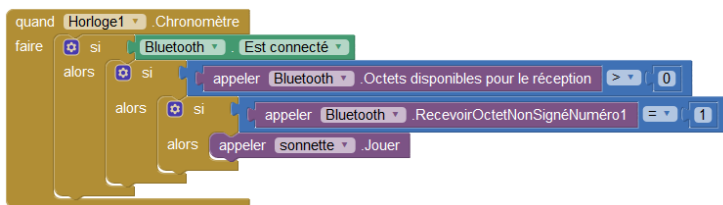
Exercice niv. 3 - B.3 : Envoi de données sur le smartphone

Objectif : jouer un son sur le smartphone lors d'un appui sur un bouton-poussoir de la maquette.

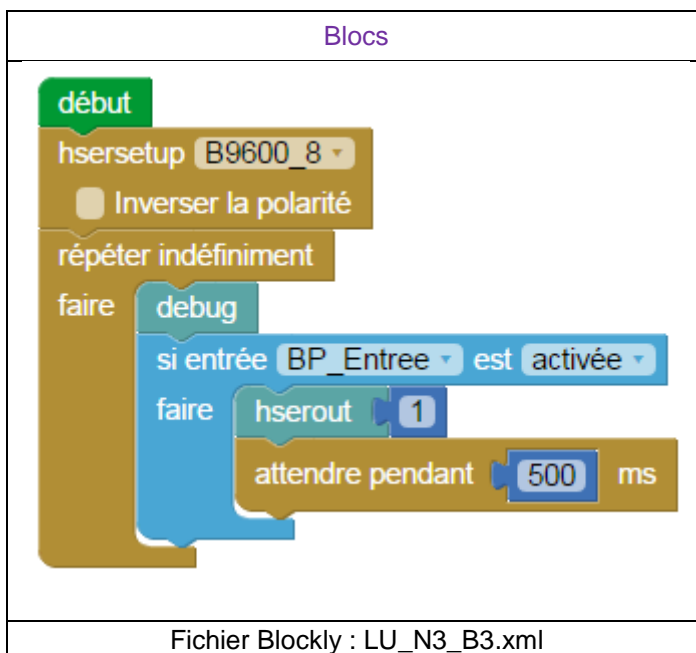
Notion abordée : Envoi de données Bluetooth vers un Smartphone.

Application Android : Lumi_B3.apk

Fichier App Inventor : Lumi_3.aia



Correction :



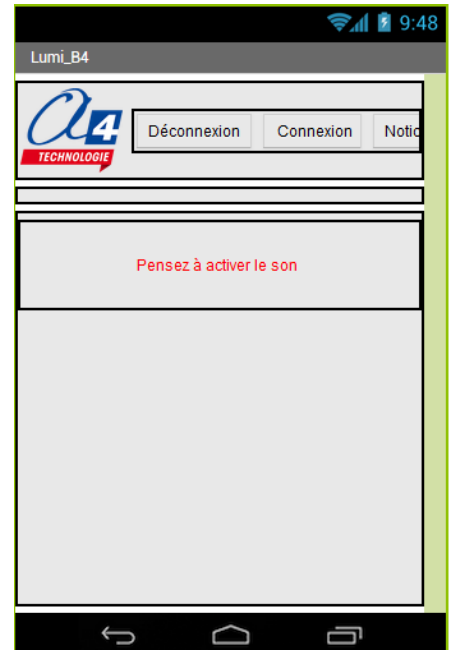
Exercice niv. 3 - B.4 : Envoyer et recevoir des données sur un smartphone

Objectif : Envoyer une information à un smartphone et la traiter sur celui-ci.

Notion abordée : Envoi et réception de données Bluetooth sur un Smartphone.

Application Android : Lumi_B4.apk
Fichier App Inventor : Lumi_4.aia

```
quand Horloge1 Chronomètre  
faire  
  si Bluetooth Est connecté  
  alors  
    si Bluetooth Octets disponibles pour la réception > 0  
    alors  
      si Bluetooth RecevoirOctetNonSignéNuméro1 = 1  
      alors  
        appeler sonnette Jouer  
        appeler Evénement Afficher fenêtre choix  
          message Souhaitez-vous allumer la lumière ?  
          Titre Bouton poussoir activé  
          Texte bouton 1 Oui (Allumer)  
          Texte bouton 2 Non (Eteindre)  
          annulable faux  
quand Evénement Après choix  
Choix  
faire  
  si obtenir Choix = "Oui (Allumer)"  
  alors  
    appeler Bluetooth Envoyer l'Octet nombre 1  
  sinon  
    appeler Bluetooth Envoyer l'Octet nombre 2
```



Correction :

```
Blocs  
début  
hsersetup B9600_8  
  Inverser la polarité  
répéter indéfiniment  
faire  
  debug  
  si entrée BP_Entree est activée  
  faire  
    hserout 1  
    attendre pendant 500 ms  
    hserin consigne  
    si consigne = 1  
    faire sortie Lumiere_Entree activée  
    sinon si consigne = 2  
    faire sortie Lumiere_Entree désactivée
```

Fichier Blockly : LU_N3_B4.xml



CONCEPTEUR ET FABRICANT DE MATÉRIELS PÉDAGOGIQUES