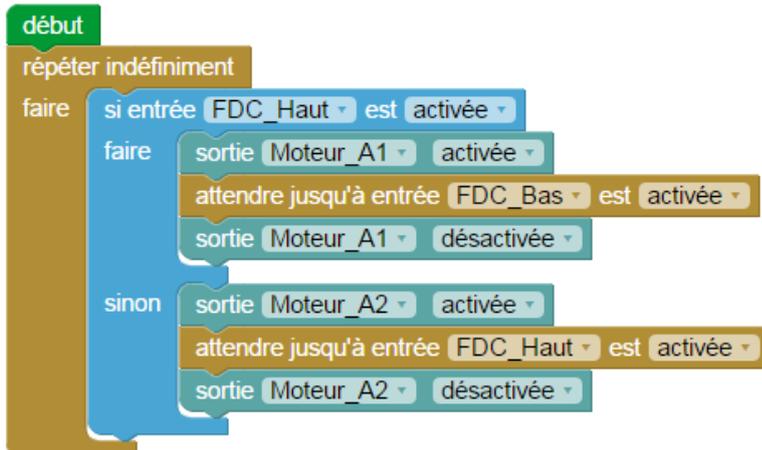


# Mini-serre

## Maquette programmable avec Editor / Blockly pour PICAXE



# Ressources disponibles pour le projet

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

## Mini-serre

- Fichiers **3D** (SolidWorks, Edrawings et Parasolid) de la maquette et de ses options.
- Dossier **technique** pour la mise en œuvre de la maquette ;

## 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 Applinventor.

**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](http://www.a4.fr)**

# SOMMAIRE

|   |           |
|---|-----------|
| <b>Introduction .....</b>   | <b>5</b>  |
| Mini-serre .....  | 5         |
| Les environnements de programmation graphique .....   | 5         |
| Le dossier .....  | 5         |
| Les fiches exercices .....  | 6         |
| Prérequis .....   | 6         |
| Caractéristiques techniques .....   | 6         |
| <b>Environnement de programmation graphique .....</b>   | <b>7</b>  |
| Personnalisation des entrées/ sorties .....   | 7         |
| Tableau d'affectation des entrées et sorties .....  | 8         |
| Personnalisation du jeu d'instructions .....  | 9         |
| Procédure de chargement d'un programme .....  | 9         |
| Mode simulation .....   | 10        |
| <b>Programmation version de base niveau 1 .....</b>   | <b>11</b> |
| <b>Niveau 1 - A .....</b>   | <b>12</b> |
| Exercice niveau 1 - A.1 : Activer / désactiver un moteur .....                                  | 12        |
| Exercice niveau 1 - A.2 : Activer / désactiver un moteur indéfiniment .....                     | 13        |
| Exercice niveau 1 - A.3 : Activer / désactiver plusieurs sorties .....                          | 14        |
| Exercice niveau 1 - A.4 : Utilisation d'un capteur fin de course avec une boucle tant que ..... | 15        |
| Exercice niveau 1 - B.1 : Utilisation d'un capteur de température .....                         | 16        |
| Exercice niveau 1 - B.2 : Utilisation d'une boucle conditionnelle Si .....                      | 17        |
| Exercice niveau 1 - B.3 : Utilisation du capteur d'humidité .....                               | 18        |
| Exercice niveau 1 - B.4 : Opérations sur une variable .....                                     | 19        |
| Exercice niveau 1 - C.1 : Utilisation de la pompe .....   | 20        |
| Exercice niveau 1 - C.2 : Arrosage automatique .....  | 21        |
| Exercice niveau 1 - D.1 : Utilisation des sous-fonctions .....                                  | 22        |
| Exercice niveau 1 - D.2 : Sous fonctions pour les capteurs .....                                | 23        |
| Exercice niveau 1 - D.3 : Conditions dans les sous-fonctions .....                              | 24        |
| Exercice niveau 1 - D.4 : Serre automatique .....   | 25        |
| <b>Programmation version de base niveau 2 .....</b>   | <b>26</b> |
| Exercice niveau 2 - A.1 : Bouton-poussoir .....   | 27        |
| Exercice niveau 2 - A.2 : Déclencher des moteurs grâce au bouton poussoir .....                 | 28        |
| Exercice niveau 2 - B.1 : Afficher un message sur l'afficheur OLED .....                        | 29        |
| Exercice niveau 2 - B.2 : Afficher une variable sur l'afficheur OLED .....                      | 30        |
| Exercice niveau 2 - B.3 : Afficher plusieurs variables sur l'afficheur OLED .....               | 31        |
| Exercice niveau 2 - B.4 : Afficher l'état d'une sortie sur l'afficheur OLED .....               | 32        |
| Exercice niveau 2 - C.1 : Utiliser une sonde hygrométrique .....                                | 33        |
| Exercice niveau 2 - C.2 : Arroser en cas d'hygrométrie trop faible .....                        | 34        |
| Exercice niveau 2 - C.3 : Arroser en cas d'hygrométrie trop faible (2) .....                    | 35        |
| Exercice niveau 2 - D.1 : Utilisation du plateau chauffant .....                                | 36        |
| Exercice niveau 2 - D.2 : Régulation de température .....                                       | 37        |
| Exercice niveau 2 - E.1 : Utilisation d'un brumisateur .....                                    | 38        |
| Exercice niveau 2 - E.2 : Régulation de l'humidité .....  | 39        |
| Exercice niveau 2 - E.3 : Régulation de l'humidité (2) .....                                    | 40        |

|  |           |
|--|-----------|
| <b>Programmation niveau 3</b> .....  | <b>41</b> |
| Exercice niveau 3 – A.1 : Programme final, automatisation de la serre.....                             | 41        |
| <b>Option : Module Bluetooth</b> .....   | <b>43</b> |
| Exercice niveau 3 - B.1 : Monter/Descendre le toit avec application Bluetooth .....                    | 46        |
| Exercice niveau 3 - B.2 : Activer/Désactiver une sortie avec une application Bluetooth .....           | 47        |
| Exercice niveau 3 - B.3 Envoi de données sur smartphone.....   | 48        |
| Exercice niveau 3 – AFF : Envoyer un message à l’afficheur OLED via Bluetooth (Option afficheur) ..... | 49        |
| Exercice niveau 3 – C.3 : Capteur de courant et variable .....   | 50        |

# Introduction

---

## Mini-serre

La maquette Mini-serre (BE-SER) est une reproduction homothétique d'une serre automatisée : Gestion de la température et de l'humidité automatique, arrosage des plantes, ouverture et fermeture du toit, 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 système de plateforme :

- 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 : brumisateur, bouton poussoir, plateau chauffant, afficheur OLED, des applications pour smartphone.

## Les environnements de programmation graphique

Tous les programmes correspondant aux activités menées autour de la maquette 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 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 : brumisateur, plateau chauffant, afficheur OLED, applications pour smartphone.

## 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 :

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

Exemple : MS\_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** : <http://www.picaxe.com/Software>
- **Maquette** Mini-serre (Réf. BE-SER).
- **Câble de programmation** Picaxe USB (Réf : CABLE-USBPICAXE).
- **Interface programmable** AutoProgX1 ou X2 (Réf. K-APV2).
- **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.

# Environnement de programmation graphique

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

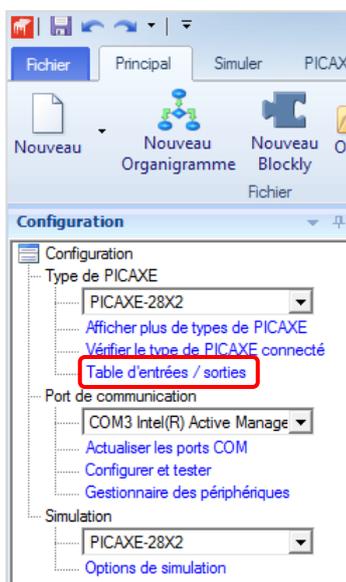
- Gratuit
- Blocs et organigrammes (proche algorithme).
- Personnalisation des noms des entrées/sorties.
- Personnalisation du jeu d'instructions.
- Mode de simulation visuelle à l'écran pour mettre au point et déboguer les programmes.

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

## Personnalisation des entrées/ sorties

Nous vous proposons le fichier **MS\_BASE.xml** dans lequel les noms des entrées/sorties ont été personnalisés pour une utilisation avec la maquette. Tous les programmes et activités proposés dans ce document se basent sur cette liste. Celle-ci reste modifiable à tout moment.

A partir de Picaxe Editor 6, dans l'explorateur d'espace de travail cliquer sur **Table d'entrées / sorties**.



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

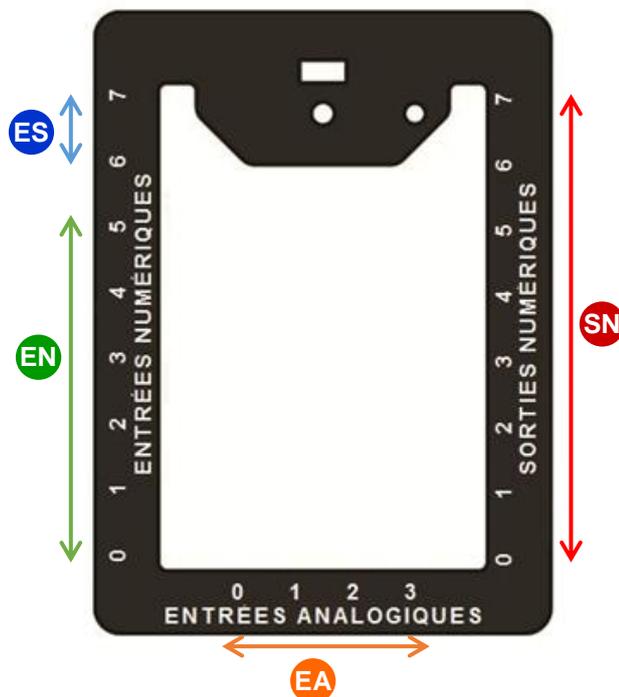
| Table d'entrées / sorties |   |
|---------------------------|---|
| c.0                       | <input type="text" value="ILS_Cuisine"/>      |
| c.1                       | <input type="text" value="ILS_Salon"/>        |
| c.2                       | <input type="text" value="ILS_Porte"/>        |
| c.3                       | <input type="text" value="Detection_PIR"/>    |
| c.4                       | <input type="text" value="Recepteur_IR"/>     |
| c.5                       | <input type="text" value="Capteur_Ultrason"/> |
| c.6                       | <input type="text" value="BLTH_TX"/>          |

OK Annuler

Valider en cliquant sur **OK**.

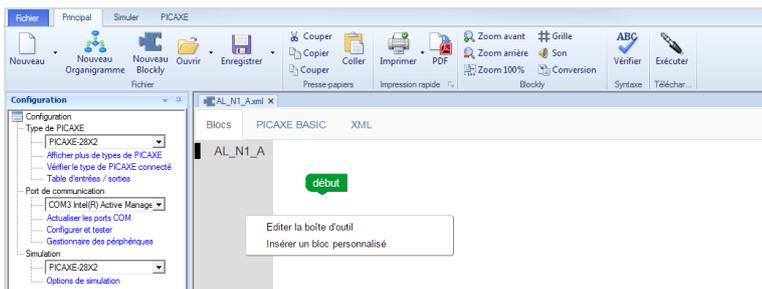
## Tableau d'affectation des entrées et sorties

| ES | MODULE DE COMMUNICATION POUR ENTRÉES / SORTIES NUMÉRIQUES | Broche Blockly | Etiquette Blockly  |
|----|---|----------------|--------------------|
| 7  | Bouton poussoir situé à l'avant de la serre               | C.7            | Bouton_Poussoir*   |
| 6  | (libre)   | C.6            |                    |
| EN | MODULES CAPTEURS POUR ENTRÉES NUMÉRIQUES                  |                |                    |
| 5  | (libre)   | C.5            |                    |
| 4  | (libre)   | C.4            |                    |
| 3  | Capteur de température                                    | C.3            | Capteur_temp       |
| 2  | (libre)   | C.2            |                    |
| 1  | Fin de course fenêtre dépliée                             | C.1            | FDC_Deplie         |
| 0  | Fin de course fenêtre repliée                             | C.0            | FDC_Replie         |
| EA | MODULES CAPTEURS POUR ENTRÉES ANALOGIQUES                 |                |                    |
| 3  | (libre)   | A.3            |                    |
| 2  | (libre)   | A.2            |                    |
| 1  | Capteur d'humidité  | A.1            | Capteur_humidite   |
| 0  | Sonde hygrométrique                                       | A.0            | Sonde_hygro*       |
| SN | MODULES ACTIONNEURS SORTIES NUMÉRIQUES                    |                |                    |
| 7  | Sortie reliée à la broche 2 du moteur                     | B.7            | Moteur_2           |
| 6  | Sortie reliée à la broche 1 du Moteur                     | B.6            | Moteur_1           |
| 5  | Ventilateur   | B.5            | Ventilateur        |
| 4  | Pompe à eau   | B.4            | Pompe              |
| 3  | Brumisateur   | B.3            | Brumisateur*       |
| 2  | Plateau chauffant   | B.2            | Plateau_chauffant* |
| 1  | Afficheur OLED  | B.1            | Afficheur_OLED*    |
| 0  | (libre)   | B.0            |                    |

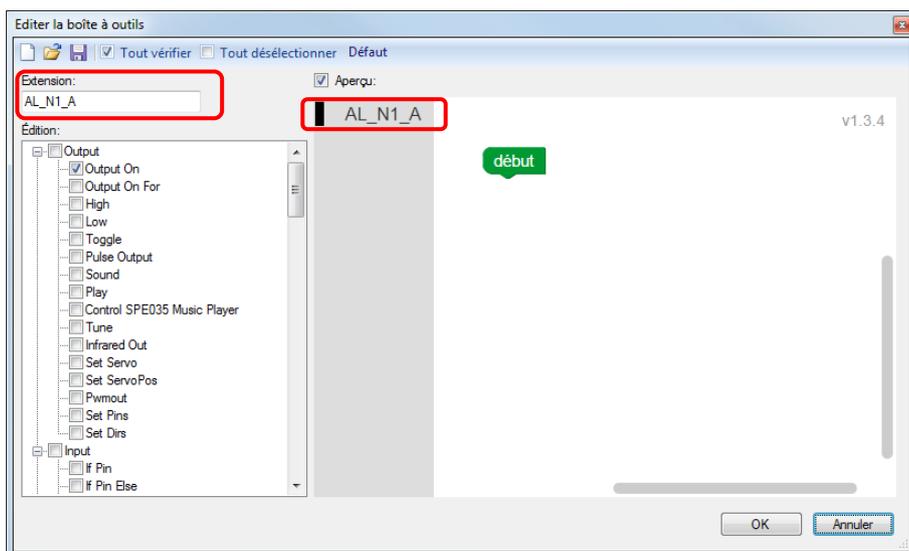


## Personnalisation du jeu d'instructions

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



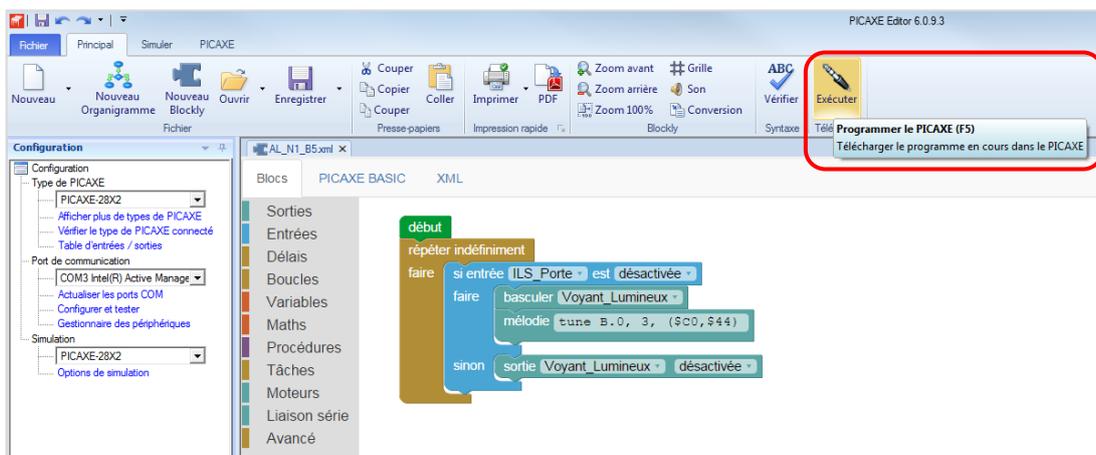
Une fenêtre s'ouvre à partir de laquelle vous pouvez sélectionner ou désélectionner les instructions de votre choix. Vous pouvez renommer le jeu d'instructions dans la zone « **Extension** ».



Valider en cliquant sur **OK**.

## Procédure de chargement d'un programme

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



A partir du menu **Principal** ou du menu **PICAXE**, cliquer sur le bouton **Exécuter**. Vous pouvez également utiliser la touche **F5** de votre clavier.

**Note** : un programme téléchargé écrase le précédent.

## Mode simulation

La simulation sur Picaxe EDITOR 6 permet de tester un programme avant de le téléverser dans la maquette. Pour lancer et contrôler une simulation, utiliser les boutons **Exécuter / Pause / Pas à pas / Arrêt** à partir du menu **Simuler**.



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

# 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 de la mini-serre, 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 « MS\_N1\_A1.xml », charger le programme modèle « MS\_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   |
|--|---|--|
| <b>Niveau 1 A - Fichier modèle : MS_BASE.xml</b> |   |  |
| MS_N1_A1.xml                                     | Allumer le moteur dans les deux sens pendant 3 secondes puis l'éteindre.                        | Fonctionnalité matérielle abordée :<br>-Gestion du moteur<br>-Utilisation de Bouton-poussoir<br>Notions de programmation abordées :<br>-Boucle qui dépend d'une entrée |
| MS_N1_A2.xml                                     | Répéter cette même action à l'infini.   |  |
| MS_N1_A3.xml                                     | Activer plusieurs sorties.  |  |
| MS_N1_A4.xml                                     | Activer un moteur jusqu'à une fin de course.  |  |
| <b>Niveau 1 B - Fichier modèle : MS_BASE.xml</b> |   |  |
| MS_N1_B1.xml                                     | Récupérer et lire une valeur sur un capteur.  | Fonctionnalité matérielle abordée :<br>-Utilisation de bouton-poussoir   |
| MS_N1_B2.xml                                     | Activer une sortie en fonction d'une variable.  |  |
| MS_N1_B3.xml                                     | Récupérer et lire une valeur analogique sur un capteur d'humidité.                              | Notions de programmation abordées :<br>-Le test d'une entrée (si/sinon)  |
| MS_N1_B4.xml                                     | Opérations sur une variable.  |  |
| <b>Niveau 1 C - Fichier modèle : MS_BASE.xml</b> |   |  |
| MS_N1_C1.xml                                     | Finaliser une action à l'aide d'une variable.   | Fonctionnalité matérielle abordée :<br>-Pompe  |
| MS_N1_C2.xml                                     | Créer un délai à partir d'une variable.   |  |
| <b>Niveau 1 D - Fichier modèle : MS_BASE.xml</b> |   |  |
| MS_N1_D1.xml                                     | Créer une sous-fonction chargée d'ouvrir ou de fermer la fenêtre et les effectuer indéfiniment. | Notions de programmation abordées :<br>-Utilisation des sous-fonctions   |
| MS_N1_D2.xml                                     | Créer des sous-fonctions chargées de récupérer les valeurs des capteurs et les appeler.         |  |
| MS_N1_D3.xml                                     | Faire tourner le ventilateur à une certaine température et l'arrêter à une autre.               |  |
| MS_N1_D4.xml                                     | Automatisation de la serre.   |  |

# Niveau 1 - A

## Exercice niveau 1 - A.1 : Activer / désactiver un moteur

Fichier modèle : MS\_BASE.xml

**Objectif** : activer un moteur pendant 3 secondes, puis l'autre pendant 3 secondes et stopper les moteurs

**A noter** : Les moteurs provoquent un déplacement de la barre située en haut à l'arrière de la maquette. Il est préférable de bouger manuellement cette barre au milieu pour ne pas bloquer le moteur.

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

**Instructions utilisées** :



**Correction** :

| Organigramme   | Blocs                          |
|--|--------------------------------|
|  |                                |
| Fichier organigramme PE6 : MS_N1_A1_Organigramme.plf | Fichier Blockly : MS_N1_A1.xml |

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

1000ms (millisecondes) = 1 seconde

# Exercice niveau 1 - A.2 : Activer / désactiver un moteur indéfiniment

**Objectif :** activer un moteur pendant 3 secondes, puis l'autre pendant 3 secondes et stopper les moteurs pendant 3 secondes. Répéter indéfiniment ces trois étapes.

**Notions abordées :** Boucle de répétition

**Instructions utilisées :**



**Correction :**

| Organigramme   | Blocs  |
|--|--|
| <pre> graph TD     Start([Start]) --&gt; M1[Moteur sens 1]     M1 --&gt; A1[Attendre 3]     A1 --&gt; M2[Moteur sens 2]     M2 --&gt; A2[Attendre 3]     A2 --&gt; STOP[Moteur STOP]     STOP --&gt; A3[Attendre 3]     A3 --&gt; Start                     </pre> | <pre> début   répéter indéfiniment     faire       sortie Moteur_1 activée       attendre 3000 ms       sortie Moteur_1 désactivée       sortie Moteur_2 activée       attendre 3000 ms       sortie Moteur_2 désactivée       attendre 3000 ms                     </pre> |
| <p>Fichier organigramme PE6 : MS_N1_A2_Organigramme.plf</p>  | <p>Fichier Blockly : MS_N1_A2.xml</p>  |

**Remarque :** La boucle répéter indéfiniment sera toujours active, il n'est donc pas possible de créer une instruction après celle-ci.

# Exercice niveau 1 - A.3 : Activer / désactiver plusieurs sorties

**Objectif :** activer un moteur pendant 3 secondes, l'arrêter et activer le ventilateur pendant 3 secondes

**Instructions utilisées :**

sortie A.0 activée

attendre pendant 500 ms

**Correction :**

| Organigramme   | Blocs   |
|--|---|
| <pre>graph TD; Start([Start]) --&gt; Moteur[Moteur sens 1]; Moteur --&gt; Attendre1[Attendre 3]; Attendre1 --&gt; MoteurStop[Moteur stop + ventilateur]; MoteurStop --&gt; Attendre2[Attendre 3]; Attendre2 --&gt; STOP[STOP]; STOP --&gt; Fin([Fin]);</pre> | <pre>graph TD; debut[<b>début</b>] --&gt; MoteurActive[sortie Moteur_1 activée]; MoteurActive --&gt; Attendre1[attendre pendant 3000 ms]; Attendre1 --&gt; MoteurDesactive[sortie Moteur_1 désactivée]; MoteurDesactive --&gt; VentilateurActive[sortie Ventilateur activée]; VentilateurActive --&gt; Attendre2[attendre pendant 3000 ms]; Attendre2 --&gt; VentilateurDesactive[sortie Ventilateur désactivée];</pre> |
| Fichier organigramme PE6 :<br>MS_N1_A3_Organigramme.plf  | Fichier Blockly : MS_N1_A3.xml  |

# Exercice niveau 1 - A.4 : Utilisation d'un capteur fin de course avec une boucle tant que

**Objectif :** Activer un moteur jusqu'à l'arrivée sur un capteur fin de course. Partir ensuite dans l'autre sens jusqu'à l'autre capteur fin de course. Cela indéfiniment.

**Notions abordées :** utilisation d'une boucle tant que et d'entrées fin de course.

**Instructions utilisées :**



**Correction :**

| Organigramme   | Blocs   |
|--|---|
| <pre> graph TD     Start([Start]) --&gt; M1[Moteur sens 1]     M1 --&gt; D1{FDC Deplié activé?}     D1 -- Non --&gt; M1     D1 -- Oui --&gt; M2[Moteur sens 2]     M2 --&gt; D2{FDC Replié activé?}     D2 -- Non --&gt; M2     D2 -- Oui --&gt; M1     </pre> | <pre> début répéter indéfiniment faire   tant que entrée FDC_Deplié est désactivée   faire     sortie Moteur_1 activée   sortie Moteur_1 désactivée   tant que entrée FDC_Replié est désactivée   faire     sortie Moteur_2 activée   sortie Moteur_2 désactivée   </pre> |
| <p>Fichier organigramme PE6 :<br/>MS_N1_A4_Organigramme.plf</p>  | <p>Fichier Blockly : MS_N1_A4.xml</p>   |

# Exercice niveau 1 - B.1 : Utilisation d'un capteur de température

**Objectif :** Récupérer une valeur de température sur un capteur et la placer dans une variable

**Notions abordées :** utilisation d'une instruction de mesure

**Instructions utilisées :**



**Correction :**

| Organigramme   | Blocs   |
|--|---|
| <pre>graph TD; Start([Start]) --&gt; LireTemp[/Lire Temp/]; LireTemp --&gt; Debug[/Debug/]; Debug --&gt; LireTemp;</pre> | <pre>graph TD; debut[debut] --&gt; repeter[répéter indéfiniment]; repeter --&gt; faire[faire]; faire --&gt; debug[debug]; debug --&gt; lire[lire température en Capteur_temp et stocker dans varA];</pre> |
| Fichier organigramme PE6 : MS_N1_B1_Organigramme.plf   | Fichier Blockly : MS_N1_B1.xml  |

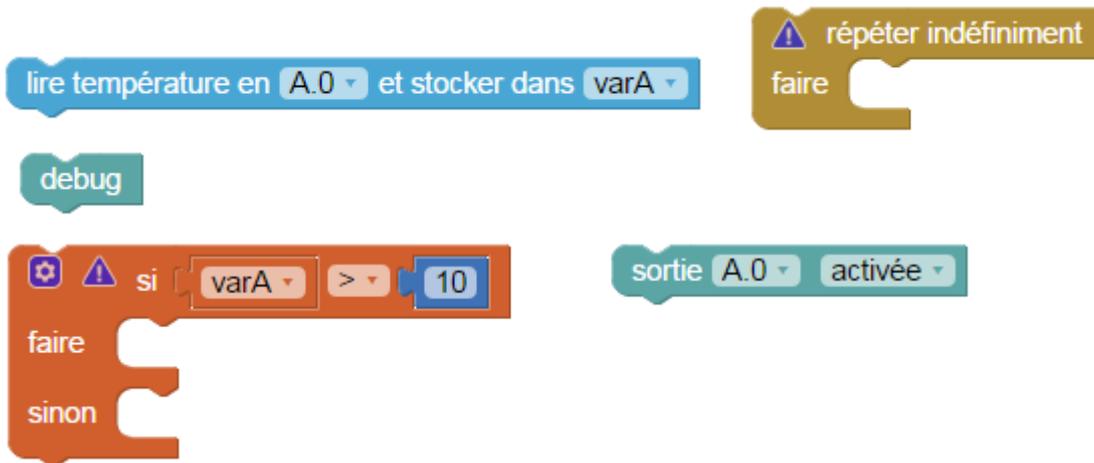
**Remarque :** La fonction  permet de récupérer chaque seconde la valeur et de la mettre dans notre variable.

# Exercice niveau 1 - B.2 : Utilisation d'une boucle conditionnelle Si

**Objectif :** Activer le ventilateur lorsque la température est supérieure à 25°C

**Notions abordées :** utilisation d'une boucle si en fonction d'une valeur

**Instructions utilisées :**



**Correction :**

| Organigramme  | Blocs                                 |
|---|---------------------------------------|
| <pre> graph TD     Start([Start]) --&gt; LireTemp[Lire Temp]     LireTemp --&gt; Debug[Debug]     Debug --&gt; Decision{varA &lt; 25}     Decision -- Oui --&gt; Desactiver[Désactiver ventilateur]     Decision -- Non --&gt; Activer[Activer ventilateur]     Desactiver --&gt; LireTemp     Activer --&gt; LireTemp     </pre> |                                       |
| <p>Fichier organigramme PE6 :<br/>MS_N1_B2_Organigramme.plf</p>   | <p>Fichier Blockly : MS_N1_B2.xml</p> |

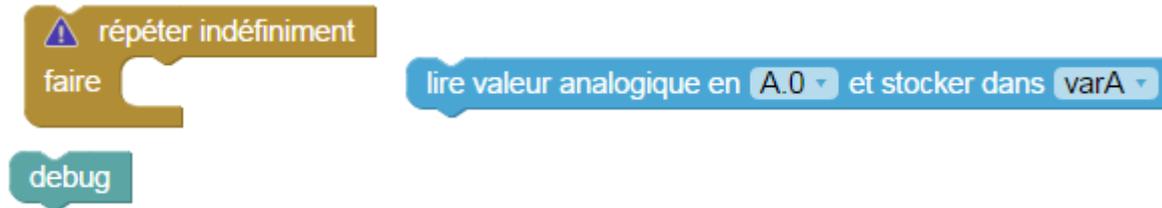
**Remarque :** Pour tester la fonctionnalité du programme, vous pouvez chauffer le capteur de température (En haut à gauche) ou alors diminuer la limite de varA dans le programme (Par exemple à 10°C).

# Exercice niveau 1 - B.3 : Utilisation du capteur d'humidité

**Objectif :** Relever une valeur analogique sur le capteur d'humidité et l'implémenter dans une variable

**Notions abordées :** Lecture d'une valeur analogique

**Instructions utilisées :**



**Correction :**

| Organigramme   | Blocs   |
|--|---|
| <pre>graph TD; Start([Start]) --&gt; LireHumidité[/Lire humidité/]; LireHumidité --&gt; Debug[/Debug/]; Debug --&gt; LireHumidité;</pre> | <pre>graph TD; debut[debut] --&gt; loop[répéter indéfiniment]; loop --&gt; faire[faire]; faire --&gt; debug[debug]; debug --&gt; read[lire valeur analogique en Capteur_humidite et stocker dans varA]; read --&gt; loop;</pre> |
| Fichier organigramme PE6 : MS_N1_B3_Organigramme.plf   | Fichier Blockly : MS_N1_B3.xml  |

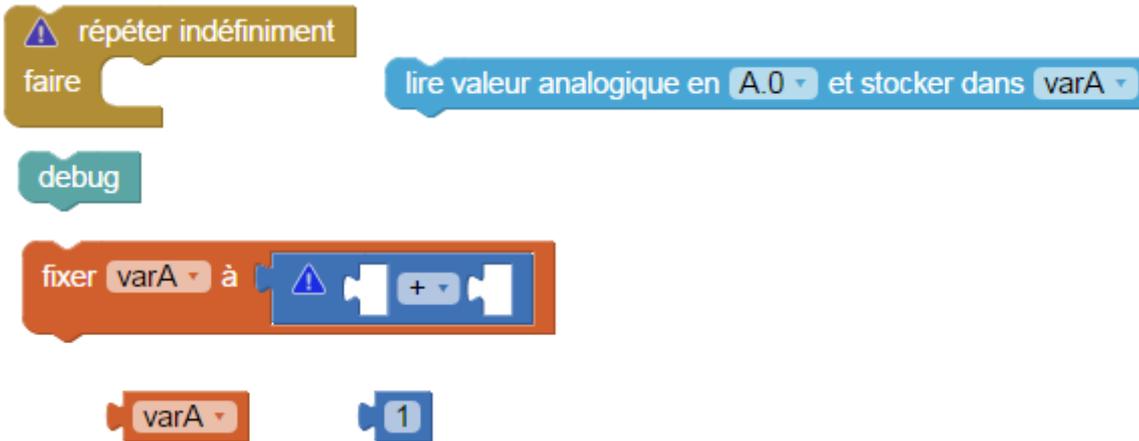
**Remarque :** la valeur analogique donnée par ce capteur varie entre 0 et 255 en fonction d'une humidité de 0 à 100%. Nous avons une tension maximale à 4V pour une humidité de 100%, soit 1V donne 25% d'humidité. 4V donne également 255 en sortie de capteur, nous avons donc une valeur analogique de 64 pour 1V. Pour récupérer l'humidité en %, il faut donc multiplier par 25 la valeur analogique puis la diviser par 64.

# Exercice niveau 1 - B.4 : Opérations sur une variable

**Objectif :** Multiplier la variable analogique par 25 puis la diviser par 64 afin d'avoir l'humidité en %

**Notions abordées :** Opérations sur une variable

**Instructions utilisées :**



**Correction :**

| Organigramme   | Blocs                                 |
|--|---------------------------------------|
| <pre> graph TD     Start([Start]) --&gt; Debug[Debug]     Debug --&gt; Lire[Lire humidité]     Lire --&gt; Calc["varB = varA * 25 / 64"]     Calc --&gt; Lire     </pre> |                                       |
| <p>Fichier organigramme PE6 : MS_N1_B4_Organigramme.plf</p>  | <p>Fichier Blockly : MS_N1_B4.xml</p> |

**Remarque :** Avec ce calcul, notre variable nous donnera directement l'humidité en %  
 Il est nécessaire de commencer par la multiplication car les divisions ne donnent que des nombres entiers.

# Exercice niveau 1 - C.1 : Utilisation de la pompe

**Description :** La pompe se situe dans le réservoir situé en bas à gauche à l'arrière de la serre. Celle-ci fait passer l'eau du réservoir dans un tuyau jusqu'à un robinet.

**Objectifs :** Allumer la pompe pendant une seconde toutes les 2 secondes. Répéter l'action 10 fois.

**Notions abordées :** Boucle de répétition gérée par une variable

**Instructions utilisées :**



**Correction :**

| Organigramme  | Blocs  |
|---|--|
| <pre> graph TD     Start([Start]) --&gt; Loop{Boucle Jusqu'à...}     Loop --&gt; Activer[Activer pompe]     Activer --&gt; Attendre1([Attendre 1])     Attendre1 --&gt; Désactiver[Désactiver pompe]     Désactiver --&gt; Attendre2([Attendre 2])     Attendre2 --&gt; FinBoucle{Fin de Boucle}     FinBoucle --&gt; Fin([Fin])         </pre> | <pre> graph TD     debut[debut] --&gt; repeter[ répéter ]     repeter --&gt; sortie1[ sortie Pompe activée ]     repeter --&gt; attendre1[ attendre pendant 1000 ms ]     repeter --&gt; sortie2[ sortie Pompe désactivée ]     repeter --&gt; attendre2[ attendre pendant 2000 ms ]     repeter --&gt; incr[ incrémenter varA de 1 ]     repeter --&gt; jusqu_a[ jusqu'à varA = 10 ]     jusqu_a --&gt; fin[ ]         </pre> |
| <p>Fichier organigramme PE6 : MS_N1_C1_Organigramme.plf</p>   | <p>Fichier Blockly : MS_N1_C1.xml</p>  |

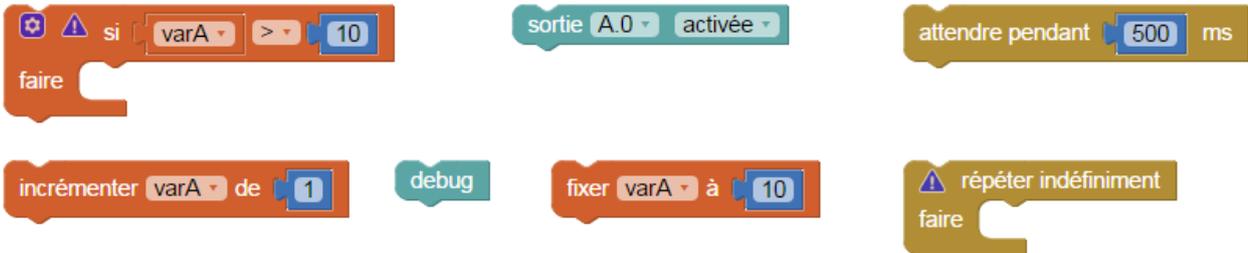
## Exercice niveau 1 - C.2 : Arrosage automatique

**Problème :** La commande d'attente ne permet d'aller que jusqu'à 65 secondes. Une solution pour dépasser ce temps, autre qu'une succession d'attentes, est possible grâce aux incréments.

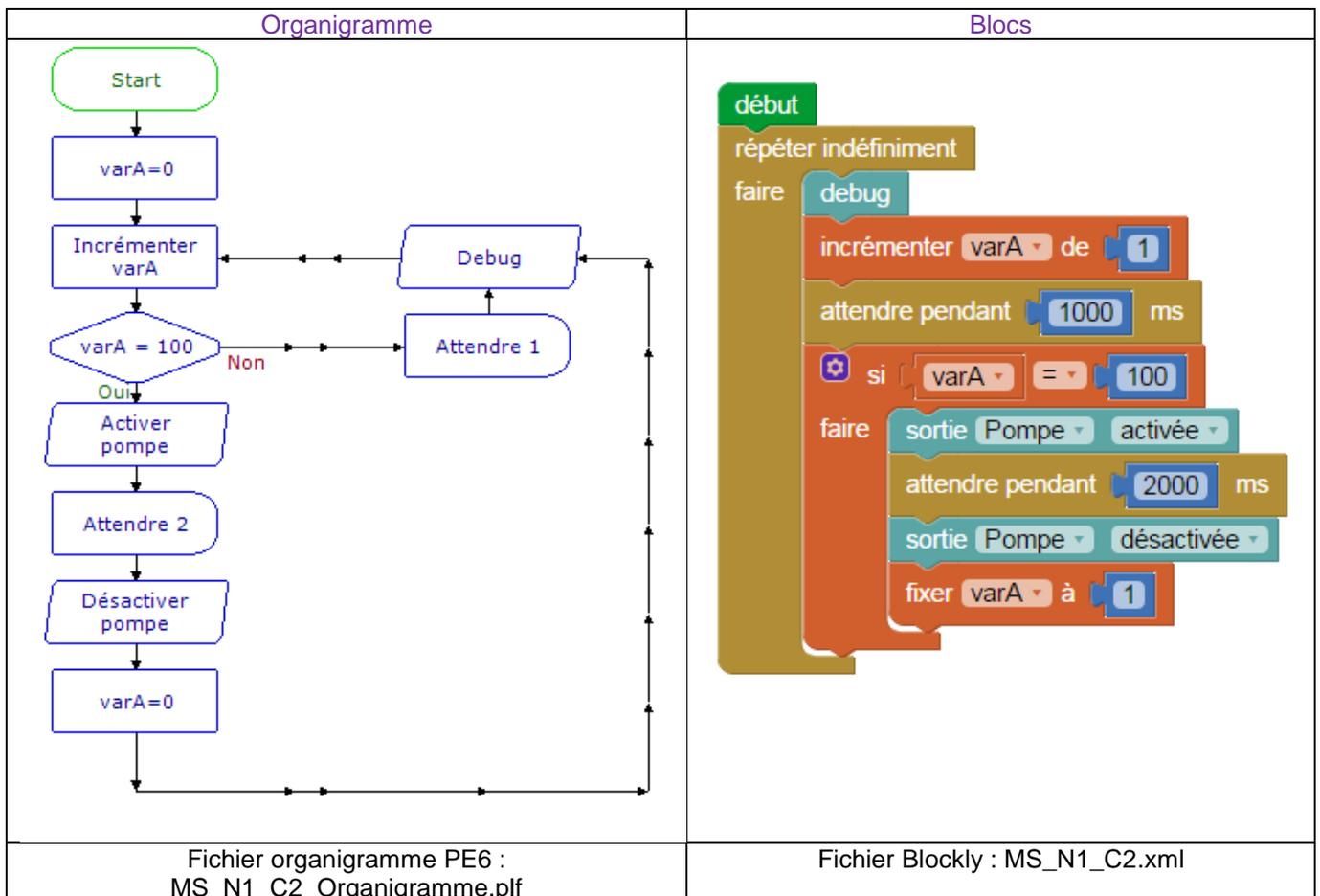
**Objectifs :** Toutes les 100 secondes, activer la pompe pendant 2 secondes.

**Notions abordées :** Incrémentation et gestion de variable.

**Instructions utilisées :**



**Correction :**

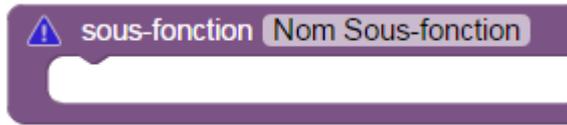


## Exercice niveau 1 - D.1 : Utilisation des sous-fonctions

**Objectifs :** Créer des sous-fonctions chargées d'ouvrir ou de fermer la fenêtre. Les répéter indéfiniment

**Notions abordées :** Sous fonctions.

**Instructions utilisées :**



**Correction :**

Blocs

Le programme est divisé en trois parties principales :

- début** (bloc vert) :
  - répéter indéfiniment (bloc orange)
  - faire (bloc brun) :
    - appeler sous-fonction Ouvrir (bloc violette)
    - appeler sous-fonction Fermer (bloc violette)
- sous-fonction Ouvrir** (bloc violette) :
  - répéter (bloc orange) :
    - sortie Moteur\_1 activée (bloc bleu)
    - jusqu'à entrée FDC\_Deplie est activée (bloc orange)
    - sortie Moteur\_1 désactivée (bloc bleu)
- sous-fonction Fermer** (bloc violette) :
  - répéter (bloc orange) :
    - sortie Moteur\_2 activée (bloc bleu)
    - jusqu'à entrée FDC\_Replie est activée (bloc orange)
    - sortie Moteur\_2 désactivée (bloc bleu)

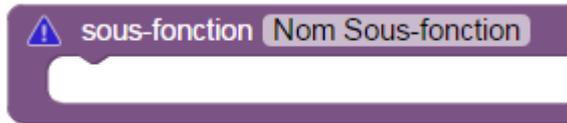
Fichier Blockly : MS\_N1\_D1.xml

## Exercice niveau 1 - D.2 : Sous fonctions pour les capteurs

**Objectifs :** Créer des sous-fonctions chargées de récupérer les valeurs des capteurs. Les répéter indéfiniment

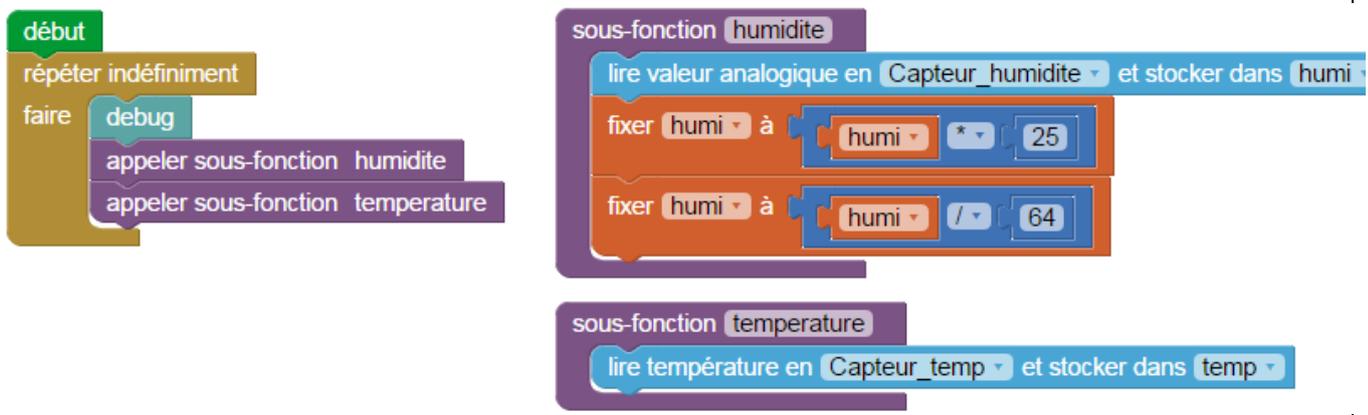
**Notions abordées :** Sous fonctions

**Instructions utilisées :**



**Correction :**

Blocs



```
graph TD
    Start[debut] --> Loop[repete indefiniment]
    Loop --> Do[faire]
    Do --> Debug[debug]
    Do --> CallHum[appeler sous-fonction humidite]
    Do --> CallTemp[appeler sous-fonction temperature]
    
    subgraph "sous-fonction humidite"
        ReadHum[lire valeur analogique en Capteur_humidite et stocker dans humi]
        FixHum1[fixer humi à humi * 25]
        FixHum2[fixer humi à humi / 64]
    end
    
    subgraph "sous-fonction temperature"
        ReadTemp[lire temperature en Capteur_temp et stocker dans temp]
    end
```

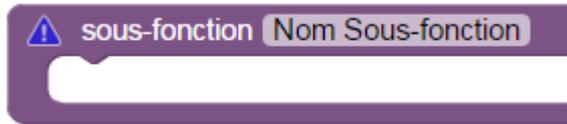
Fichier Blockly : MS\_N1\_D2.xml

## Exercice niveau 1 - D.3 : Conditions dans les sous-fonctions

**Objectifs :** Reprendre l'exercice précédent. Faire tourner le ventilateur si la température dépasse 25°C, l'arrêter si la température descend à 20°C. Il ne doit y avoir qu'une boucle répéter, un bloc debug et des sous-fonctions sous le bloc début.

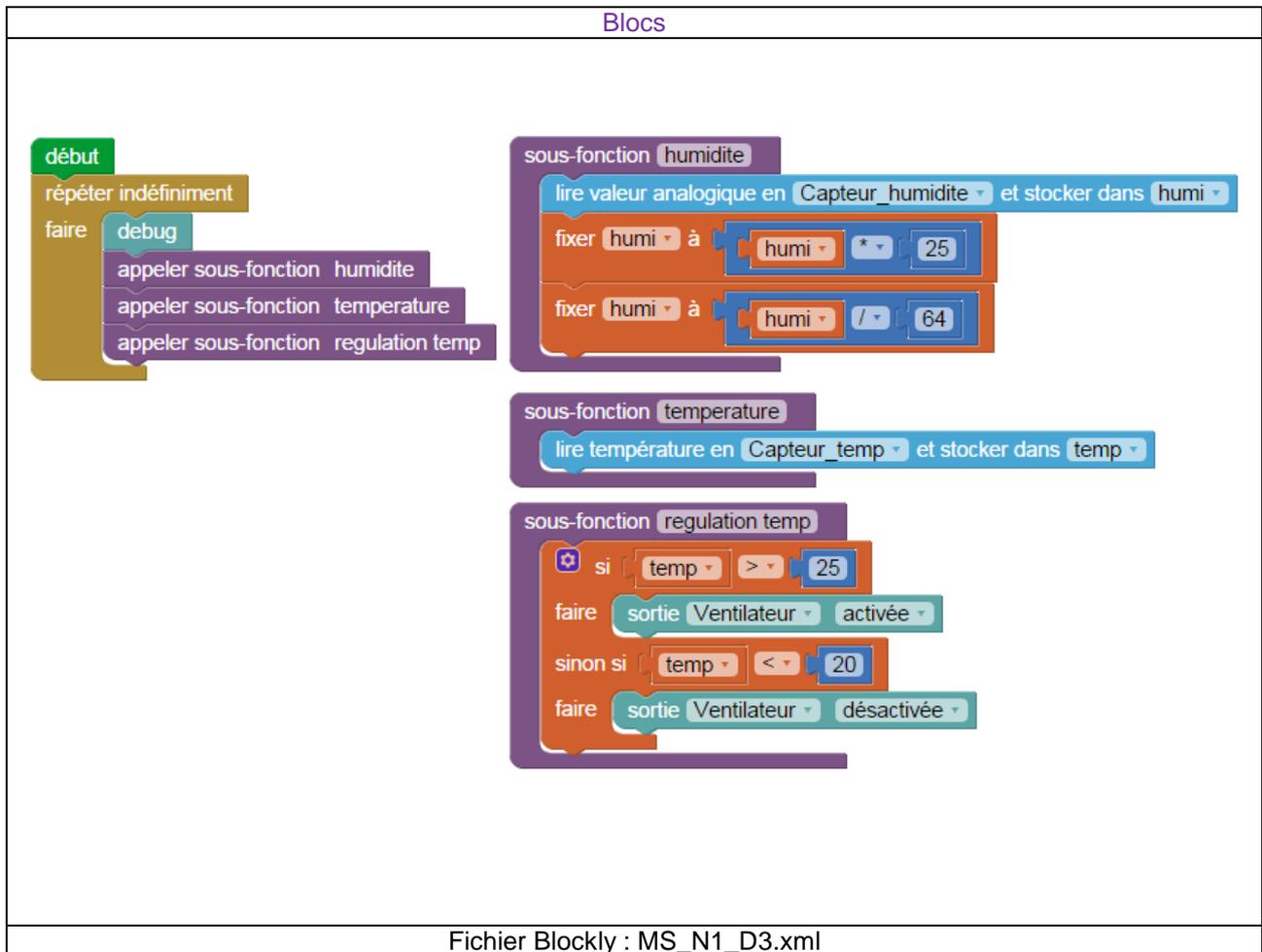
**Notions abordées :** Sous fonctions

**Instructions utilisées :**



**Correction :**

Blocs



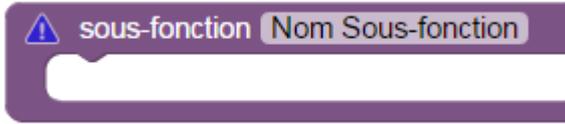
Fichier Blockly : MS\_N1\_D3.xml

# Exercice niveau 1 - D.4 : Serre automatique

**Objectifs :** Reprendre l'exercice précédent, y ajouter une ouverture de la fenêtre pour une humidité supérieure à 70% et une fermeture pour une humidité inférieure à 50%.  
Ajouter également un arrosage automatique de 10 secondes toutes les 900 secondes.

**Notions abordées :** Sous fonctions

**Instructions utilisées :**



**Correction :**

Blocs

Fichier Blockly : MS\_N1\_D4.xml

Détails de la correction Blockly :

- début**
  - répéter indéfiniment
    - faire
      - debug
      - appeler sous-fonction humidite
      - appeler sous-fonction temperature
      - appeler sous-fonction regulation temp
      - appeler sous-fonction regulation humi
      - appeler sous-fonction arrosage automatique

- sous-fonction arrosage automatique**
- incrémenter varosage de 1
- si varosage = 900
  - faire
    - sortie Pompe activée
    - attendre pendant 10000 ms
    - sortie Pompe désactivée
    - fixer varosage à 1
- sous-fonction Ouvrir**
- répéter sortie Moteur\_1 activée
- jusqu'à entrée FDC\_Deplie est activée
- sortie Moteur\_1 désactivée
- sous-fonction humidite**
- lire valeur analogique en Capteur\_humidite et stocker dans humi
- fixer humi à  $humi * 25$
- fixer humi à  $humi / 64$
- sous-fonction temperature**
- lire température en Capteur\_temp et stocker dans temp
- sous-fonction regulation temp**
- si temp > 25
  - faire sortie Ventilateur activée
- sinon si temp < 20
  - faire sortie Ventilateur désactivée
- sous-fonction regulation humi**
- si humi > 70
  - faire appeler sous-fonction Ouvrir
- sinon si humi < 50
  - faire appeler sous-fonction Fermer
- sous-fonction Fermer**
- répéter sortie Moteur\_2 activée
- jusqu'à entrée FDC\_Replie est activée
- sortie Moteur\_2 désactivée

# Programmation version de base niveau 2

Objectifs : Utiliser les modules plus complexes : afficheur OLED, sonde hygrométrique, brumisateurs...

Le niveau 2 n'intègre pas de nouvelles notions de programmation mais de nouveaux blocs permettant d'utiliser les modules options.

| Nom du fichier                                   | Description  | Objectif  |
|--|--|---|
| <b>Niveau 1 A - Fichier modèle : MS_BASE.xml</b> |  |   |
| MS_N2_A1.xml                                     | Déclencher un événement sur l'appui d'un bouton poussoir.                                  | Fonctionnalité matérielle abordée :<br>- Bouton poussoir          |
| MS_N2_A2.xml                                     | Descendre ou monter la fenêtre en appuyant sur un bouton poussoir.                         |   |
| <b>Niveau 1 B - Fichier modèle : MS_BASE.xml</b> |  |   |
| MS_N2_B1.xml                                     | Afficher un message sur un afficheur OLED.   | Fonctionnalité matérielle abordée :<br>- Afficheur OLED           |
| MS_N2_B2.xml                                     | Afficher une variable.   |   |
| MS_N2_B3.xml                                     | Afficher plusieurs variables sur plusieurs lignes.   | Notions de programmation abordées :<br>Commandes BASIC            |
| MS_N2_B4.xml                                     | Afficher l'état d'une sortie.  |   |
| <b>Niveau 1 C - Fichier modèle : MS_BASE.xml</b> |  |   |
| MS_N2_C1.xml                                     | Lire l'humidité d'un élément dans lequel plonger la sonde.                                 | Notions de programmation abordées :<br>lire une entrée analogique |
| MS_N2_C2.xml                                     | Activer ou désactiver l'arrosage en fonction de l'humidité.                                |   |
| MS_N2_C3.xml                                     | Exercice N2_C2 avec des sous fonctions.  |   |
| <b>Niveau 1 D - Fichier modèle : MS_BASE.xml</b> |  |   |
| MS_N2_D1.xml                                     | Activer le plateau chauffant lorsque la température est trop basse.                        | Fonctionnalité matérielle abordée :<br>- Plateau chauffant        |
| MS_N2_D2.xml                                     | Réguler la température à l'aide du plateau et de la fenêtre.                               |   |
| <b>Niveau 1 E - Fichier modèle : MS_BASE.xml</b> |  |   |
| MS_N2_E1.xml                                     | Activer le brumisateurs pendant 6 secondes.  | Fonctionnalité matérielle abordée :<br>- Brumisateurs             |
| MS_N2_E2.xml                                     | Reprendre l'exercice MS_N1_B4.xml et activer le brumisateurs pour une humidité trop basse. |   |
| MS_N2_E3.xml                                     | Régulation de l'humidité.  |   |

## Exercice niveau 2 - A.1 : Bouton-poussoir

**Objectifs :** Déclencher un événement sur l'appui d'un bouton poussoir (par exemple allumer le ventilateur)

**Notions abordées :** Utilisation d'une condition avec un élément déclencheur

**Instructions utilisées :**



**Correction :**

Blocs

The screenshot shows a Blockly workspace with the following script:

- début** (green)
- répéter indéfiniment** (brown)
- faire** (blue) - **si entrée Bouton\_Poussoir est activée** (blue)
  - faire** (blue) - **sortie Ventilateur activée** (green)
  - sinon** (blue) - **sortie Ventilateur désactivée** (green)

Fichier Blockly : MS\_N2\_A1.xml

## Exercice niveau 2 - A.2 : Déclencher des moteurs grâce au bouton poussoir

**Objectifs :** Lors d'un appui sur le bouton poussoir, les moteurs vont faire monter ou descendre la fenêtre en fonction du capteur fin de course initial

**Notions abordées :** Utilisation d'une condition avec un élément déclencheur

**Instructions utilisées :**



**Correction :**

Blocs

Le script principal est :

- début
- répéter indéfiniment
- faire
- si entrée Bouton\_Poussoir est activée
- faire
- si entrée FDC\_Replie est activée
- faire
- appeler sous-fonction Ouvrir
- sinon
- appeler sous-fonction Fermer

Les sous-fonctions sont :

- sous-fonction Ouvrir
- répéter
- sortie Moteur\_1 activée
- jusqu'à entrée FDC\_Deplie est activée
- appeler sous-fonction arrêt

- sous-fonction Fermer
- répéter
- sortie Moteur\_2 activée
- jusqu'à entrée FDC\_Replie est activée
- appeler sous-fonction arrêt

- sous-fonction arrêt
- sortie Moteur\_1 désactivée
- sortie Moteur\_2 désactivée

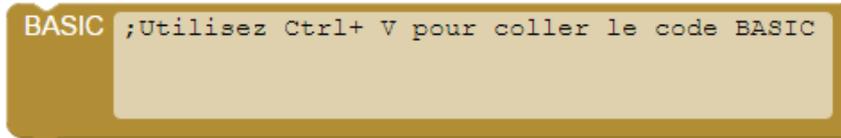
Fichier Blockly : MS\_N2\_A2.xml

## Exercice niveau 2 - B.1 : Afficher un message sur l'afficheur OLED

**Objectifs :** Afficher un premier message sur un afficheur OLED  
Ne pas utiliser les blocs « afficher sur lcd » pour un afficheur 4 lignes.

**Notions abordées :** Utilisation d'une commande BASIC

**Instructions utilisées :**



**Correction :**

Blocs

```
BASIC serout B.1, N2400, ( 254, 0x80 )
serout B.1, N2400, ( "Hello world" )
```

Fichier Blockly : MS\_N2\_B1.xml

**Important :** Pour cet exercice, il est préférable de donner aux élèves le code BASIC à rentrer dans la fonction.

Celui-ci étant :

```
serout B.1, N2400, ( 254, 0x80 )
serout B.1, N2400, ( "Hello world" )
```

La première ligne indique l'emplacement du message, avec 0x80 donnant la première ligne.

La seconde ligne donne le message à mettre, par exemple ici « Hello world », mais vous pouvez le changer à votre guise.

Pour la suite, le code pour se positionner sur la 2ème ligne sera 0xC0, 0x94 pour la troisième et 0xD4 pour la quatrième.

Il est également préférable d'effacer ce qui est sur l'afficheur dans un premier temps car il réécrit des données sur certaines déjà existantes. Donc si il ne réécrit rien sur un message qui était la auparavant, le message restera.

## Exercice niveau 2 - B.2 : Afficher une variable sur l'afficheur OLED

**Objectifs :** Afficher une température sur l'afficheur LCD

**Notions abordées :** Utilisation d'une autre commande BASIC

**Instructions utilisées :**

Le diagramme de blocs Blockly comprend les éléments suivants :

- Un bloc "répéter indéfiniment" (orange) contenant un bloc "faire" (orange).
- Un bloc "lire température en Capteur\_temp et stocker dans Temp" (bleu).
- Un bloc "envoyer sur LCD effacer en Afficheur\_OLED" (vert).
- Un bloc "debug" (bleu).
- Un bloc "BASIC" (orange) contenant le code :

```
;Utilisez Ctrl+ V pour coller le code BASIC
```

**Correction :**

Le programme de correction est structuré comme suit :

- Bloc "début" (vert).
- Bloc "envoyer sur LCD effacer en Afficheur\_OLED" (vert).
- Bloc "répéter indéfiniment" (orange) contenant un bloc "faire" (orange).
- À l'intérieur du bloc "faire" :
  - Bloc "debug" (bleu).
  - Bloc "lire température en Capteur\_temp et stocker dans Temp" (bleu).
  - Bloc "BASIC" (orange) contenant le code :

```
serout B.1, N2400, ( 254, 0x80 )  
serout B.1, N2400, ( "Temp : ", #Temp, $D2, "C" )
```

Fichier Blockly : MS\_N2\_B2.xml

**Important :** Pour cet exercice, il est préférable de donner aux élèves le code BASIC à rentrer dans la fonction.

Celui-ci étant :

```
serout B.1, N2400, ( 254, 0x80 )
```

```
serout B.1, N2400, ( "Temp : ", #Temp, $D2, "C" )
```

Les instructions entre guillemets sont du texte à afficher. Une instruction avec un # et le nom de la variable renvoie la valeur de la variable à l'écran. \$D2 correspond au code ASCII du °, car certains caractères nécessitent un certain code pour être affichées via le bloc BASIC.

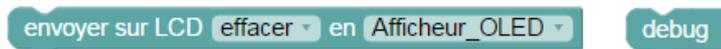
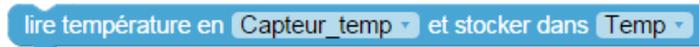
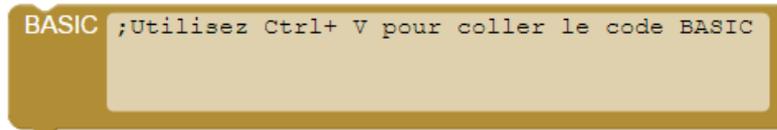
Un espace dans le texte effectuera un espace sur l'écran.

## Exercice niveau 2 - B.3 : Afficher plusieurs variables sur l'afficheur OLED

**Objectifs :** Afficher la température et le taux d'humidité sur l'afficheur LCD

**Notions abordées :** Utilisation d'une commande BASIC

**Instructions utilisées :**



**Correction :**

Blocs

```

début
envoyer sur LCD effacer en Afficheur_OLED
répéter indéfiniment
faire
  debug
  lire température en Capteur_temp et stocker dans Temp
  lire valeur analogique en Capteur_humidite et stocker dans Hum
  fixer Hum à Hum * 25
  fixer Hum à Hum / 64
  BASIC serout B.1, N2400, ( 254, 0x80 )
  BASIC serout B.1, N2400, ( "Temp : ", #Temp, $D2, "C" )
  BASIC serout B.1, N2400, ( 254, 0xC0 )
  BASIC serout B.1, N2400, ( "Hum : ", #Hum, "%" )

```

Fichier Blockly : MS\_N2\_B3.xml

## Exercice niveau 2 - B.4 : Afficher l'état d'une sortie sur l'afficheur OLED

**Objectifs :** Afficher si oui ou non une sortie est activée (exemple : ventilateur), vous pouvez par exemple allumer et éteindre le ventilateur toutes les 5 secondes en changeant son état via le bloc Basculer

**Notions abordées :** Utilisation d'une commande BASIC

**Instructions utilisées :**

basculer B.0

**Correction :**

Blocs

```
Blockly code description:  
- Start: début  
- Action: envoyer sur LCD effacer en Afficheur_OLED  
- Loop: répéter indéfiniment  
  - Action: faire  
    - Action: basculer Ventilateur  
    - Condition: si entrée Ventilateur est activée  
      - Action: faire  
        - Action: envoyer sur LCD effacer en Afficheur_OLED  
        - BASIC serout B.1, N2400, ( 254, 0x80 )  
        - BASIC serout B.1, N2400, ( "Actif" )  
      - Action: sinon  
        - Action: envoyer sur LCD effacer en Afficheur_OLED  
        - BASIC serout B.1, N2400, ( 254, 0x80 )  
        - BASIC serout B.1, N2400, ( "Non actif" )  
    - Action: attendre pendant 5000 ms
```

Fichier Blockly : MS\_N2\_B4.xml

**Remarque :** Une ligne ne peut contenir que 20 caractères. Au-delà de cette limite, l'afficheur sautera une ligne. Il est donc préférable de mettre moins de 20 caractères.

## Exercice niveau 2 - C.1 : Utiliser une sonde hygrométrique

**Description :** La sonde hygrométrique possède deux pattes qui pourront être introduites dans la terre pour mesurer un taux d'humidité. C'est un capteur qui fonctionne de la même façon que le capteur d'humidité, c'est-à-dire qu'il envoie des valeurs de 0 à 255 de 0% à 100%

**Objectifs :** Créer un programme semblable à celui de la sonde hygrométrique pour récupérer la valeur d'humidité en %

**Notions abordées :** Utilisation d'un second capteur analogique

**Instructions utilisées :**



**Correction :**

Blocs

début

répéter indéfiniment

faire

debug

lire valeur analogique en Sonde\_hygro et stocker dans varA

fixer varA à [varA \* 100]

fixer varA à [varA / 255]

Fichier Blockly : MS\_N2\_C1.xml

**Remarque :** Effectuez bien la multiplication avant la division

## Exercice niveau 2 - C.2 : Arroser en cas d'hygrométrie trop faible

**Objectifs :** établir un programme permettant de réguler l'arrosage avec une activation à 20% et une désactivation à 40%

**Notions abordées :** Utilisation d'un second capteur analogique

**Instructions utilisées :**



**Correction :**

Blocs

```

début
répéter indéfiniment
faire
  debug
  lire valeur analogique en Sonde_hygro et stocker dans varA
  fixer varA à varA * 100
  fixer varA à varA / 255
  si varA < 20
  faire
    sortie Pompe activée
  sinon si
    varA > 40
  faire
    sortie Pompe désactivée

```

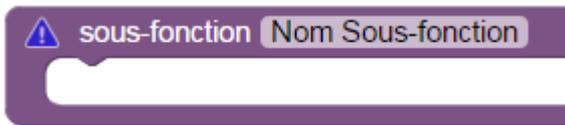
Fichier Blockly : MS\_N2\_C2.xml

## Exercice niveau 2 - C.3 : Arroser en cas d'hygrométrie trop faible (2)

**Objectifs :** Reprendre l'exercice précédent en n'utilisant que des sous-fonctions. Seul les blocs debug et répéter indéfiniment ainsi que des appels de sous-fonctions seront utilisés en dessous du bloc début.

**Notions abordées :** Utilisation d'un second capteur analogique

**Instructions utilisées :**



**Correction :**

Blocs

```
graph TD
    subgraph "début"
        A[répéter indéfiniment]
        subgraph "faire"
            B[debug]
            C[appeler sous-fonction arrosage]
            D[appeler sous-fonction sonde hygro]
        end
    end
    subgraph "sous-fonction sonde hygro"
        E[lire valeur analogique en Sonde_hygro et stocker dans varA]
        F[fixer varA à varA * 100]
        G[fixer varA à varA / 255]
    end
    subgraph "sous-fonction arrosage"
        H[si varA < 20]
        I[faire sortie Pompe activée]
        J[sinon si varA > 40]
        K[faire sortie Pompe désactivée]
    end
```

Fichier Blockly : MS\_N2\_C3.xml

## Exercice niveau 2 - D.1 : Utilisation du plateau chauffant

**Description :** Le plateau chauffant est la plaque métallique constituant le sol de la serre. Celui-ci possède une résistance interne qui émettra de la chaleur lorsque de l'électricité la traverse.

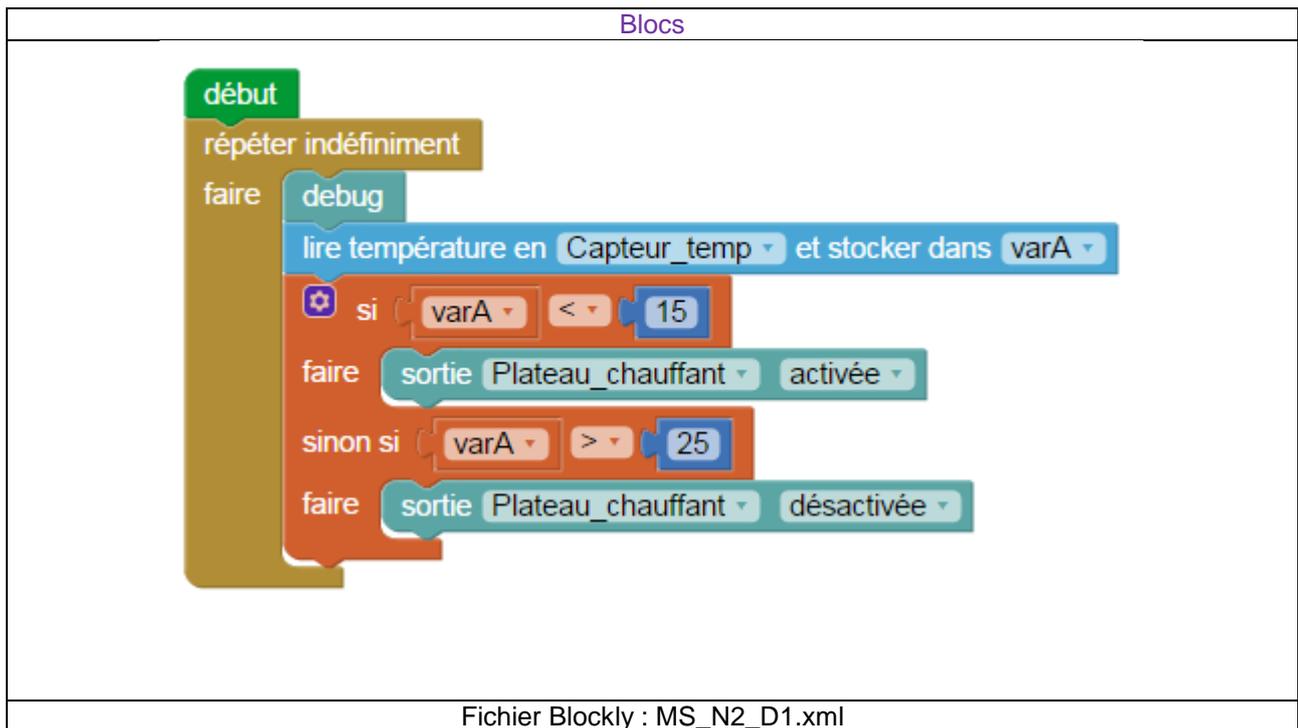
**Objectifs :** Créer un programme qui active le plateau chauffant en dessous de 15°C et le désactive au-dessus de 25°C.

**Notions abordées :** Utilisation d'une sortie en fonction d'une variable

**Instructions utilisées :**



**Correction :**



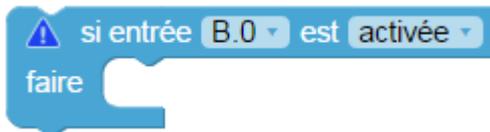
**Remarque :** vous pouvez modifier les seuils pour vérifier le bon fonctionnement du programme

## Exercice niveau 2 - D.2 : Régulation de température

**Objectifs :** Reprendre le programme précédent. Lorsque le plateau chauffant est activé, la fenêtre doit être fermée, lorsqu'on le désactive, on ouvre la fenêtre

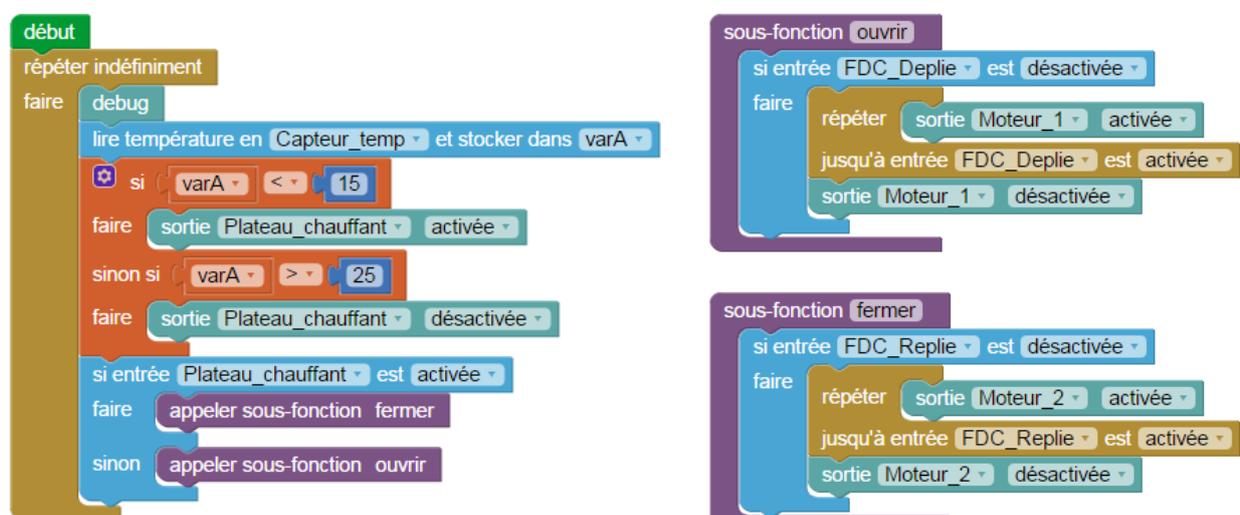
**Notions abordées :** Utilisation de conditions

**Instructions utilisées :**



**Correction :**

Blocs



Fichier Blockly : MS\_N2\_D2.xml

## Exercice niveau 2 - E.1 : Utilisation d'un brumisateur

**Description :** Le brumisateur permet d'augmenter le taux d'humidité dans la sphère. Pour cela il faut d'abord remplir le réservoir d'eau, puis appuyer trois fois sur le bouton poussoir lorsqu'il est activé pour créer de la « brume ».

**Objectifs :** Activer le brumisateur pendant 6 secondes

**Notions abordées :** Utilisation d'une sortie

**Instructions utilisées :**



**Correction :**

Blocs

Un script Blockly complet dans un cadre. Il commence par un bloc 'début' vert. Ensuite, il y a un bloc 'sortie Brumisateur activée' vert clair. Cela est suivi d'un bloc 'attendre pendant 6000 ms' orange. Le script se termine par un bloc 'sortie Brumisateur désactivée' vert clair.

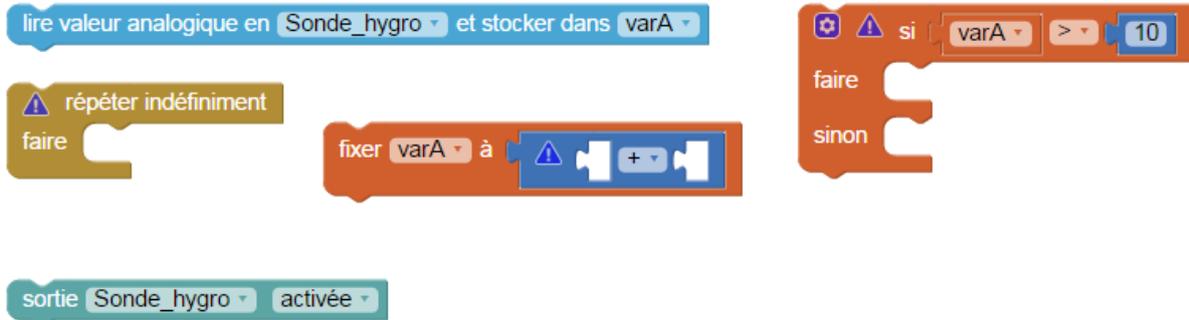
Fichier Blockly : MS\_N2\_E1.xml

## Exercice niveau 2 - E.2 : Régulation de l'humidité

**Objectifs :** Reprendre l'exercice Niveau 1 B.4 (MS\_N1\_B4.xml), lorsque le taux d'humidité passe en dessous de 50%, activer le brumisateurs jusqu'à une humidité de 60%

**Notions abordées :** Utilisation d'une condition dépendant d'une variable

**Instructions utilisées :**



**Correction :**

Blocs

Le programme de correction est structuré comme suit :

- début
- répéter indéfiniment
  - faire
    - debug
    - lire valeur analogique en Capteur\_humidite et stocker dans varA
    - fixer varA à  $varA * 25$
    - fixer varA à  $varA / 64$
    - si  $varA < 50$ 
      - faire sortie Brumisateur activée
    - sinon si  $varA > 60$ 
      - faire sortie Brumisateur désactivée

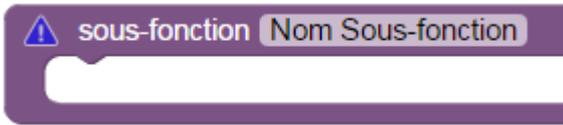
Fichier Blockly : MS\_N2\_E2.xml

## Exercice niveau 2 - E.3 : Régulation de l'humidité (2)

**Objectifs :** Reprendre l'exercice précédent et le refaire avec des sous-fonctions. Les blocs sous début ne doivent être que des sous-fonctions, un bloc de répétition, et un bloc debug.

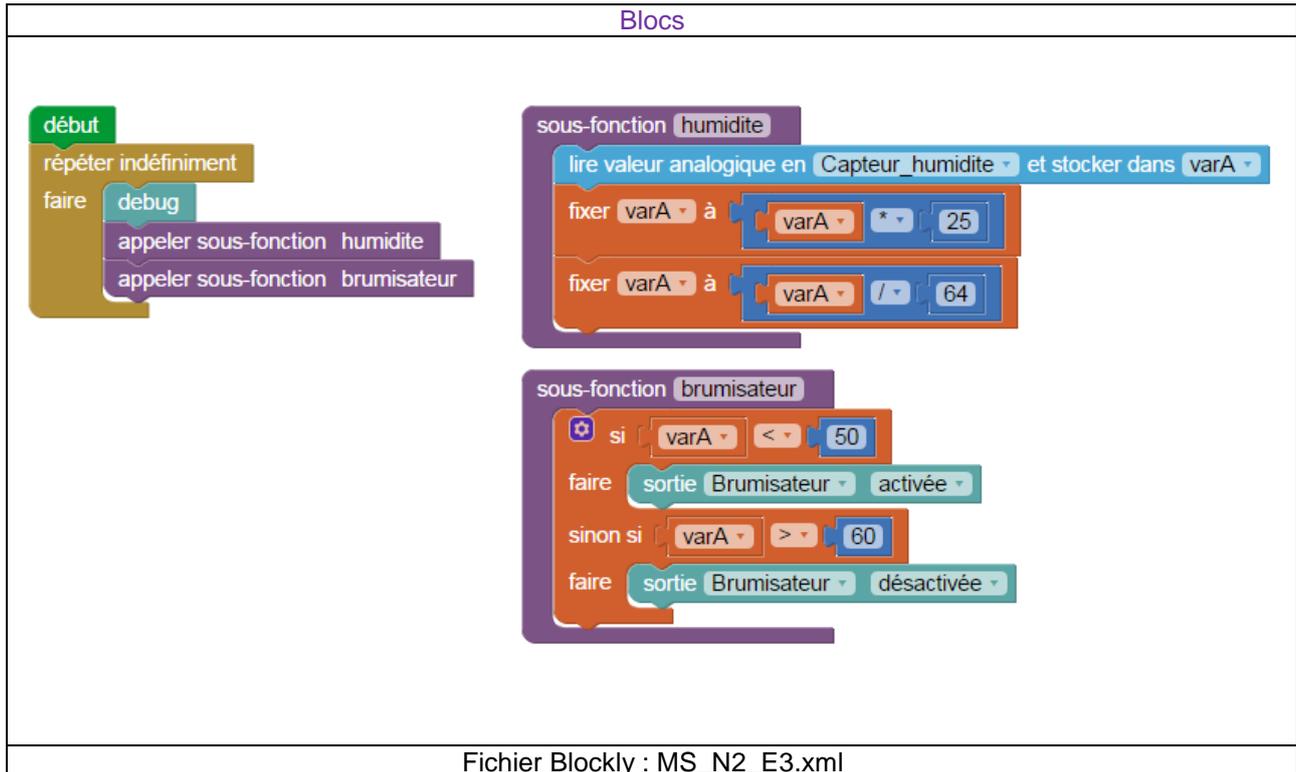
**Notions abordées :** Utilisation d'une condition dépendant d'une variable

**Instructions utilisées :**



**Correction :**

Blocs



```
graph TD
    subgraph "début"
        A[debug]
        B[appeler sous-fonction humidite]
        C[appeler sous-fonction brumisateur]
    end
    subgraph "humidite"
        D[lire valeur analogique en Capteur_humidite et stocker dans varA]
        E[fixer varA à varA * 25]
        F[fixer varA à varA / 64]
    end
    subgraph "brumisateur"
        G[si varA < 50]
        H[faire sortie Brumisateur activée]
        I[sinon si varA > 60]
        J[faire sortie Brumisateur désactivée]
    end
```

Fichier Blockly : MS\_N2\_E3.xml

# Programmation niveau 3

---

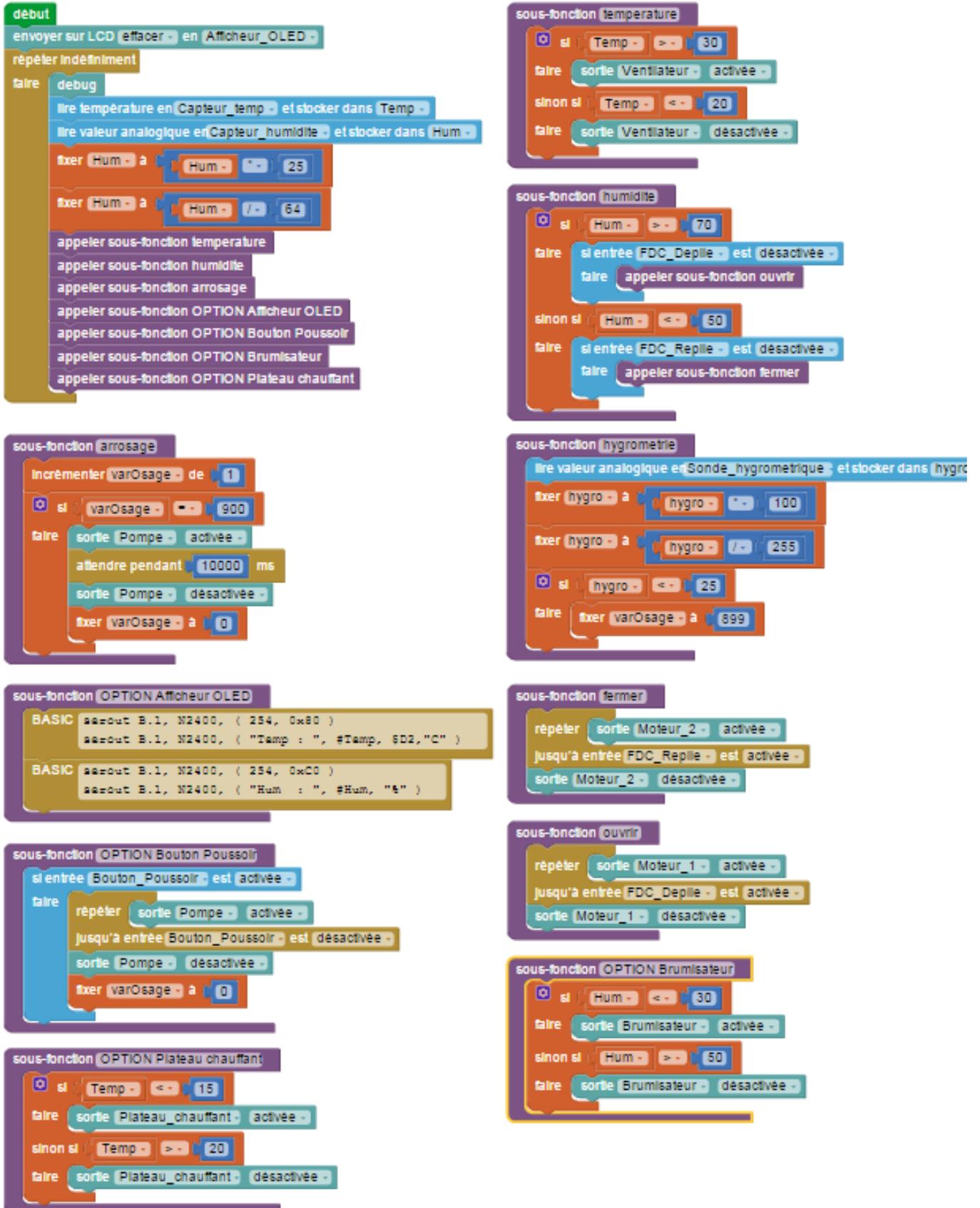
## Exercice niveau 3 – A.1 : Programme final, automatisation de la serre

**Objectifs :** En vous servant de tous les exercices et en fonction des options, faire :

- Mesurer et pouvoir lire les valeurs d'humidité et de température sur le programme
- Activer le ventilateur pour une température supérieure à 30°C, l'éteindre lorsque la température est inférieure à 20°C
- Ouvrir la serre lorsque l'humidité dépasse 70%, la fermer lorsqu'elle passe en dessous de 50%
- Arroser la serre pendant 10 secondes toutes les 900 secondes

**Avec options :**

- Le bouton poussoir permet d'activer manuellement l'arrosage. Après un appui sur le bouton, l'arrosage automatique se fera après 900 secondes.
- Sur l'afficheur OLED, mettre les valeurs de température et d'humidité, qui devront suivre celles du programme.
- Si l'hygrométrie passe en dessous de 25%, activer l'arrosage automatique (On peut par exemple augmenter la variable incrémentée dans l'arrosage automatique).
- Activer le plateau chauffant lorsque la température est en dessous de 15°C, le désactiver lorsqu'elle dépasse les 20°C
- Si l'humidité passe en dessous de 30%, activer le brumisateur. Le désactiver lorsque l'humidité atteint les 50%.



Fichier Blockly : MS\_N3\_A1.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.

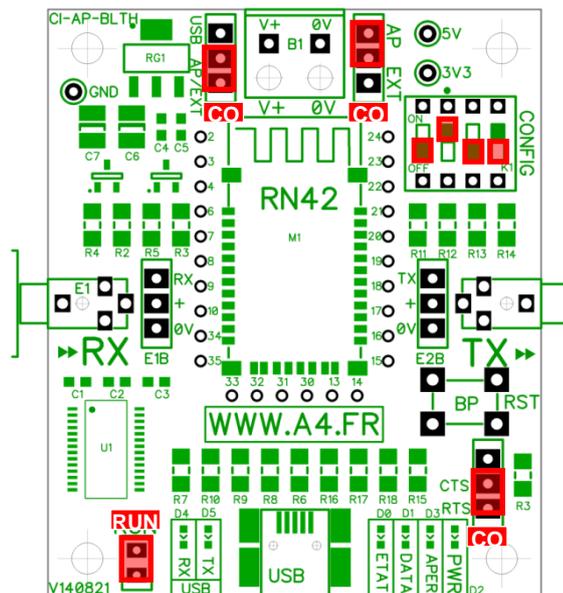
En mode avancé, il peut être configuré au travers d'une liaison par connexion USB à un PC ou par l'envoi de commandes au travers de ses liaisons RX et TX.

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.

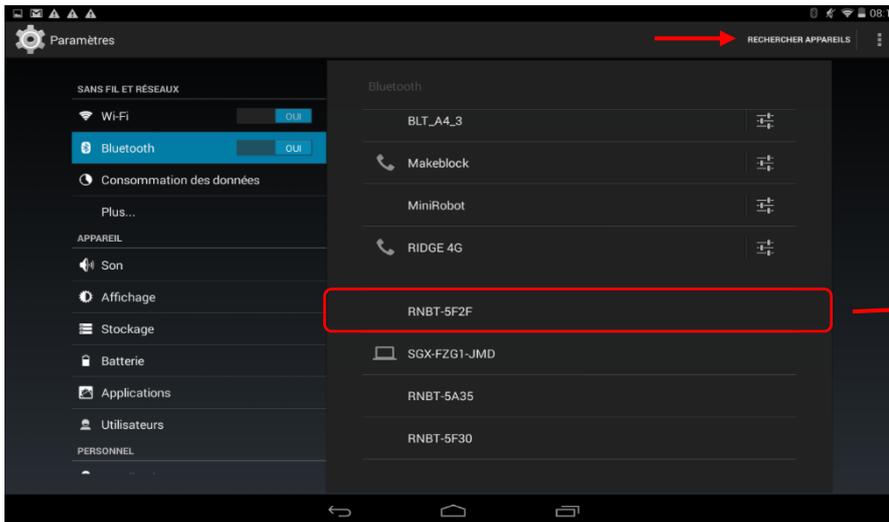
## Témoins lumineux

- 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.

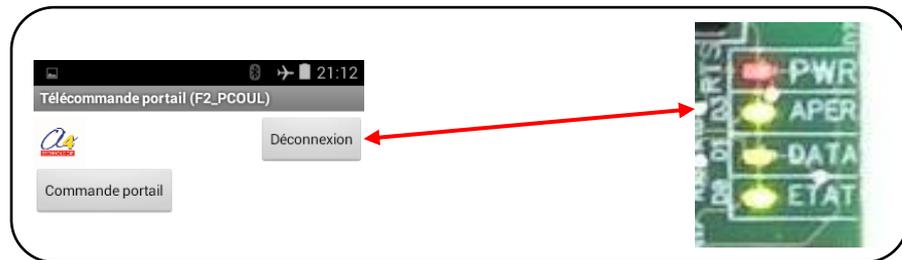
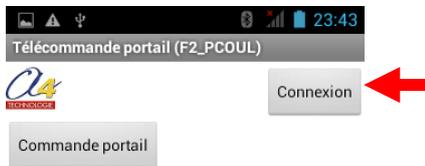


Une fois cette étape passée vous pourrez vous connecter au module à partir du programme Applinventor à chaque fois.

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.

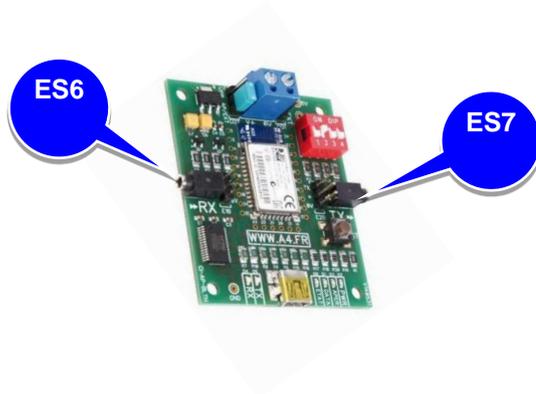
L'appui sur le bouton d'envoi de données, dans cet exemple **Commande portail**, déclenche l'allumage fugitif de ce témoin.



## 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  | (libre)   | C.5            |                    |
| 4  | (libre)   | C.4            |                    |
| 3  | Capteur de température                                    | C.3            | Capteur_temp       |
| 2  | (libre)   | C.2            |                    |
| 1  | Fin de course fenêtre dépliée                             | C.1            | FDC_Deplie         |
| 0  | Fin de course fenêtre repliée                             | C.0            | FDC_Replie         |
| EA | MODULES CAPTEURS POUR ENTRÉES ANALOGIQUES                 |                |                    |
| 3  | (libre)   | A.3            |                    |
| 2  | (libre)   | A.2            |                    |
| 1  | Capteur d'humidité  | A.1            | Capteur_humidite   |
| 0  | Sonde hygrométrique                                       | A.0            | Sonde_hygro*       |
| SN | MODULES ACTIONNEURS SORTIES NUMÉRIQUES                    |                |                    |
| 7  | Sortie reliée à la broche 2 du moteur                     | B.7            | Moteur_2           |
| 6  | Sortie reliée à la broche 1 du Moteur                     | B.6            | Moteur_1           |
| 5  | Ventilateur   | B.5            | Ventilateur        |
| 4  | Pompe à eau   | B.4            | Pompe              |
| 3  | Brumisateur   | B.3            | Brumisateur*       |
| 2  | Plateau chauffant   | B.2            | Plateau_chauffant* |
| 1  | Afficheur OLED  | B.1            | Afficheur_OLED*    |
| 0  | (libre)   | B.0            |                    |

### Câblage du module bluetooth (K-AP-MBLTH)



# Exercice niveau 3 - B.1 : Monter/Descendre le toit avec application Bluetooth

**Objectif :** contrôler les moteurs via Bluetooth à partir d'une application pour Smartphone

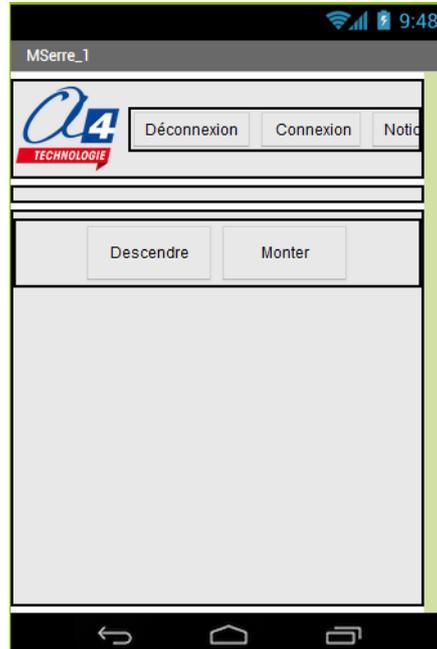
**Notion abordée :** réception de données Bluetooth envoyées par un Smartphone.

**Application Android :** Serre\_1.apk

**Fichier App Inventor :** Serre\_1.aia

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

quand Monter .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
      appeler sous-fonction Descendre
    si consigne = 2
    faire
      appeler sous-fonction Monter

sous-fonction Monter
  si entrée FDC_Deplie est désactivée
  faire
    répéter
      sortie Moteur_1 activée
    jusqu'à entrée FDC_Deplie est activée
    sortie Moteur_1 désactivée

sous-fonction Descendre
  si entrée FDC_Replie est désactivée
  faire
    répéter
      sortie Moteur_2 activée
    jusqu'à entrée FDC_Replie est activée
    sortie Moteur_2 désactivée
```

Fichier Blockly : MS\_N3\_B1.xml

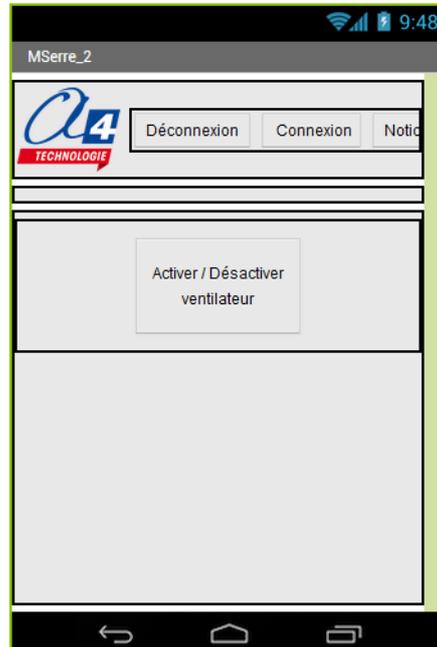
# Exercice niveau 3 - B.2 : Activer/Désactiver une sortie avec une application Bluetooth

**Objectif** : contrôler le ventilateur grâce à une application pour Smartphone

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

**Application Android** : Serre\_2.apk

**Fichier App Inventor** : Serre\_2.aia



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

**Correction :**

Blocs

A screenshot of a Blockly code editor showing a sequence of blocks. It starts with a 'début' block, followed by an 'hersetup B9600\_8' block with an 'Inverser la polarité' checkbox. A 'répéter indéfiniment' loop contains a 'faire' block with a 'debug' block, an 'hserin consigne' block, and a 'si consigne = 1' block. Inside the 'si' block, there is a 'faire' block with two conditional actions: 'si entrée Ventilateur est activée' leading to 'sortie Ventilateur désactivée', and 'sinon' leading to 'sortie Ventilateur activée'.

Fichier Blockly : MS\_N3\_B2.xml

# Exercice niveau 3 - B.3 Envoi de données sur smartphone

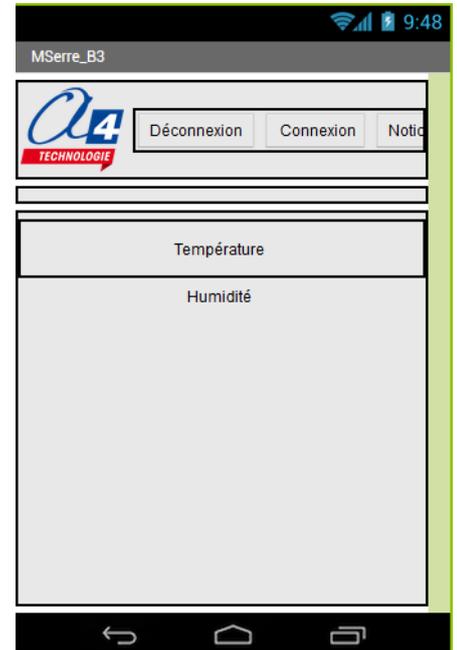
**Objectif :** Afficher sur le smartphone les valeurs de température et d'humidité dans la serre

**Notion abordée :** réception de données Bluetooth sur smartphone

**Application Android :** Serre\_3.apk

**Fichier App Inventor :** Serre\_3.aia

```
quand Horloge1 Chronomètre
faire
  si Bluetooth Est connecté
  alors
    si Bluetooth Octets disponibles pour le réception > 0
    alors
      mettre Temp Texte à joint "Température:"
      appeler Bluetooth RecevoirOctetSignéNuméro1
      joint "°C"
      mettre Humidité Texte à joint "Humidité:"
      appeler Bluetooth RecevoirOctetSignéNuméro1
      joint "%"
```



**Correction :**

Blocs

```
début
  hsersetup B9600_8
  Inverser la polarité
  répéter indéfiniment
  faire
    debug
    appeler sous-fonction Temperature
    appeler sous-fonction humidite
    hserout Temp
    hserout Hum

sous-fonction Temperature
  lire température en Capteur_temp et stocker dans Temp

sous-fonction humidite
  lire valeur analogique en Capteur_humidite et stocker dans Hum
  fixer Hum à Hum * 25
  fixer Hum à Hum / 64
```

Fichier Blockly : MS\_N3\_B3.xml

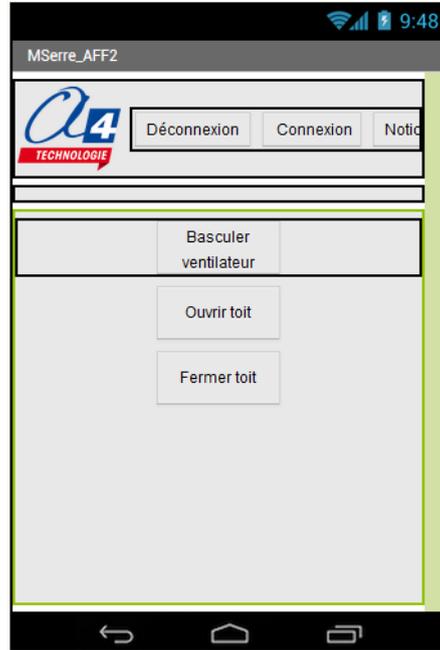
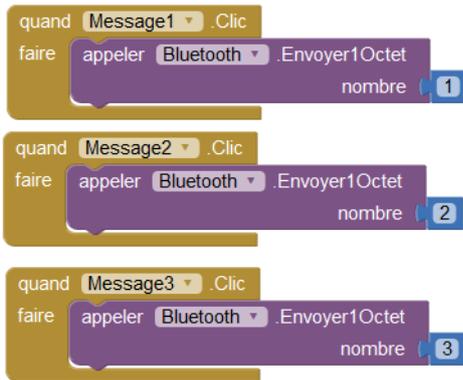
# Exercice niveau 3 – AFF : Envoyer un message à l'afficheur OLED via Bluetooth (Option afficheur)

**Objectif :** Reprendre les exercices Niveau 3 B.1 et B.2, afficher sur l'afficheur OLED si le ventilateur est activé ou non et si le toit est ouvert. Afficher également les valeurs des capteurs sur les lignes 3 et 4  
**Rappel :** La ligne 3 s'appelle avec 0x94 et la ligne 4 avec 0xD4.

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

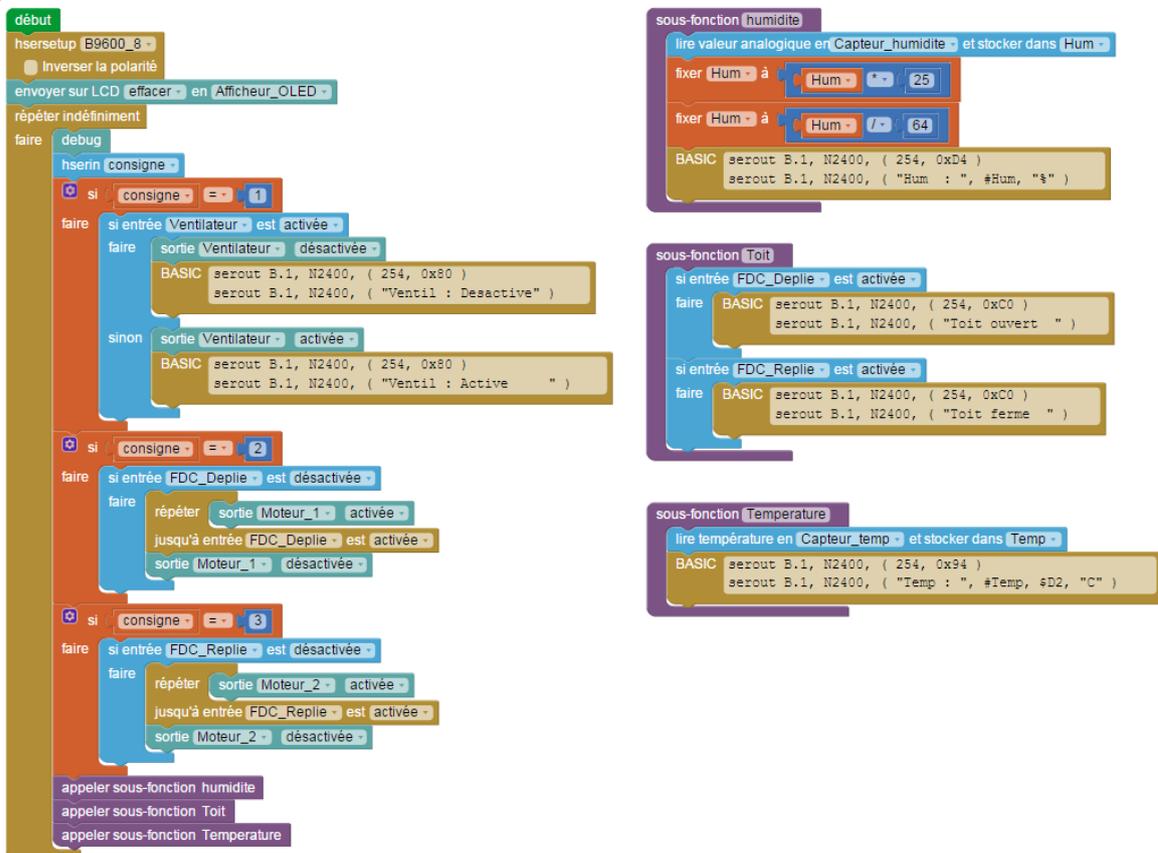
**Application Android :** Serre\_AFF.apk

**Fichier App Inventor :** Serre\_AFF.aia



**Correction :**

Blocs



Fichier Blockly : MS\_N3\_AFF.xml

## Exercice niveau 3 – C.3 : Capteur de courant et variable

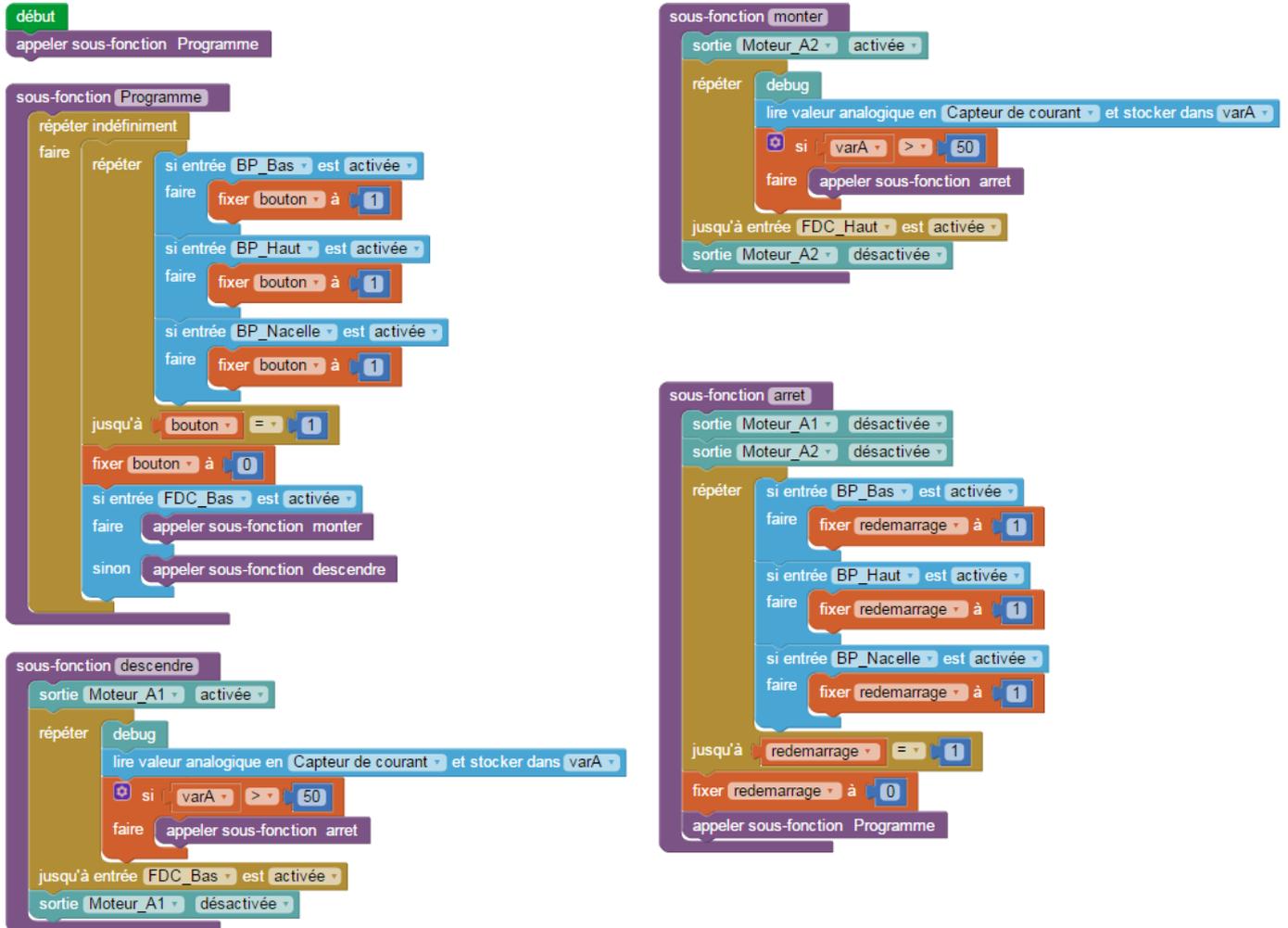
**Objectif :** Reprendre l'exercice MS\_N2\_A3.xml

Rajouter la sécurité du capteur de courant et rajouter le BP de la Nacelle pour activer le programme (utiliser des conditions Si).

**Notion abordée :** lire et interpréter une donnée d'une entrée analogique.

**Correction :**

### Blocs



Fichier Blockly : MS\_N3\_C3.xml





CONCEPTEUR ET FABRICANT DE MATÉRIELS PÉDAGOGIQUES