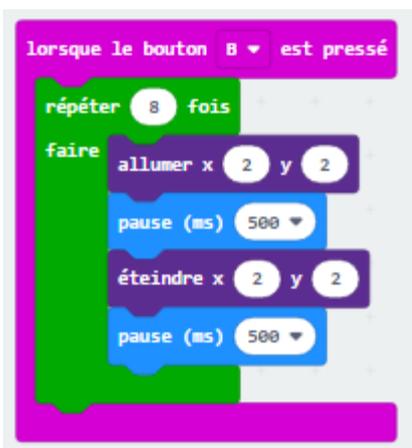
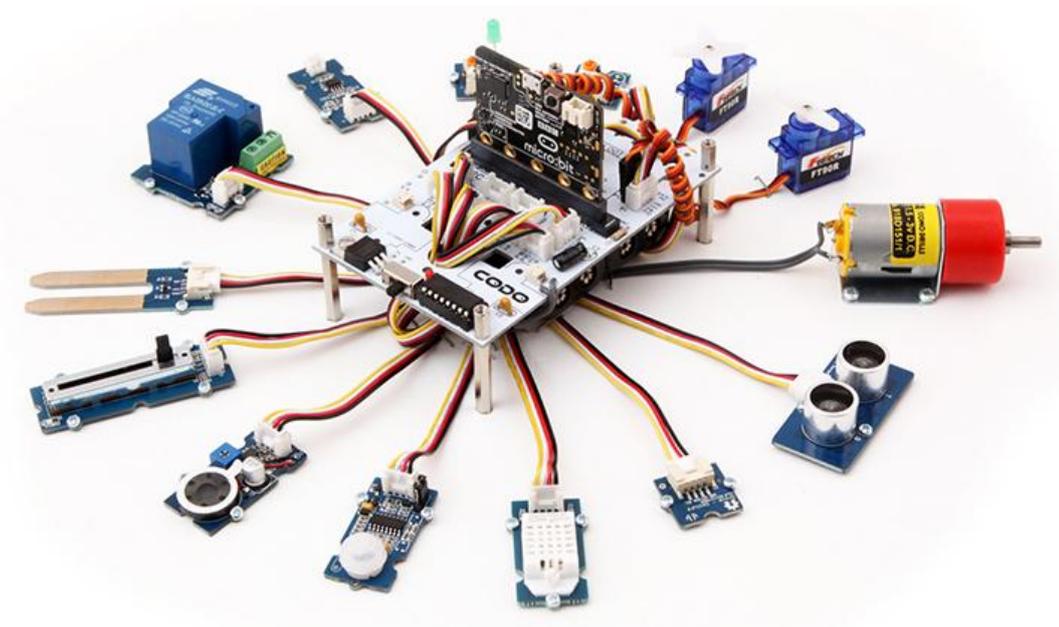


# Activités de programmation avec MakeCode et Mu Python Editor

## CODO

### Grove micro:bit project board



```

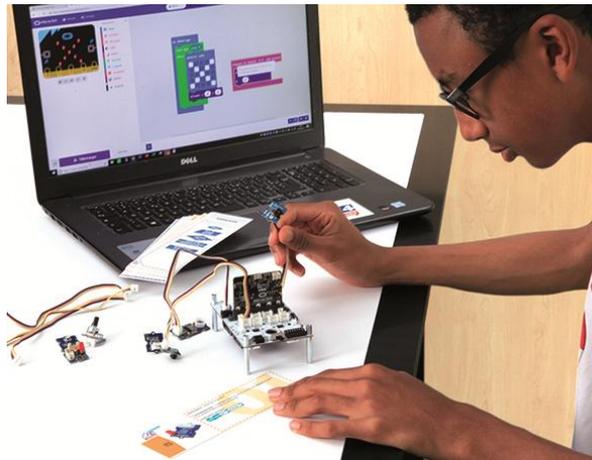
1 from microbit import *
2
3 k = 0
4
5 led = Image("00000:"
6             "00900:"
7             "00000:"
8             "00000:"
9             "00000:")
10
11 while True:
12     if button_b.is_pressed():
13         for k in range(0, 8, 1):
14             display.show(led)
15             sleep(500)
16             display.clear()
17             sleep(500)

```

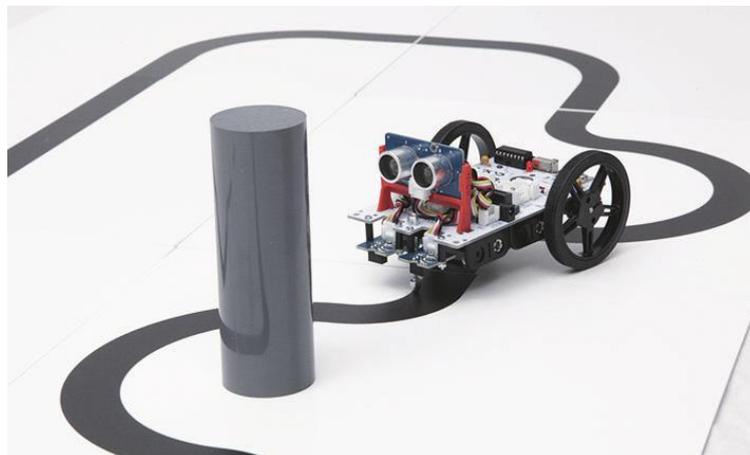


## Activités avec la carte CODO

Programmation en blocs et en Python avec les modules Grove et les cartes ExoProg



Défis robotiques avec le robot CODOK : suivi de ligne, détection d'obstacles, ...



Impression 3 D et découpe laser  
(réalisation de pare-chocs, porte-stylo, roulette avant pivotante, châssis robotique...)

# Ressources disponibles autour de la carte CODO

Nous vous proposons un ensemble de **ressources téléchargeables gratuitement** sur [www.a4.fr](http://www.a4.fr) en tapant CODO dans le moteur de recherche.

## Activités / Programmation

Dossier avec activités progressives

Fichiers de correction (programmes pour MakeCode, Mu et AppInventor).

## Carte CODO et options

- Fichier STL pour le pare-chocs en 3D, le porte-stylo et les carters infrarouge et la roulette avant
- Fichier pour la découpe laser de la plateforme robotique

## Robot CODOK

Notice de montage

## Environnements de programmation :

MakeCode

Mu (Python Editor)

App Inventor

**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 de A4 Technologie.
- **SA** : La diffusion des documents modifiés ou adaptés doit se faire sous le même régime.

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



Formation offerte en visio interactive sur internet  
Planning des sessions et inscriptions gratuites  
sur [www.a4.fr/formations](http://www.a4.fr/formations)

# SOMMAIRE

<b>Introduction.....</b>	<b>2</b>
<b>Présentation.....</b>	<b>3</b>
La carte micro:bit.....	3
La carte CODO.....	4
Les environnements de programmation.....	5
Les cartes d'activités ExoProg.....	9
<b>Programmer avec la carte micro:bit.....</b>	<b>10</b>
Matrice LED 1.....	11
Boutons A et B.....	12
Matrice LED 2.....	13
Capteur de température.....	14
Capteur de lumière.....	15
Accéléromètre.....	16
Boussole.....	18
Communication radio.....	20
Communication Bluetooth.....	22
<b>Programmation avec des modules Grove.....</b>	<b>24</b>
LED.....	26
Bouton-poussoir.....	28
Afficheur 4 digits.....	30
Télémètre à ultrasons.....	31
Speaker.....	32
Potentiomètre.....	33
Capteur de mouvement.....	34
Néopixel.....	36
Néopixel (suite).....	38
<b>Programmer avec les options CODO.....</b>	<b>40</b>
Motorisation.....	40
Porte-stylo.....	42
Pare-chocs.....	44
Détection de ligne.....	46
Joystick.....	48
<b>Programmation avancée.....</b>	<b>50</b>
Matrice LED.....	51
Minuteur.....	54
Graphiques.....	56
Matrice LED1 en Python standard.....	57
Boutons A et B en Python standard.....	58
<b>Montage des options et modules sur la CODO.....</b>	<b>60</b>
Motorisation.....	60
Plateforme robotique.....	62
Option porte-stylo.....	64
Autres modules Grove pour aller plus loin.....	65

# Introduction

---

La carte CODO a été réalisée pour connecter simultanément différents modules Grove à la carte micro:bit.

Elle peut accueillir différentes options telles que des moteurs pour fixer des roues et être transformée en robot CODO. Une fois cette option installée, il est également possible d'ajouter l'option porte-stylo et la plateforme robotique avec pare-chocs.

Enfin, pour encore plus de plaisir à programmer, munissez-vous de deux cartes micro:bit et de deux cartes CODO (une étant le robot et l'autre la carte CODO de base) et modélisez une télécommande à l'aide d'un joystick par exemple.

Tous les programmes correspondant aux activités menées autour de la carte CODO ont été réalisés sous MakeCode (blocs) et certains avec l'éditeur mu (python).

Les activités de programmation s'adressent aussi bien à des utilisateurs totalement débutants qu'à des utilisateurs avertis pour créer des scénarios de programmation élaborés allant jusqu'à l'interaction du robot avec un smartphone.

## Prérequis pour la programmation

- Câble de programmation USB
- Installer mu Python Editor
- Avoir une connexion internet pour utiliser MakeCode en ligne

### Pour l'option Bluetooth :

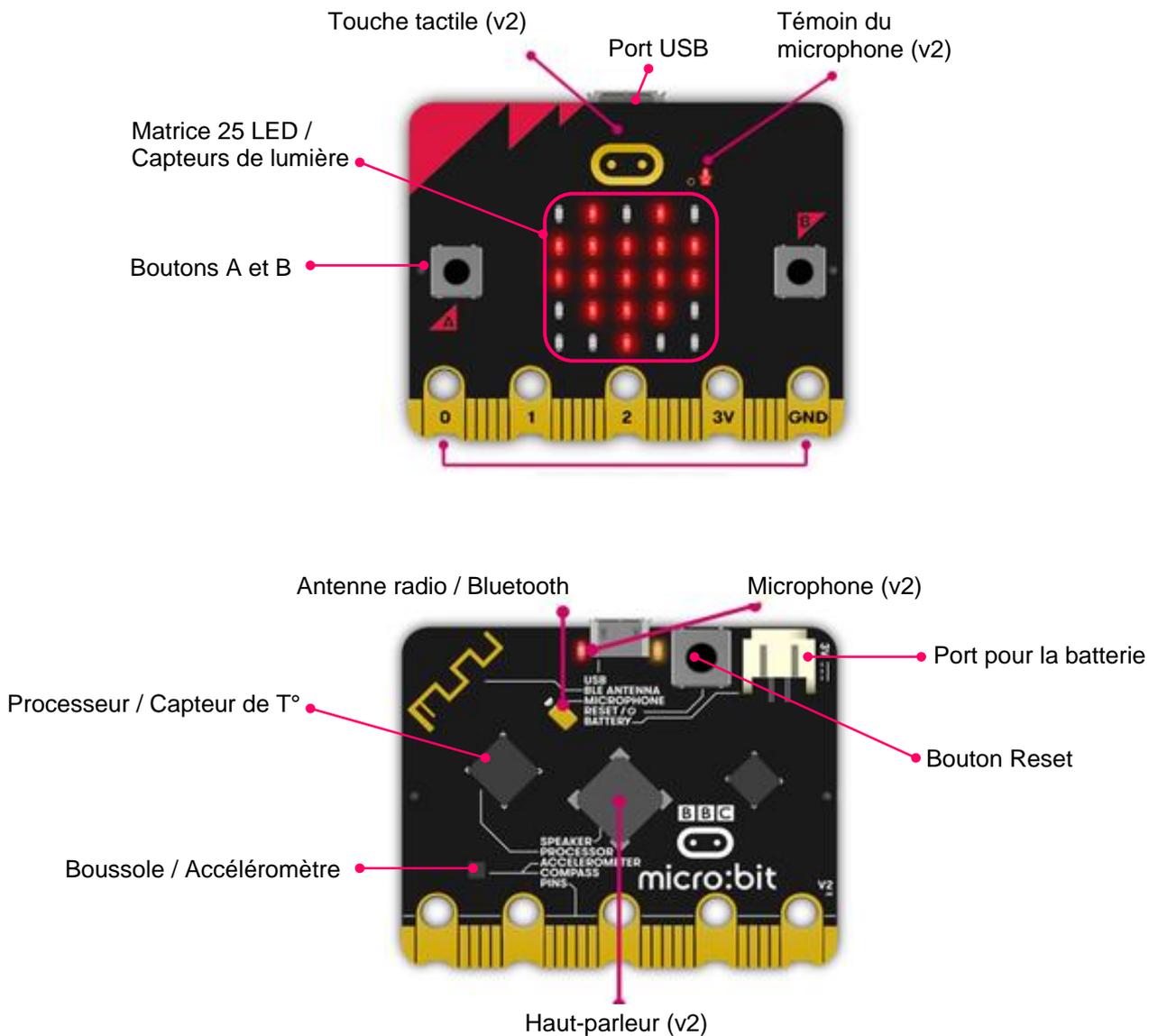
- Tablette ou smartphone Android 5 ou + équipés de Bluetooth V3.
- Connexion internet pour accéder à App Inventor : <http://ai2.appinventor.mit.edu/>
- Compte Gmail.

# Présentation

## LA CARTE MICRO:BIT

Mini carte programmable conçue pour l'apprentissage de la programmation. Il est possible de réaliser une multitude de programmes.

Les activités de programmation proposées dans le premier chapitre s'appuient sur les fonctionnalités offertes par la carte micro:bit : matrice LED, boutons-poussoirs, capteur de température, capteur de lumière, accéléromètre et boussole.



## LA CARTE CODO

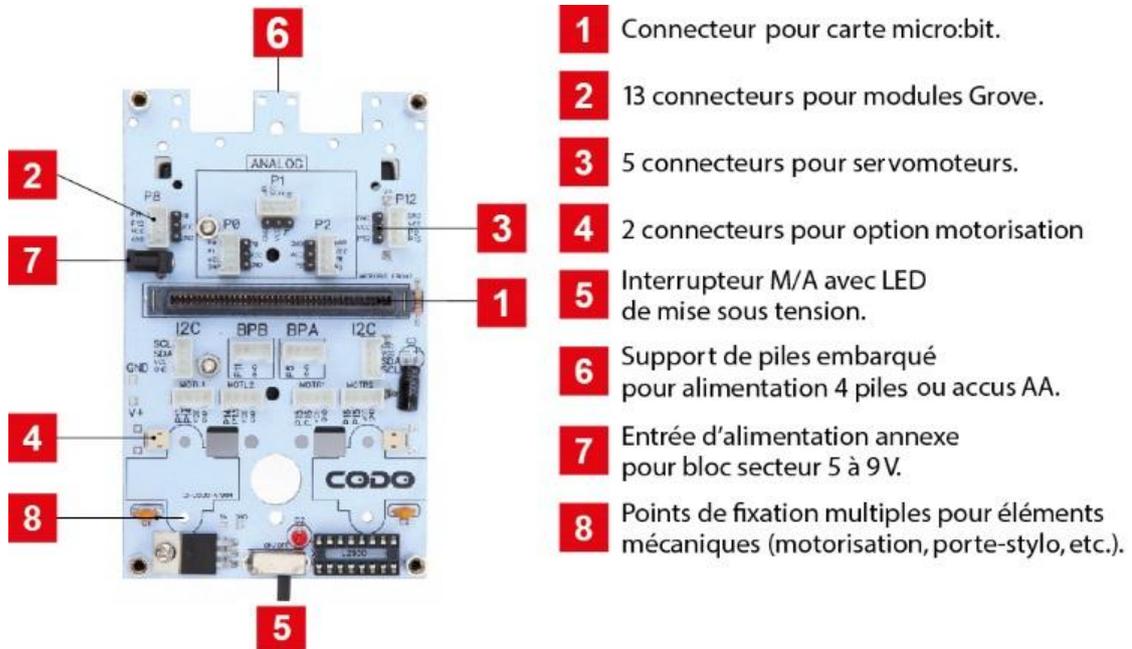
La carte CODO a été réalisée pour connecter simultanément différents modules Grove à la carte micro:bit. Elle est équipée de 13 entrées/ sorties pour modules Grove.

Elle est conçue pour être alimentée en 6V grâce à 4 emplacements pour piles 1,5V.

La carte micro:bit peut être fixée sur la carte CODO via un connecteur prévu à cet effet.

Cet assemblage est idéal pour être codé en blocs et en python.

Attention toutefois, ce dossier n'explique pas l'utilisation de la carte micro:bit ni celle des modules Grove. Il est donc recommandé d'avoir des notions de bases sur ces éléments avant d'utiliser la carte CODO.



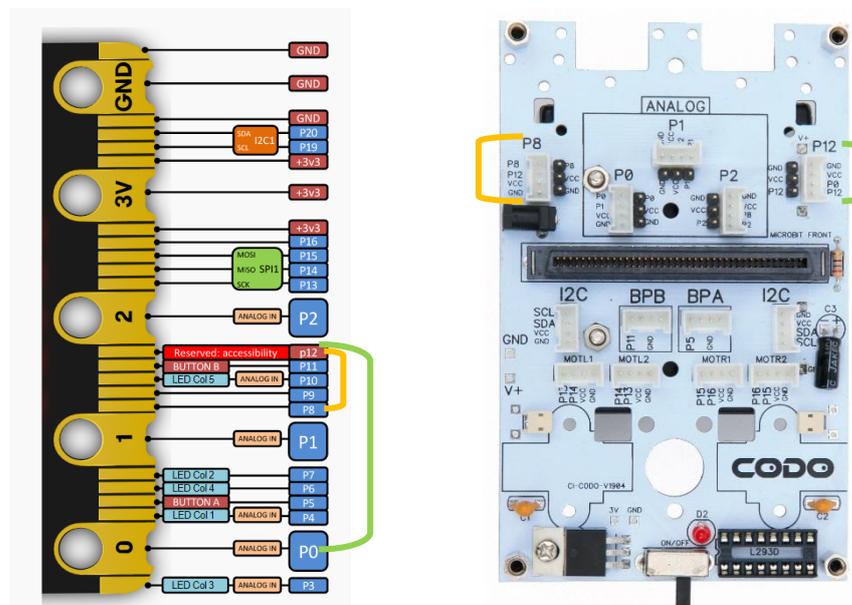
Dimensions : 72 x 113 mm.

Sur les 13 connecteurs Grove, deux correspondent à une connectique I2C, deux sont reliés aux boutons A et B de la carte micro:bit et trois sont analogiques.

La carte contient 22 perçages pour fixer divers éléments mécaniques (pièces réalisées avec imprimante 3D, découpe laser, etc.).

Embase Ø 1,3 mm pour bloc d'alimentation externe (24 V max).

Livrée avec 4 entretoises FF M3 hauteur 30 mm et 4 entretoises MF M3 hauteur 10 mm.



## LES ENVIRONNEMENTS DE PROGRAMMATION

La carte CODO est adaptée pour programmer sur le logiciel MakeCode pour coder en blocs ainsi que sur l'éditeur mu pour coder en python. Le site ApplInventor2 peut être utilisé pour la communication Bluetooth.



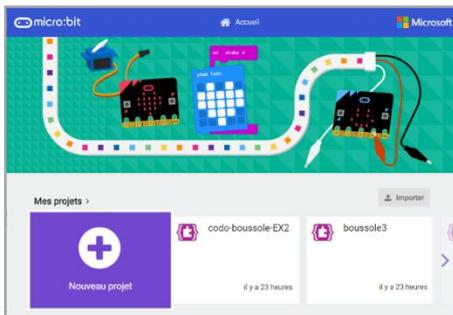
### MakeCode

Environnement de programmation en ligne.  
Programmation séquentielle ou événementielle en blocs.  
Traduction automatique en JavaScript.

Jeu d'instructions de base en français.  
Possibilité d'enrichir le jeu d'instructions.  
Espace de simulation.

Pour accéder à ce logiciel disponible gratuitement sur internet, copier le lien suivant :

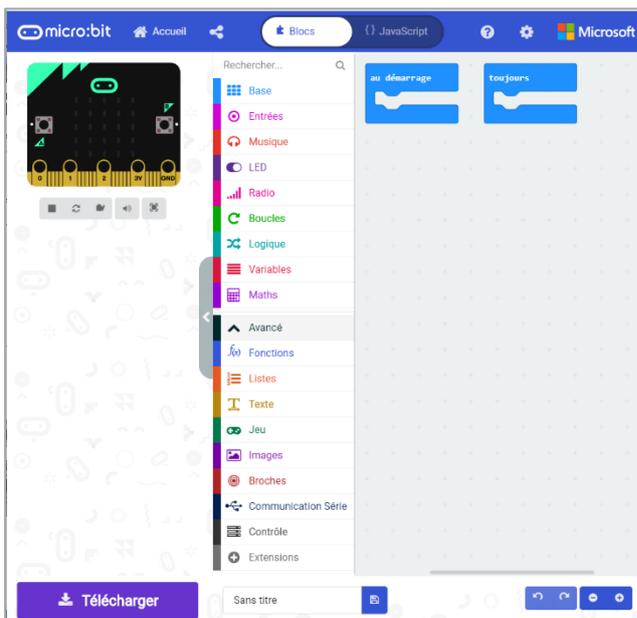
<https://makecode.microbit.org>



Cliquez sur **Nouveau projet** pour commencer à coder.

Vous pouvez importer un fichier depuis votre ordinateur avec **Importer**.

**++** Vous pouvez également faire glisser un programme directement dans la fenêtre MakeCode.



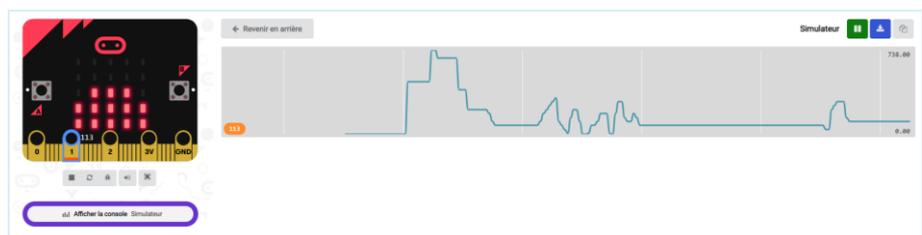
Vous pouvez coder en blocs et en JavaScript. Ici, nous nous concentrons sur le mode blocs. Sur la gauche de l'écran, vous trouvez les différents blocs disponibles. En haut, les blocs de bases sont affichés.

La carte micro:bit en haut à gauche de l'écran permet d'avoir une simulation du programme en cours.

**++** Lorsque vous cliquez droit sur un bloc et aide, vous obtiendrez une explication sur le bloc correspondant en anglais.

**++** Vous pouvez traduire l'aide en français avec un nouveau clic droit.

### Mode simulation



Relier ensuite la carte micro:bit à l'ordinateur avec le câble de programmation USB. Dans le logiciel MakeCode, créer votre programme et le télécharger. Le copier ensuite sur la carte micro:bit. Votre programme est lancé.

## Extension CODO

Vous pouvez ajouter des blocs supplémentaires.  
A partir de la rubrique **Avancé**, cliquez sur **Extensions**.



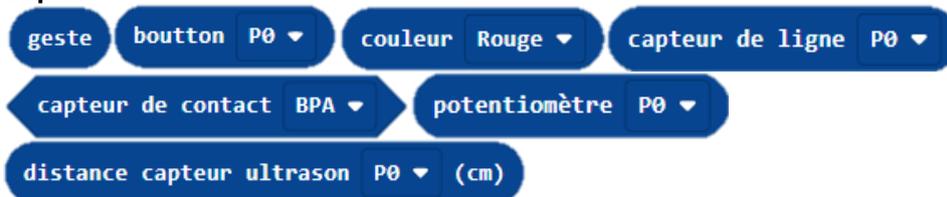
Tapez « CODO » dans le moteur de recherche.  
Pour obtenir les blocs correspondant aux moteurs du robot,  
aux modules Grove abordés dans les cartes ExoProg.



### Moteurs



### Capteurs



### Actionneurs



### Afficheurs



### Plus



### Autres extensions : Néopixel, Bluetooth, Grove ...

Entrez les liens suivants :

<https://github.com/Microsoft/pxt-neopixel.git> pour obtenir les blocs correspondant au Neopixel.

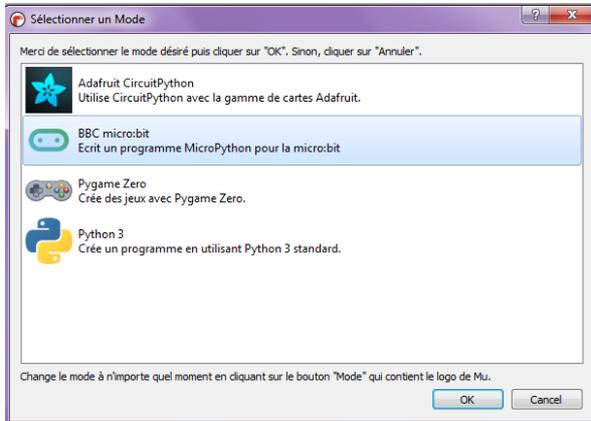
<https://github.com/Seeed-Studio/pxt-grove.git> pour avoir des blocs correspondant aux autres modules Grove.

## Mu Python Editor

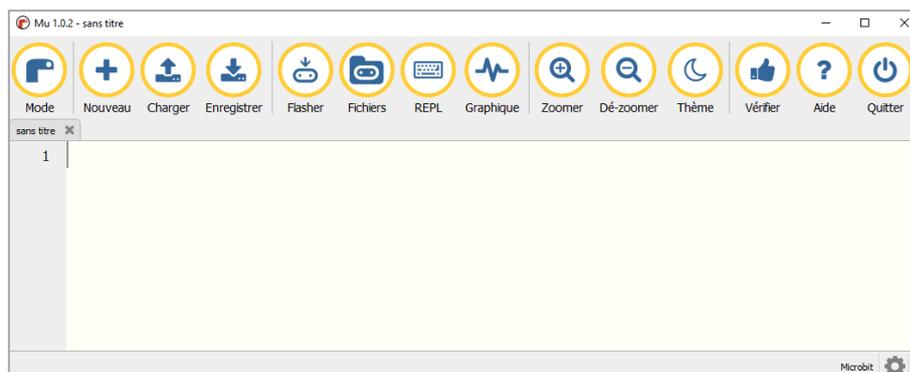


Environnement (gratuit) de programmation en Python.  
Offre des fonctionnalités plus avancées que l'environnement de base MicroPython (flashage direct de la carte micro:bit, fonction d'auto-complétion du code).  
Sur PC Windows ou Mac OSX. A télécharger gratuitement.

Le logiciel mu permet de coder en python standard et en micro python pour la carte micro:bit. Le logiciel est à télécharger avec le lien suivant par exemple : <https://codewith.mu/>



Pour lancer un programme, écrivez-le et cliquez sur « flasher » en haut de l'écran. Vous pouvez également importer un fichier existant sur votre ordinateur en cliquant sur « charger ».



Guide les librairies :

Vous trouverez de nombreuses librairies sur le site suivant :

<https://microbit-micropython.readthedocs.io/fr/latest/tutorials/introduction.html>

## App Inventor 2

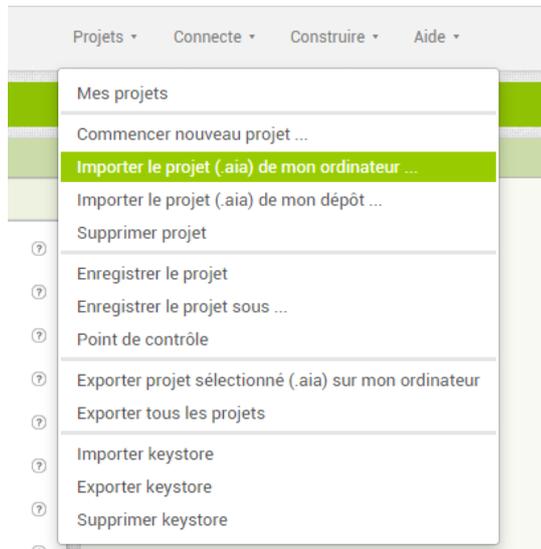


Environnement de développement pour concevoir des applications pour smartphone ou tablette Android, développé par le MIT pour l'éducation.

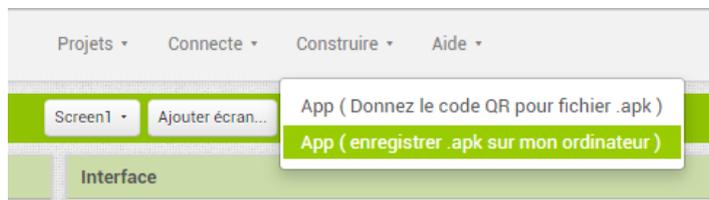
Pour les activités menées avec un smartphone ou une tablette, les programmes et applications ont été réalisés sous Appinventor2.

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

- 1) Activer le Bluetooth et appairer votre matériel (smartphone ou tablette) à la carte micro:bit.
- 2) Se connecter sur Appinventor2 et charger le fichier « Documentdebase.aia ».



- 3) Compléter l'application et l'enregistrer en fichier .apk



- 4) Installer l'application correspondante sur votre smartphone.
- 5) Charger le programme **codobluetoothbase.hex** sur MakeCode.
- 6) Charger l'extension <https://microbit-blockytalky.glitch.me/> sur MakeCode.
- 7) Compléter le programme suivant l'exercice demandé.
- 8) Télécharger le programme et le copier sur la carte micro bit.

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

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

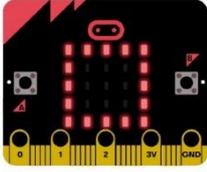
## LES CARTES D'ACTIVITES EXOPROG

Toutes les cartes sont élaborées sur le modèle suivant :

- au recto, présentation du module et des instructions utilisées.
- au verso, 3 exercices de difficulté progressive.



**MATRICE LED 1**



**Description :** les LED émettent de la lumière.

**Caractéristiques :** la carte micro:bit possède 25 LED, toutes programmables individuellement, ce qui permet d'afficher du texte, des nombres et des images.

**Instruction de base :**

montrer l'icône 

Exercices d'application au dos

EXERCICES

**NIVEAU 1**

Afficher « Hello ! » lorsque le bouton A est pressé.

python™  
codo-TEXTE-EX1.hex

**NIVEAU 2**

Activer la LED en bas à droite pendant 1 seconde lorsque le bouton A est pressé.

python™  
codo-TEXTE-EX2.hex

**NIVEAU 3**

Faire clignoter la LED centrale 8 fois lorsque le bouton B est pressé.

python™  
codo-TEXTE-EX3.hex

A4 Technologie

Pour chaque module ou option, nous vous proposons des exercices avec un fichier de correction qui propose un exemple de programme réalisé sous MakeCode en blocs. Lorsque le logo python apparaît à côté cela signifie qu'une correction en python sous mu est disponible.

Deux approches :

- Avec les corrections de programmes, les utilisateurs découvrent les principes de la programmation 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, ils peuvent ensuite le comparer au fichier de correction.

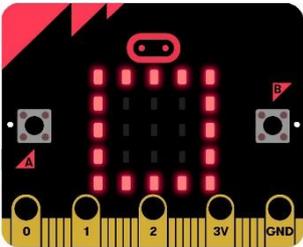
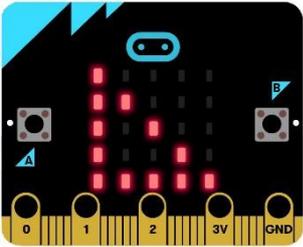
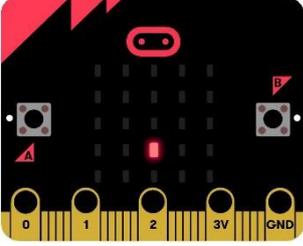
### Modules Grove traités dans les cartes d'activités ExoProg Grove /micro:bit

Module	Référence
Bouton-poussoir	S-101020003
Potentiomètre	S-101020017
Détection de ligne	S-101020009
Télémètre à ultrasons	S-101020010
LED rouge	S-104030005
Afficheur 4 digits	S-104030003
Ruban de LED	S-104020108
Joystick	S-101020028
Haut-parleur miniature	S-107020001
Capteur de mouvement	S-101020083

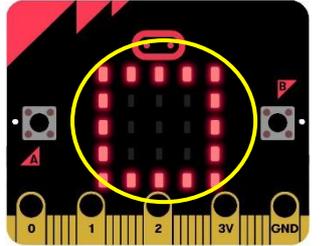


Les exercices proposés sont transposables sur d'autres modules de la gamme Grove.

# Programmer avec la carte micro:bit

Modules	Enoncés	Corrections
<b>Matrice LED 1</b> 	Niv1 : afficher un carré.	codo-MOTIFS-EX1
	Niv2 : afficher un carré et une seconde après un triangle.	codo-MOTIFS-EX2
	Niv3 : répéter l'affichage d'un carré, d'un triangle, puis d'une croix avec un intervalle de 1 seconde.	codo-MOTIFS-EX3
<b>Bouton A et B</b> 	Niv1 : afficher un triangle quand le bouton A est pressé.	codo-BOUTONS-EX1
	Niv2 : afficher un triangle quand le bouton A est pressé et effacer l'écran après 3 secondes.	codo-BOUTONS-EX2
	Niv3 : afficher une croix lorsque vous appuyez sur A, le nombre 15 lorsque vous appuyez sur B et la première lettre de votre prénom lorsque vous appuyez sur A et B.	codo-BOUTONS-EX3
<b>Matrice LED 2</b> 	Niv1 : afficher « Hello ! » lorsque vous appuyez sur le bouton A.	codo-TEXTE-EX1
	Niv2 : activer la LED en bas à droite pendant 1 seconde lorsque vous appuyez sur le bouton A.	codo-MATRICE-EX1
	Niv3 : faire clignoter la LED centrale 8 fois lorsque le bouton B est pressé.	codo-MATRICE-EX2
<b>Capteur de température</b> 	Niv1 : afficher la valeur de la température.	codo-TEMP-EX1
	Niv2 : afficher la valeur de la température et allumer la LED en haut à gauche si la température dépasse 28°C.	codo-TEMP-EX2
	Niv3 : faire clignoter la LED en haut à gauche de plus en plus rapidement en fonction de la valeur de la température.	codo-TEMP-EX3

# MATRICE LED 1

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li>Base</li> </ul>	

## Niv1 : Afficher un carré

Avec MakeCode : codo-MOTIFS-EX1.hex	Avec mu Python Editor : codo-MOTIFS-EX1.py
	<pre data-bbox="845 660 1300 772"> 1 from microbit import * 2 3 display.show(Image.SQUARE) </pre>

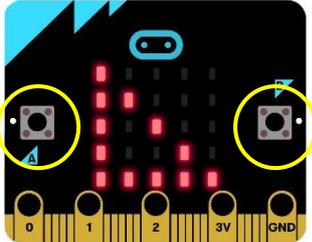
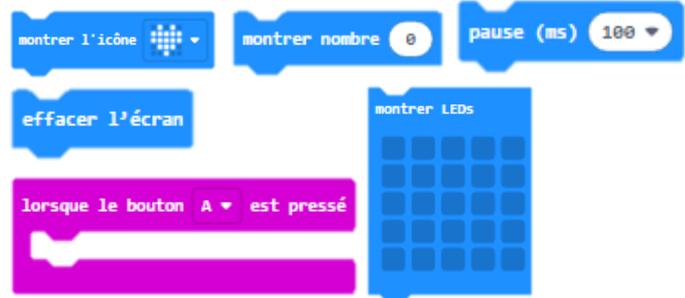
## Niv2 : afficher un carré et une seconde après un triangle

Avec MakeCode : codo-MOTIFS-EX2.hex	Avec mu Python Editor : codo-MOTIFS-EX2.py
	<pre data-bbox="845 952 1300 1108"> 1 from microbit import * 2 3 display.show(Image.SQUARE) 4 sleep(1000) 5 display.show(Image.TRIANGLE) </pre>

## Niv3 : répéter l'affichage d'un carré, d'un triangle, puis d'une croix avec un intervalle de 1 sec.

Avec MakeCode : codo-MOTIFS-EX3.hex	Avec mu Python Editor : codo-MOTIFS-EX3.py
	<pre data-bbox="845 1422 1372 1758"> 1 from microbit import * 2 3 k = 0 4 5 for k in range(0, 10000000000, 1): 6     display.show(Image.SQUARE) 7     sleep(1000) 8     display.show(Image.TRIANGLE) 9     sleep(1000) 10    display.show(Image.NO) 11    sleep(1000) </pre>

# BOUTONS A ET B

<b>Module</b> 	<b>Instructions / Fonctions utilisées</b> <ul style="list-style-type: none"> <li>Base</li> <li>Entrées</li> </ul>	
--	--	--

## Niv1 : afficher un triangle quand le bouton A est pressé

Avec MakeCode : codo-BOUTONS-EX1.hex 	Avec mu Python Editor : codo-BOUTONS-EX1.py <pre data-bbox="837 622 1372 788"> 1 from microbit import * 2 3 while True: 4     if button_a.is_pressed(): 5         display.show(Image.TRIANGLE_LEFT) 6 7                     </pre>
---	---

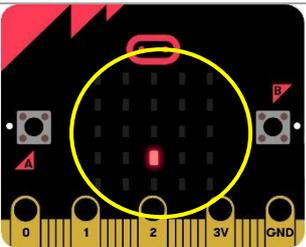
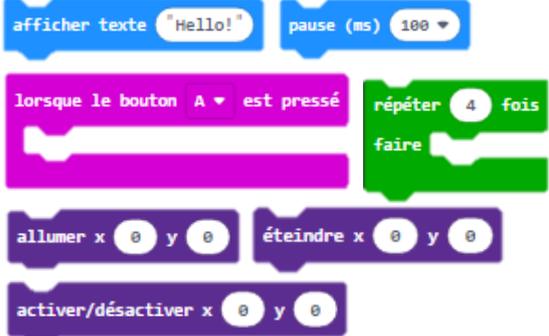
## Niv2 : afficher un triangle quand le bouton A est pressé et effacer l'écran après 3 sec.

Avec MakeCode : codo-BOUTONS-EX2.hex 	Avec mu Python Editor : codo-BOUTONS-EX2.py <pre data-bbox="837 934 1362 1131"> 1 from microbit import * 2 3 while True: 4     if button_a.is_pressed(): 5         display.show(Image.TRIANGLE_LEFT) 6         sleep(3000) 7         display.clear() 8                     </pre>
--	--

## Niv3 : afficher une croix lorsque le bouton A est pressé, le nombre 15 lorsque le bouton B est pressé et la première lettre de votre prénom lorsque les boutons A et B sont pressés.

Avec MakeCode : codo-BOUTONS-EX3.hex 	Avec mu Python Editor : codo-BOUTONS-EX3.py <pre data-bbox="837 1382 1350 1859"> 1 from microbit import * 2 3 lettre = Image("00000:" 4                 "00900:" 5                 "00900:" 6                 "00990:" 7                 "00000:") 8 9 while True: 10     if button_a.is_pressed(): 11         display.show(Image.NO) 12         if button_b.is_pressed(): 13             display.show(lettre) 14     elif button_b.is_pressed(): 15         display.scroll(15) 16     display.clear()                     </pre>
---	--

## MATRICE LED 2

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li><span style="color: blue;">■</span> Base</li> <li><span style="color: magenta;">■</span> Entrées</li> <li><span style="color: purple;">■</span> LED</li> <li><span style="color: green;">■</span> Boucles</li> </ul>	

### Niv1 : afficher « Hello ! » lorsque le bouton A est pressé

Avec MakeCode : codo-TEXTE-EX1.hex	Avec mu Python Editor : codo-TEXTE-EX1.py
	<pre data-bbox="847 660 1482 806"> 1 from microbit import * 2 3 while True: 4     if button_a.is_pressed(): 5         display.scroll("HELLO!")                     </pre>

### Niv2 : activer la LED en bas à droite pendant 1 seconde lorsque le bouton A est pressé.

Avec MakeCode : codo-MATRICE-EX1.hex	Avec mu Python Editor : codo-MATRICE-EX1.py
	<pre data-bbox="847 949 1482 1330"> 1 from microbit import * 2 3 led = Image("00000:" 4               "00000:" 5               "00000:" 6               "00000:" 7               "00009:") 8 9 while True: 10     if button_a.is_pressed(): 11         display.show(led) 12         sleep(1000) 13         display.clear()                     </pre>

### Niv3 : faire clignoter la LED centrale 8 fois lorsque le bouton B est pressé.

Avec MakeCode : codo-MATRICE-EX2.hex	Avec mu Python Editor : codo-MATRICE-EX2.py
	<pre data-bbox="847 1473 1482 1939"> 1 from microbit import * 2 3 k = 0 4 5 led = Image("00000:" 6               "00900:" 7               "00000:" 8               "00000:" 9               "00000:") 10 11 while True: 12     if button_b.is_pressed(): 13         for k in range(0, 8, 1): 14             display.show(led) 15             sleep(500) 16             display.clear() 17             sleep(500)                     </pre>

# CAPTEUR DE TEMPERATURE

<p><b>Module</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li> Base</li> <li> Entrées</li> <li> Logique</li> <li> LED</li> </ul>	
--	--	--

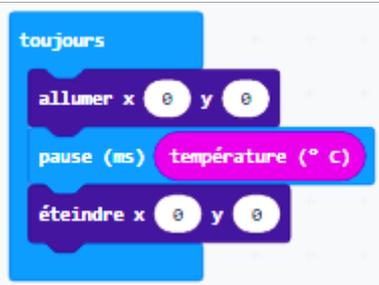
## Niv1 : afficher la valeur de la température

<p>Avec MakeCode : codo-TEMP-EX1.hex</p> 	<p>Avec mu Python Editor : codo-TEMP-EX1.py</p> <pre data-bbox="847 611 1452 824"> 1 from microbit import * 2 3 t = temperature() 4 5 while True: 6     display.scroll(t) 7     sleep(1000)                 </pre>
--	--

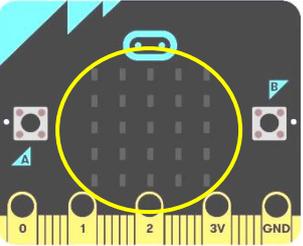
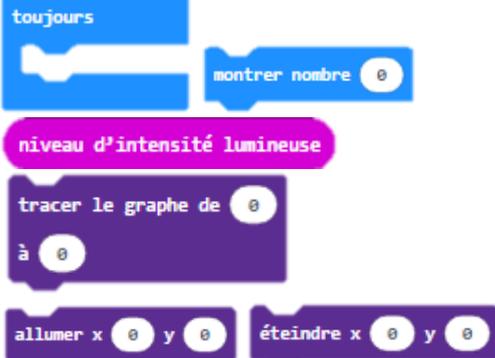
## Niv 2 : afficher la valeur de la T° et allumer la LED en haut à gauche si la T° dépasse 28°C

<p>Avec MakeCode : codo-TEMP-EX2.hex</p> 	<p>Avec mu Python Editor : codo-TEMP-EX2.py</p> <pre data-bbox="847 963 1452 1451"> 1 from microbit import * 2 3 t = temperature() 4 5 i = Image("90000:" 6           "00000:" 7           "00000:" 8           "00000:" 9           "00000:") 10 11 while True: 12     if t &gt; 28: 13         display.show(i) 14     else: 15         display.scroll(t)                 </pre>
---	---

## Niv 3 : faire clignoter la LED en haut à gauche de plus en plus rapidement en fonction de la T°

<p>Avec MakeCode : codo-TEMP-EX3.hex</p> 	<p>Avec mu Python Editor : codo-TEMP-EX3.py</p> <pre data-bbox="847 1592 1452 2016"> 1 from microbit import * 2 3 i = Image("90000:" 4           "00000:" 5           "00000:" 6           "00000:" 7           "00000:") 8 9 while True: 10     if temperature() &gt; 0: 11         display.show(i) 12         sleep(temperature()) 13         display.clear() 14         sleep(temperature())                 </pre>
--	--

# CAPTEUR DE LUMIERE

<p><b>Module</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li>Base</li> <li>Entrées</li> <li>LED</li> </ul>	
--	---	--

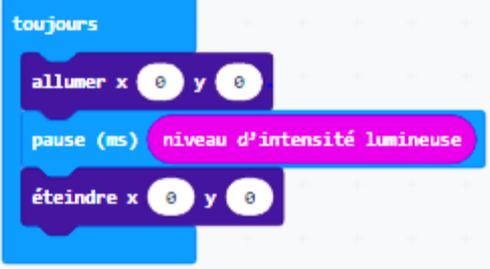
## Niv1 : tracer le graphe du niveau d'intensité lumineuse

<p>Avec MakeCode : codo-LUM-EX1.hex</p> 	<p>Avec mu Python Editor : codo-LUM-EX1.py</p> <pre> 1 from microbit import * 2 3 lumière = display.read_light_level() 4 while True: 5     display.scroll(lumière) 6     sleep(1000) </pre>
---	---

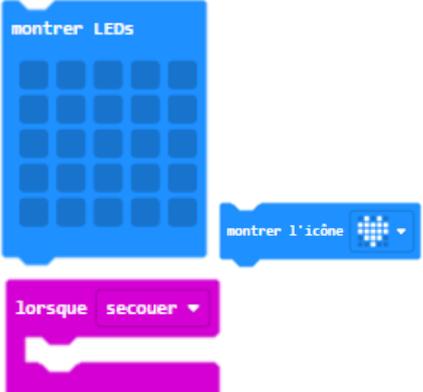
## Niv2 : afficher la valeur du niveau d'intensité lumineuse

<p>Avec MakeCode : codo-LUM-EX2.hex</p> 	<p>Avec mu Python Editor : codo-LUM-EX2.py</p> <pre> 1 from microbit import * 2 3 lumière = display.read_light_level() 4 intensité = (lumière/255)*100 5 while True: 6     display.scroll(intensité) 7     sleep(1000) </pre>
---	---

## Niv3 : faire clignoter une LED en fonction de l'intensité mesurée par le capteur de lumière

<p>Avec MakeCode : codo-LUM-EX3.hex</p> 	<p>Avec mu Python Editor : codo-LUM-EX3.py</p> <pre> 1 from microbit import * 2 3 LED = Image ("00000:" 4               "00000:" 5               "00900:" 6               "00000:" 7               "00000:") 8 9 lumière = display.read_light_level() 10 11 while True: 12     display.show(LED) 13     sleep(lumière) 14     display.clear() 15     sleep(lumière) </pre>
---	--

# ACCELEROMETRE

<p><b>Module</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li>Base</li> <li>Entrées</li> </ul>	
--	--	--

**Niv1 : lorsque l'on secoue la carte, afficher l'icône "fâché".**

<p>Avec MakeCode : codo-accel-EX1.hex</p> 	<p>Avec mu Python Editor : codo-accel-EX1.py</p> <pre> 1 from microbit import * 2 3 while True: 4     if accelerometer.was_gesture("shake"): 5         display.show(Image.ANGRY)         </pre>
---	---

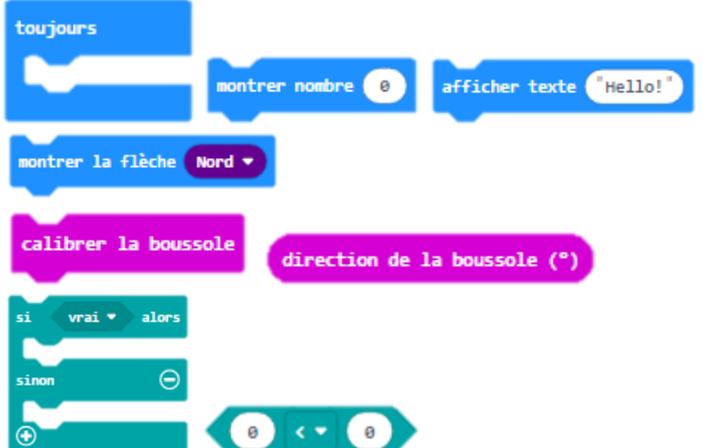
**Niv2 : lorsque l'on incline la carte à droite, afficher une flèche à droite et afficher une flèche à gauche lorsque l'on incline la carte à gauche**

<p>Avec MakeCode : codo-accel-EX2.hex</p> 	<p>Avec mu Python Editor : codo-accel-EX2.py</p> <pre> 1 from microbit import * 2 3 gauche = Image("00900:" 4               "09000:" 5               "99999:" 6               "09000:" 7               "00900:") 8 9 droite = Image("00900:" 10              "00090:" 11              "99999:" 12              "00090:" 13              "00900:") 14 15 while True: 16     if accelerometer.was_gesture("right"): 17         display.show(droite) 18     if accelerometer.was_gesture("left"): 19         display.show(gauche)         </pre>
---	---

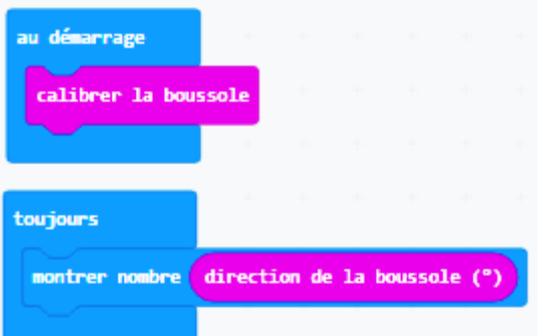
### Niv3 : lorsqu'un petit choc est détecté, allumer toutes les LED et les éteindre 1 seconde après

Avec MakeCode : codo-accel-EX3.hex	Avec mu Python Editor : codo-accel-EX3.py
 <p>The screenshot shows the MakeCode editor interface. At the top, there is a pink 'when 3g' trigger block. Below it is a blue 'show LEDs' block with a 5x5 grid of white squares. This is followed by a 'pause (ms)' block set to 1000. At the bottom, there is another blue 'show LEDs' block with a 5x5 grid of dark blue squares.</p>	<pre data-bbox="858 197 1310 517">1 from microbit import * 2 3 allumer = Image("99999:" 4                 "99999:" 5                 "99999:" 6                 "99999:" 7                 "99999:") 8 9 while True: 10     if accelerometer.was_gesture("3g"): 11         display.show(allumer) 12         sleep(1000) 13         display.clear() 14</pre>

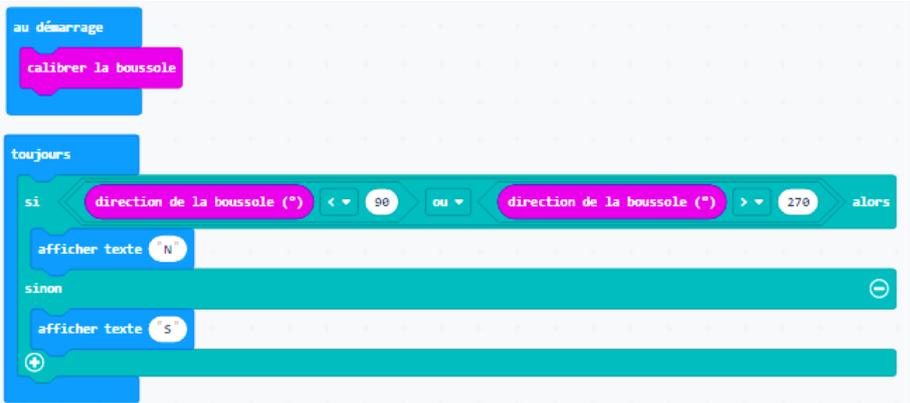
# BOUSSELE

<p><b>Module</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li>Base</li> <li>Entrées</li> <li>Logique</li> </ul>	
--	---	--

**Niv 1 : afficher la direction de la boussole, la faire pivoter et observer les variations.**

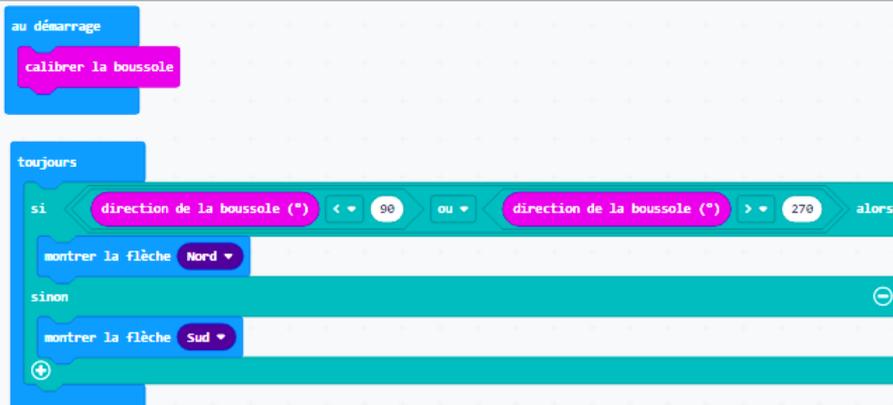
<p>Avec MakeCode : codo-boussole-EX1.hex</p> 	<p>Avec mu Python Editor : codo-boussole-EX1.py</p> <pre> 1 from microbit import * 2 3 compass.calibrate() 4 5 while True: 6     display.scroll(compass.heading()) 7     sleep(1000) 8     display.clear() </pre>
---	---

**Niv 2 : si la boussole est pointée vers le nord, afficher N ; si elle est pointée vers le sud, afficher S**

<p>Avec MakeCode : codo-boussole-EX2.hex</p> 
<p>Avec mu Python Editor : codo-boussole-EX2.py</p> <pre> 1 from microbit import * 2 3 compass.calibrate() 4 5 while True: 6     if compass.heading() &lt; 90 or compass.heading() &gt; 270: 7         display.show("N") 8         sleep(1000) 9         display.clear() 10    else: 11        display.show("S") 12        sleep(1000) 13        display.clear() </pre>

**Niv3 : si la boussole est pointée vers le nord, montrer la flèche nord et si elle est pointée vers le sud, montrer la flèche sud**

Avec MakeCode : codo-boussole-EX3.hex



The image shows a Scratch-style code editor with the following blocks:

- au démarrage** (when started):
  - calibrer la boussole (calibrate compass)
- toujours** (forever loop):
  - si** (if) block: direction de la boussole (\*) < 90 ou direction de la boussole (\*) > 270 alors (then) block:
    - montrer la flèche Nord (show arrow North)
  - sinon** (else) block:
    - montrer la flèche Sud (show arrow South)

Avec mu Python Editor : codo-boussole-EX3.py

```
1 from microbit import *
2
3 compass.calibrate()
4
5 N = Image("00000:"
6           | "00000:"
7           | "00000:"
8           | "00000:"
9           | "00000:")
10
11 S = Image("00000:"
12           | "00000:"
13           | "00000:"
14           | "00000:")
15
16
17 while True:
18     if compass.heading() < 90 or compass.heading() > 270:
19         display.show(N)
20         sleep(1000)
21         display.clear()
22     else:
23         display.show(S)
24         sleep(1000)
25         display.clear()
```

## COMMUNICATION RADIO

L'outil **Radio** sur le logiciel MakeCode permet la communication entre plusieurs cartes micro:bit. Il est nécessaire de créer deux programmes, un pour chaque carte micro:bit. Elles doivent être programmées selon le même « groupe radio » pour que les programmes fonctionnent.

<p><b>Radio</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li> Base</li> <li> Entrées</li> <li> Radio</li> <li> Variables</li> <li> Logique</li> </ul>	<pre> au démarrage   montrer nombre 0   effacer l'écran  lorsque le bouton A est pressé   radio définir groupe 1   envoyer le nombre 0 par radio  quand une donnée est reçue par radio receivedNumber   si vrai alors   sinon   </pre>
---	---	--

### Niveau 1 : Communication de la carte micro:bit 1 vers la carte micro:bit 2.

Lorsque l'on appuie sur le bouton A sur la carte 1, afficher 1 sur les deux cartes.

<p>Avec MakeCode : codo-RADIO-EX1A.hex</p> <pre> au démarrage   radio définir groupe 1  lorsque le bouton A est pressé   envoyer le nombre 1 par radio   montrer nombre 1   </pre>	<p>Avec MakeCode : codo-RADIO-EX1B.hex</p> <pre> au démarrage   radio définir groupe 1  Quand une donnée est reçue par onde radio receivedNumber   montrer nombre receivedNumber   </pre>
--	---

## Niveau 2 : Communication bidirectionnelle.

Lorsque l'on appuie sur le bouton A de la carte 1 le chiffre 3 apparait sur la carte 2, lorsque l'on appuie sur le bouton A sur la carte 2 le chiffre 3 disparaît et apparait sur la carte 1.

Avec MakeCode : codo-RADIO-EX2A.hex	Avec MakeCode : codo-RADIO-EX2B.hex
	

**Niveau 3 : Si vous appuyez sur le bouton A, envoyer 1 par radio et allumer les 30 LED du Neopixel en rouge. Si vous appuyez sur le bouton B, envoyer 2 par radio et allumer les 30 LED en vert.**

Avec MakeCode : codo-RADIO-EX3A.hex	Avec MakeCode : codo-RADIO-EX3B.hex
	

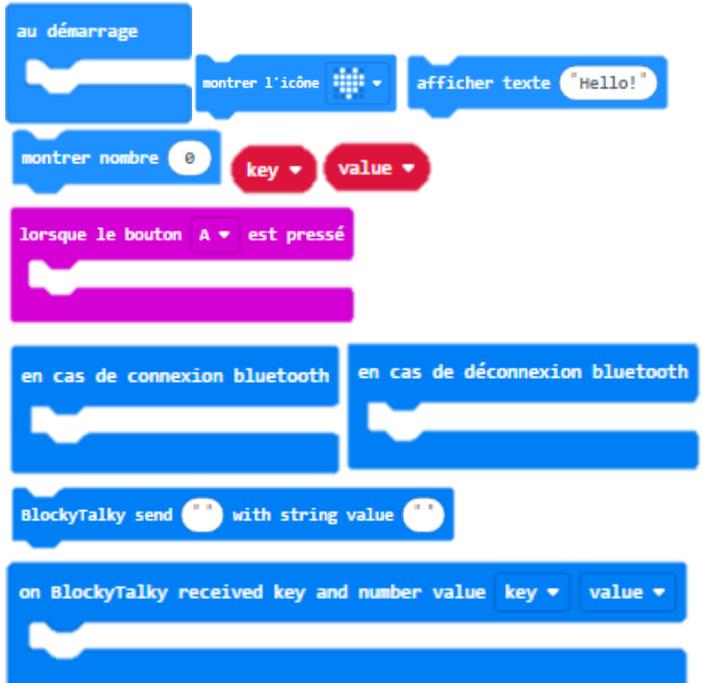
## COMMUNICATION BLUETOOTH

Permet de transmettre des données au travers d'une liaison sans fil entre un smartphone Android et la carte micro:bit. Les applications proposées sont réalisées sous App Inventor 2.

Le fichier « Documentdebase.aia » sert de base pour coder une application sur AppInventor. Il contient toute la partie connexion/déconnexion de la carte CODO par Bluetooth.

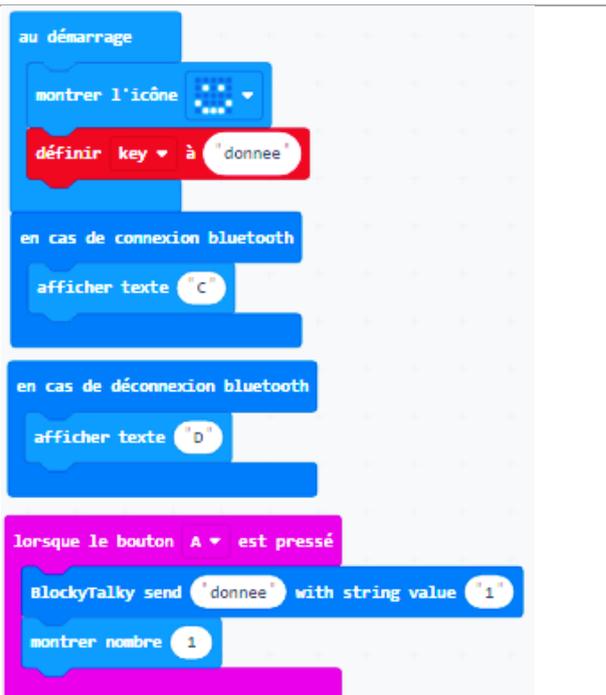
Pour chaque exercice, nous proposons 3 fichiers de correction :

- un fichier **.hex** à charger dans la carte micro:bit.
- un fichier **.apk** à installer sur le smartphone.
- le fichier **.aia** modifiable avec AppInventor.

Bluetooth	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"><li>Base</li><li>Entrées</li><li>Variables</li><li>Bluetooth</li><li>Blockytalky</li></ul>	 <p>The code blocks in the App Inventor interface include: 'au démarrage' (when started) with 'montrer l'icône' (show icon) and 'afficher texte "Hello!"' (show text); 'montrer nombre' (show number) with 'key' and 'value' dropdowns; 'lorsque le bouton A est pressé' (when button A is pressed); 'en cas de connexion bluetooth' (when bluetooth connected) and 'en cas de déconnexion bluetooth' (when bluetooth disconnected); 'BlockyTalky send' with 'with string value'; and 'on BlockyTalky received key and number value' with 'key' and 'value' dropdowns.</p>

**Niveau 1 : Communication de la carte micro:bit vers l'application : lorsque vous appuyez sur le bouton A, afficher "1" sur la carte et sur l'application.**

Avec MakeCode : codo-BLUETOOTH-EX1.hex



The MakeCode code blocks are: 'au démarrage' (when started) with 'montrer l'icône' (show icon) and 'définir key à "donnee"' (define key to 'donnee'); 'en cas de connexion bluetooth' (when bluetooth connected) with 'afficher texte "C"' (show text 'C'); 'en cas de déconnexion bluetooth' (when bluetooth disconnected) with 'afficher texte "D"' (show text 'D'); and 'lorsque le bouton A est pressé' (when button A is pressed) with 'BlockyTalky send "donnee" with string value "1"' (send 'donnee' with string value '1') and 'montrer nombre 1' (show number 1).

**Niveau 2 : Communication de l'application vers la carte micro:bit - Lorsque vous appuyez sur un bouton dans l'application, un chiffre est envoyé et affiché sur la carte.**

Avec MakeCode : codo-BLUETOOTH-EX2.hex

```
au démarrage
  montrer l'icône
  définir key à "donnee"

en cas de connexion bluetooth
  afficher texte "C"

en cas de déconnexion bluetooth
  afficher texte "D"

on BlocklyTalky received key and number value key value
  montrer nombre value
```

**Niveau 3 Communication bidirectionnelle – Lorsque vous appuyez sur le bouton A, le chiffre 3 apparaît sur l'application. Lorsque vous appuyez sur le bouton « Envoyer une valeur » dans l'application, le chiffre 3 disparaît et s'affiche sur la carte.**

Avec MakeCode : codo-BLUETOOTH-EX3.hex

```
au démarrage
  montrer l'icône
  définir key à "donnee"

en cas de connexion bluetooth
  afficher texte "C"

en cas de déconnexion bluetooth
  afficher texte "D"

on BlocklyTalky received key and number value key value
  montrer nombre value

lorsque le bouton A est pressé
  effacer l'écran
  BlocklyTalky send "donnee" with string value "3"
```

# Programmation avec des modules Grove

Des modules Grove peuvent être connectés avec un câble Grove à la carte micro:bit grâce à la carte CODO présentée précédemment.

Modules	Enoncés	Corrections
<b>LED</b> 	Niv1 : activer la LED pendant 5 secondes puis l'éteindre.	codo-led-EX1
	Niv2 : faire clignoter la LED toutes les 0,5 secondes.	codo-led-EX2
	Niv3 : faire varier la vitesse de clignotement de la LED.	codo-led-EX3
<b>Bouton-poussoir</b> 	Niv1 : lorsque l'on appuie sur le bouton poussoir, montrer une icône au choix, sinon ne rien montrer.	codo-BP-EX1
	Niv2 : lorsque le bouton poussoir est pressé, allumer la LED et l'éteindre lorsqu'il est relâché.	codo-BP-EX2
	Niv3 : éteindre la LED 3 secondes après avoir relâché le bouton.	codo-BP-EX3
<b>Afficheur 4 digits</b> 	Niv1 : afficher "1234".	codo-digits-EX1
	Niv2 : afficher la valeur du capteur de température.	codo-digits-EX2
	Niv3 : afficher tous les caractères possibles en les faisant défiler chacun leur tour.	codo-digits-EX3
<b>Télémètre à ultrasons</b> 	Niv1 : activer la LED si la distance mesurée est inférieure à 20 cm sinon la désactiver.	codo-ULTRA-EX1
	Niv2 : activer la LED si la distance mesurée est inférieure à 20 cm et la réactiver quand l'objet détecté s'est éloigné de plus de 25 cm.	codo-ULTRA-EX2
	Niv3 : faire clignoter la LED à une vitesse variant avec la distance mesurée.	codo-ULTRA-EX3
<b>Speaker</b> 	Niv1 : actionner le buzzer.	codo-speaker-EX1
	Niv2 : actionner le buzzer toutes les secondes.	codo-speaker-EX2
	Niv3 : jouer une mélodie une fois en accéléré et quand elle est finie allumer la LED en haut à gauche.	codo-speaker-EX3

<b>Potentiomètre</b> 	Niv1 : faire clignoter une LED à une fréquence dépendant de la valeur du potentiomètre.	codo-POT-EX1
	Niv2 : tracer le graphe de la valeur du potentiomètre.	codo-POT-EX2
	Niv3 : ramener la valeur du potentiomètre à une échelle de 0 à 100 et l'afficher sur l'afficheur 4 digits.	codo-POT-EX3
<b>Capteur de mouvement</b> 	Niv1 : lorsque l'on tourne dans le sens des aiguilles d'une montre afficher ☺ et dans le sens antihoraire afficher ☹.	codo-CM-EX1
	Niv2 : afficher le nombre de fois qu'un mouvement à droite a été effectué.	codo-CM-EX2
	Niv3 : lors d'un mouvement dans le sens horaire, allumer une LED choisie au hasard. S'il s'agit de la LED centrale, attendre 1 seconde et émettre un son.	codo-CM-EX3
<b>Neopixel</b> 	Niv1 : allumer les 30 LED en arc en ciel.	codo-Neo-EX1
	Niv2 : afficher les LED en vert lorsque le bouton A est pressé.	codo-Neo-EX2
	Niv3 : allumer 15 LED dans un mélange de rouge, vert et bleu.	codo-Neo-EX3
<b>Neopixel (suite)</b> 	Niv1 : faire une rotation des LED toutes les 0.5 secondes.	codo-Neo2-EX1
	Niv2 : allumer le nombre de LED égal au nombre de lettres du mot "Bonjour" en orange.	codo-Neo2-EX2
	Niv3 : faire apparaître un arc en ciel des LED et faire varier les couleurs à l'aide du potentiomètre.	codo-Neo2-EX3

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li><span style="color: blue;">■</span> Base</li> <li><span style="color: red;">■</span> Broches</li> <li><span style="color: red;">■</span> Variables</li> </ul>	

## Niv1 : activer la LED pendant 5 secondes, puis l'éteindre

Avec MakeCode : codo-led-EX1.hex	Avec mu Python Editor : codo-led-EX1.py
	<pre> 1 from microbit import * 2 3 pin0.write_digital(1) 4 sleep(5000) 5 pin0.write_digital(0) </pre>

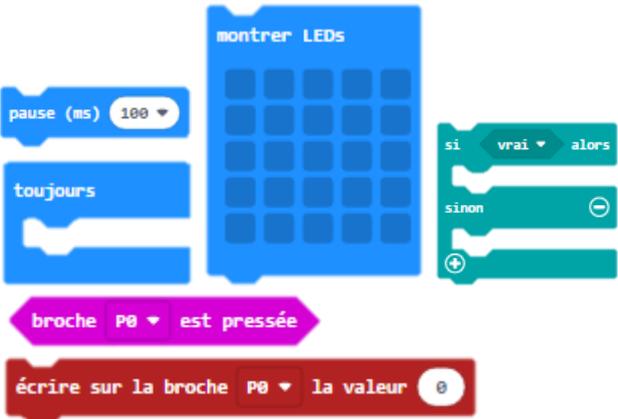
## Niv2 : faire clignoter la LED toutes les 0,5 secondes

Avec MakeCode : codo-led-EX2.hex	Avec mu Python Editor : codo-led-EX2.py
	<pre> 1 from microbit import * 2 3 while True: 4     pin0.write_digital(1) 5     sleep(500) 6     pin0.write_digital(0) 7     sleep(500) </pre>

### Niv3 : faire varier la vitesse de clignotement de la LED

Avec MakeCode : codo-led-EX3.hex	Avec mu Python Editor : codo-led-EX3.py
	<pre data-bbox="858 197 1380 459">1 from microbit import * 2 3 while True: 4     for i in range(0, 5000, 50): 5         pin0.write_digital(1) 6         sleep(i) 7         pin0.write_digital(0) 8         sleep(i)</pre>

# BOUTON-POUSSOIR

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li><span style="color: blue;">■</span> Base</li> <li><span style="color: magenta;">■</span> Entrées</li> <li><span style="color: teal;">■</span> Logique</li> <li><span style="color: red;">■</span> Broches</li> </ul>	

**Niv1 : lorsque l'on appuie sur le bouton-poussoir, montrer une icône au choix, sinon ne rien montrer**

Avec MakeCode : codo-BP-EX1.hex	Avec mu Python Editor : codo-BP-EX1.py
	<pre> 1 from microbit import * 2 3 while True: 4     if pin12.read_digital()==1: 5         display.show(Image.PITCHFORK) 6         sleep(1000) 7         display.clear()         </pre>

**Niv2 : lorsque l'on appuie sur le bouton-poussoir, allumer la LED et l'éteindre lorsqu'il est relâché**

Avec MakeCode : codo-BP-EX2.hex	Avec mu Python Editor : codo-BP-EX2.py
	<pre> 1 from microbit import * 2 3 while True: 4     if pin12.read_digital()==1: 5         pin0.write_digital(1) 6     else: 7         pin0.write_digital(0)         </pre>

**Niv3 : éteindre la LED 3 secondes après avoir relâché le bouton**

Avec MakeCode : codo-BP-EX3.hex	Avec mu Python Editor : codo-BP-EX3.py
---------------------------------	--

```
toujours
si broche P1 est pressée alors
  écrire sur la broche P0 la valeur 0
  écrire sur la broche P0 la valeur 1
  pause (ms) 3000
sinon
```

```
1 from microbit import *
2
3 while True:
4     if pin12.read_digital()==1:
5         pin0.write_digital(1)
6     else:
7         sleep(3000)
8         pin0.write_digital(0)
```

# AFFICHEUR 4 DIGITS

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li><span style="color: blue;">■</span> Base</li> <li><span style="color: magenta;">■</span> Entrées</li> <li><span style="color: blue;">■</span> CODO</li> <li><span style="color: red;">■</span> Variables</li> </ul>	

## Niv1 : afficher "1234"

Avec MakeCode : codo-digits-EX1.hex



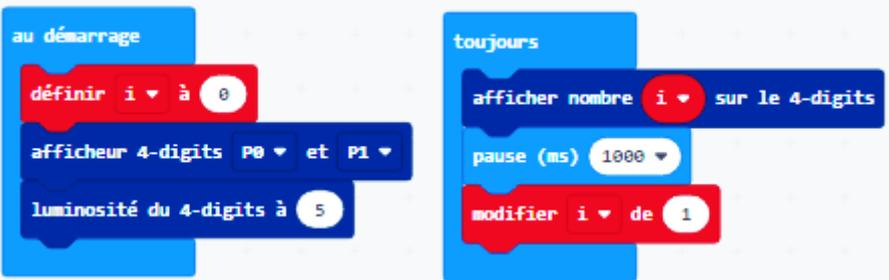
## Niv2 : afficher la valeur du capteur de température

Avec MakeCode : codo-digits-EX2.hex



## Niv3 : afficher tous les caractères possibles en les faisant défiler chacun leur tour

Avec MakeCode : codo-digits-EX3.hex

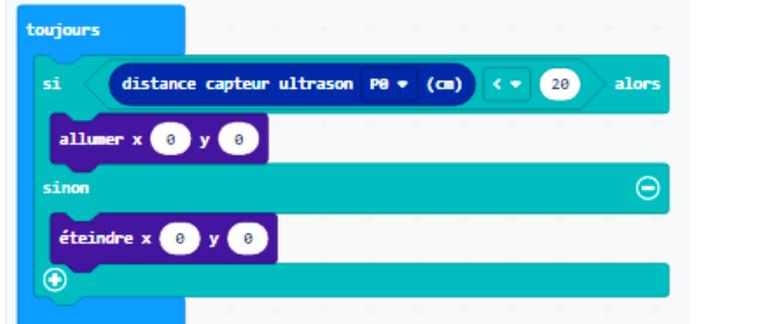


## TELEMETRE A ULTRASONS

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> Base</li> <li> LED</li> <li> Logique</li> <li> CODO</li> </ul>	 <p>The code snippet shows a 'toujours' loop containing a 'pause (ms) 100' block, an 'allumer x 0 y 0' block, an 'si vrai alors' block with a 'distance capteur ultrason P0 (cm)' block and a '&lt;' comparison, and a 'sinon' block with an 'éteindre x 0 y 0' block.</p>

**Niv1 : activer le module LED si la distance mesurée est inférieure à 20 cm sinon la désactiver**

Avec MakeCode : codo-ULTRA-EX1.hex



The code for Niv1 is a 'toujours' loop containing an 'si' block with the condition 'distance capteur ultrason P0 (cm) < 20'. The 'alors' branch contains an 'allumer x 0 y 0' block, and the 'sinon' branch contains an 'éteindre x 0 y 0' block.

**Niv2 : activer le module LED si la distance mesurée est inférieure à 20 cm et la réactiver quand l'objet détecté s'est éloigné de plus de 25 cm**

Avec MakeCode : codo-ULTRA-EX2.hex



The code for Niv2 is a 'toujours' loop containing an 'si' block with the condition 'distance capteur ultrason P1 (cm) < 20 ou distance capteur ultrason P1 (cm) > 25'. The 'alors' branch contains an 'allumer x 0 y 0' block, and the 'sinon' branch contains an 'éteindre x 0 y 0' block.

**Niv3 : faire clignoter le module LED à une fréquence variant avec la distance mesurée.**

Avec MakeCode : codo-ULTRA-EX3.hex



The code for Niv3 is a 'toujours' loop containing an 'allumer x 0 y 0' block, a 'pause (ms) distance capteur ultrason P1 (cm)' block, an 'éteindre x 0 y 0' block, and another 'pause (ms) distance capteur ultrason P1 (cm)' block.

# SPEAKER

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> Base</li> <li> Musique</li> </ul>	

## Niv1 : actionner le buzzer

Avec MakeCode : codo-speaker-EX1.hex	Avec mu Python Editor : codo-speaker-EX1.py
	<pre data-bbox="863 831 1294 1046"> 1 from microbit import * 2 import music 3 4 tune = "C4:1" 5 6 music.play(tune) </pre>

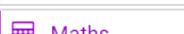
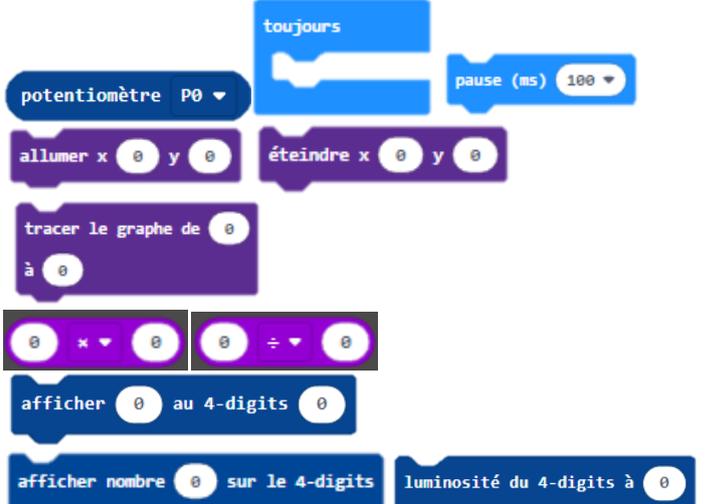
## Niv2 : actionner le buzzer toutes les secondes

Avec MakeCode : codo-speaker-EX2.hex	Avec mu Python Editor : codo-speaker-EX2.py
	<pre data-bbox="863 1198 1294 1467"> 1 from microbit import * 2 import music 3 4 tune = "C4:1" 5 6 while True: 7     music.play(tune) 8     sleep(1000) </pre>

## Niv3 : jouer une mélodie une fois en accéléré et quand elle est finie allumer la LED en haut à gauche

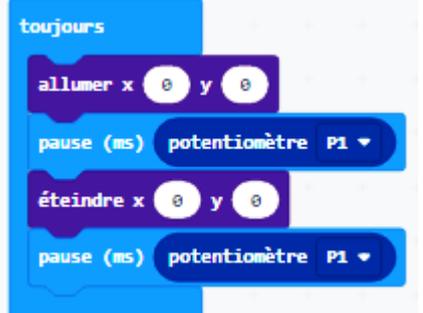
Avec MakeCode : codo-speaker-EX3.hex	Avec mu Python Editor : codo-speaker-EX3.py
	<pre data-bbox="863 1677 1517 1998"> 1 from microbit import * 2 import music 3 4 tempo = music.get_tempo() 5 result = tempo[0] + int(400) 6 result = 0 if result &lt; 0 else result 7 music.set_tempo(bpm=result) 8 music.play(music.PRELUDE) 9 display.set_pixel(int(0), int(0), 9) </pre>

# POTENTIOMETRE

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> CODO</li> <li> Base</li> <li> LED</li> <li> Maths</li> </ul>	

**Niv1 : faire clignoter une LED à une fréquence dépendant de la valeur du potentiomètre**

Avec MakeCode : codo-POT-EX1\_v2.hex



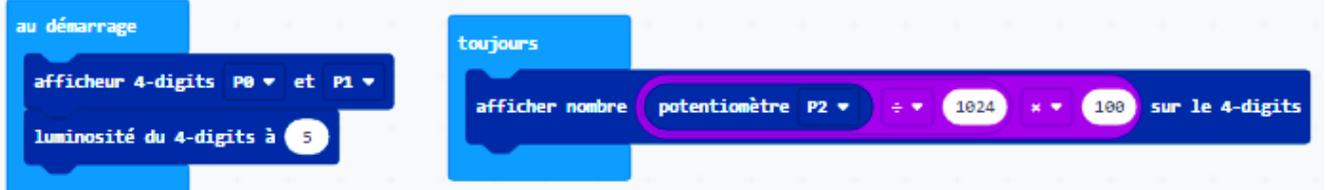
**Niv2 : tracer le graphe de la valeur du potentiomètre**

Avec MakeCode : codo-POT-EX2.hex



**Niv3 : ramener la valeur du potentiomètre à une échelle de 0 à 100 et l'afficher sur l'afficheur**

Avec MakeCode : codo-POT-EX3.hex



## CAPTEUR DE MOUVEMENT

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> Base</li> <li> Logique</li> <li> CODO</li> </ul>	<p><b>Valeurs renvoyées par le capteur :</b></p> <ul style="list-style-type: none"> <li>Aucun : 0</li> <li>Droite : 1</li> <li>Gauche : 2</li> <li>Vers le haut : 3</li> <li>Vers le bas : 4</li> <li>Tout droit : 5</li> <li>En arrière : 6</li> <li>Sens horaire : 7</li> <li>Sens antihoraire : 8</li> <li>Vague : 9</li> </ul>

**Niv1 :** lorsque l'on tourne dans le sens des aiguilles d'une montre afficher ☺ et dans le sens antihoraire afficher ☹

Avec MakeCode : codo-CM-EX1.hex

```

toujours
  si geste = 7 alors
    montrer l'icône ☺
  sinon
    si geste = 8 alors
      montrer l'icône ☹
    sinon
      effacer l'écran
  
```

**Niv2 :** afficher le nombre de fois qu'un mouvement à droite a été effectué

Avec MakeCode : codo-CM-EX2.hex

```

au démarrage
  définir Nb mouvements à 0
  montrer nombre Nb mouvements

toujours
  si geste = 1 alors
    modifier Nb mouvements de 1
    montrer nombre Nb mouvements
  
```

**Niv3 : lors d'un mouvement dans le sens horaire, allumer une LED choisie au hasard. S'il s'agit de la LED centrale, attendre 1 seconde et émettre un son.**

Avec MakeCode : codo-CM-EX3.hex

```
Scratch code description:  
- Loop: 'toujours' (forever)  
- Conditional: 'si geste = 7 alors' (if gesture is 7 then)  
- Random selection: 'définir Colonne à choisir au hasard de 0 à 4' and 'définir Ligne à choisir au hasard de 0 à 4'  
- Action: 'allumer x Colonne y Ligne' (turn on LED at column, line)  
- Conditional: 'si allumée à x 2 y 2 alors' (if LED at (2,2) is on then)  
- Action: 'pause (ms) 1000' (wait 1000ms)  
- Action: 'buzz (Hz) Middle C' (play Middle C sound)  
- Expansion: Two empty 'ajouter un bloc' (add block) slots are visible below the main code.
```

Module	Instructions / Fonctions utilisées	au démarrage	lorsque le bouton A est pressé
	<ul style="list-style-type: none"> <li>Base</li> <li>Variables</li> <li>Neopixel</li> <li>Entrées</li> </ul>		
			
			
			

## Niv1 : allumer les 30 LED en arc en ciel

Avec MakeCode : codo-Neo-EX1.hex



## Niv2 : afficher les LED en vert lorsque le bouton A est pressé

Avec MakeCode : codo-Neo-EX2.hex



Avec mu Python Editor : codo-Neo-EX2.py

```

1 from microbit import *
2 import neopixel
3 np = neopixel.NeoPixel(pin0, 30)
4 i = 0
5
6 while True:
7     if button_a.is_pressed():
8         for k in range(0, 29, 1):
9             np[i] = (0, 255, 0)
10            np.show()
11            i = i + 1
12

```

### Niv3 : allumer 15 LED dans un mélange de rouge, vert et bleu

Avec MakeCode : codo-Neo-EX3.hex



Avec mu Python Editor : codo-Neo-EX3.py

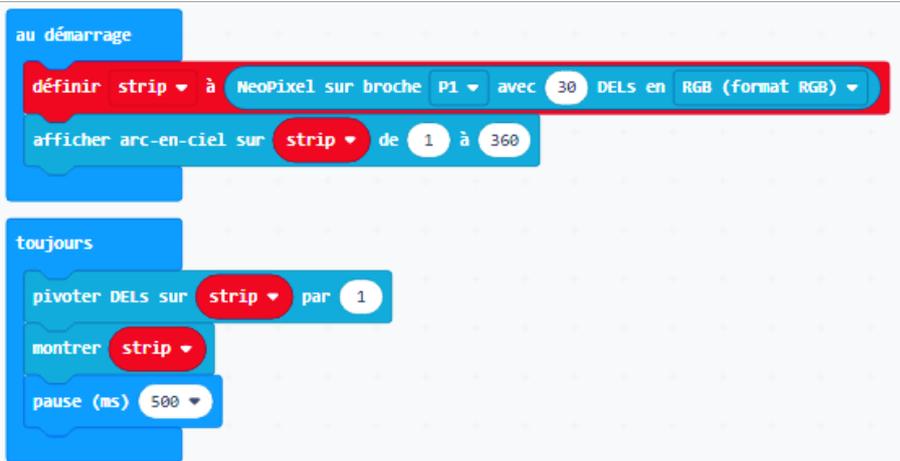
```
1 from microbit import *
2 import neopixel
3 np = neopixel.NeoPixel(pin0, 15)
4 i = 0
5
6 for k in range(0, 15, 1):
7     np[i] = (100, 150, 200)
8     np.show()
9     i = i + 1
```

## NEOPIXEL (suite)

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"><li>Base</li><li>Variables</li><li>Neopixel</li></ul>	<p>au démarrage</p> <p>définir strip à NeoPixel sur broche P0 avec 24 DELS en RGB (format RGB)</p> <p>afficher arc-en-ciel sur strip de 1 à 360</p> <p>régler couleur sur strip sur rouge</p>

### Niv1 : faire une rotation des LED toutes les 0,5 secondes

Avec MakeCode : codo-Neo2-EX1.hex



```
au démarrage
définir strip à NeoPixel sur broche P1 avec 30 DELS en RGB (format RGB)
afficher arc-en-ciel sur strip de 1 à 360

toujours
  pivoter DELS sur strip par 1
  montrer strip
  pause (ms) 500
```

Avec mu Python Editor : codo-Neo2-EX1.py

```
1 from microbit import *
2 import neopixel
3
4 i = 0
5 np = neopixel.NeoPixel(pin0, 30)
6
7 while True:
8     for k in range(0, 29, 1):
9         np[i] = (255, 0, 0)
10        np.show()
11        sleep(500)
12        i = i + 1
13    for n in range(29, 0, -1):
14        np[i] = (255, 130, 70)
15        np.show()
16        sleep(500)
17        i = i - 1
18    np.clear()
```

## Niv2 : allumer en orange autant de LED qu'il y a de lettres dans le mot « bonjour »

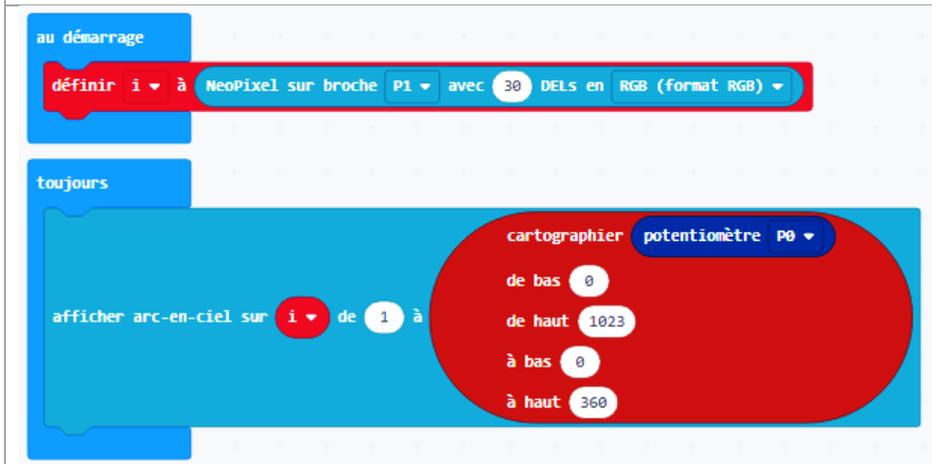
Avec MakeCode : codo-Neo2-EX2.hex



```
au démarrage
définir string à longueur de "Bonjour"
définir strip à NeoPixel sur broche P1 avec string DELS en RGB (format RGB)
régler couleur sur strip sur orange
```

## Niv3 : faire apparaître un arc en ciel des LED et faire varier les couleurs à l'aide du potentiomètre

Avec MakeCode : codo-Neo2-EX3.hex

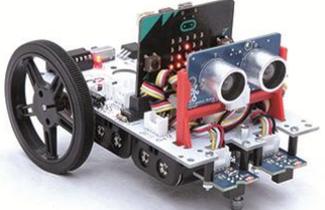


```
au démarrage
définir i à NeoPixel sur broche P1 avec 30 DELS en RGB (format RGB)

toujours
cartographier potentiomètre P0
de bas 0
de haut 1023
à bas 0
à haut 360
afficher arc-en-ciel sur i de 1 à
```

# Programmer avec les options CODO

## MOTORISATION

Option motorisation	Instructions / Fonctions utilisées	
	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;">  Base         </div> <div style="border: 1px solid #ccc; padding: 5px;">  CODO         </div>	

**Niv1 : faire avancer le robot pendant 1 seconde à sa vitesse moyenne**

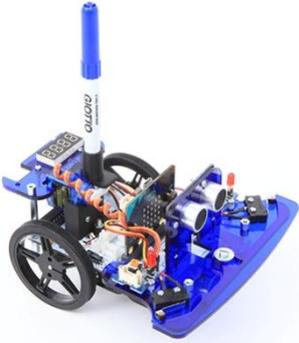
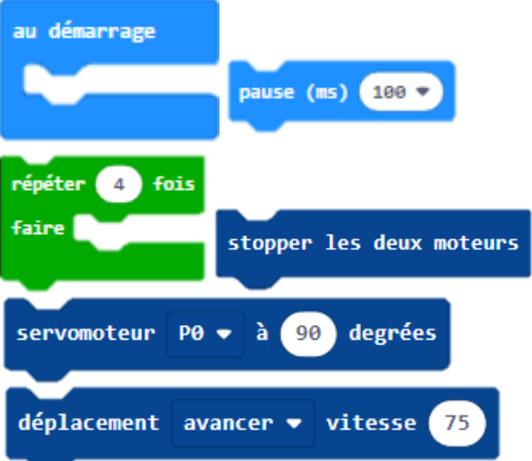
Avec MakeCode : codo-ROBOT-EX1	Avec mu Python Editor : codo-ROBOT-EX1.py
	<pre> 1 from microbit import * 2 3 pin14.write_analog(512) 4 pin13.write_analog(0) 5 pin16.write_analog(512) 6 pin15.write_analog(0) 7 sleep(1000) 8 pin14.write_analog(0) 9 pin13.write_analog(0) 10 pin16.write_analog(0) 11 pin15.write_analog(0) </pre>

**Niv2 : faire reculer le robot pendant 1 seconde à sa vitesse moyenne**

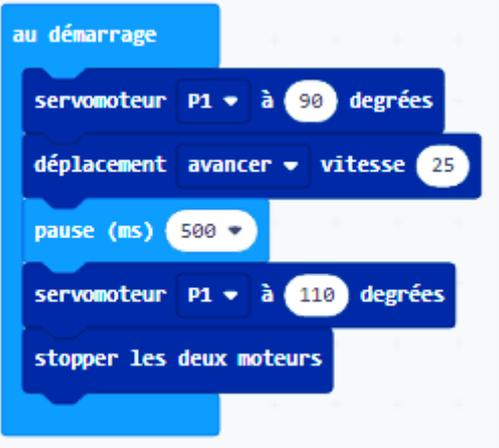
Avec MakeCode : codo-ROBOT-EX2	Avec mu Python Editor : codo-ROBOT-EX2.py
	<pre> 1 from microbit import * 2 3 pin13.write_analog(512) 4 pin14.write_analog(0) 5 pin15.write_analog(512) 6 pin16.write_analog(0) 7 sleep(1000) 8 pin13.write_analog(0) 9 pin14.write_analog(0) 10 pin15.write_analog(0) 11 pin16.write_analog(0) </pre>

### Niv3 : faire avancer le robot pendant 0,5 seconde puis le faire tourner à droite

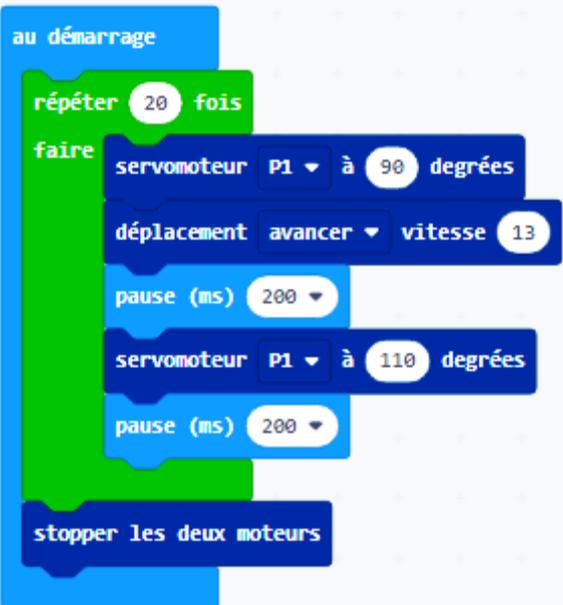
Avec MakeCode : codo-ROBOT-EX3	Avec mu Python Editor : codo-ROBOT-EX3.py
	<pre data-bbox="845 194 1474 768">1 from microbit import * 2 3 pin14.write_analog(512) 4 pin13.write_analog(0) 5 pin16.write_analog(512) 6 pin15.write_analog(0) 7 sleep(500) 8 pin14.write_analog(512) 9 pin13.write_analog(0) 10 pin16.write_analog(0) 11 pin15.write_analog(512) 12 sleep(300) 13 pin14.write_analog(0) 14 pin13.write_analog(0) 15 pin16.write_analog(0) 16 pin15.write_analog(0)</pre>

Option porte-stylo	Instructions / Fonctions utilisées	
	<p>Base</p> <p>CODO</p>	

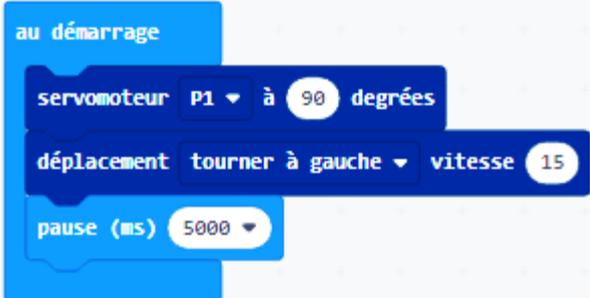
Niveau 1 : dessiner un segment

Avec MakeCode : codo-STYLO-EX1.hex	Avec mu Python Editor : codo-STYLO-EX1.py
	<pre> 1 from microbit import * 2 3 4 pin1.write_analog(70) 5 pin14.write_analog(512) 6 pin13.write_analog(0) 7 pin16.write_analog(512) 8 pin15.write_analog(0) 9 sleep(500) 10 pin1.write_analog(90) 11 pin14.write_analog(0) 12 pin13.write_analog(0) 13 pin16.write_analog(0) 14 pin15.write_analog(0) </pre>

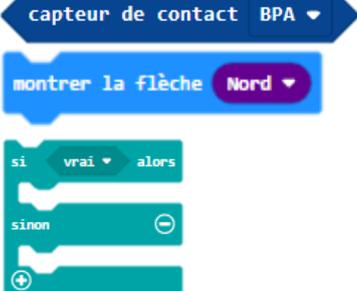
Niveau 2 : dessiner des traits en pointillés

Avec MakeCode : codo-STYLO-EX2.hex	Avec mu Python Editor : codo-STYLO-EX2.py
	<pre> 1 from microbit import * 2 3 while True: 4     pin1.write_analog(70) 5     pin14.write_analog(125) 6     pin13.write_analog(0) 7     pin16.write_analog(125) 8     pin15.write_analog(0) 9     sleep(200) 10    pin1.write_analog(90) 11    pin14.write_analog(125) 12    pin13.write_analog(0) 13    pin16.write_analog(125) 14    pin15.write_analog(0) 15    sleep(100) </pre>

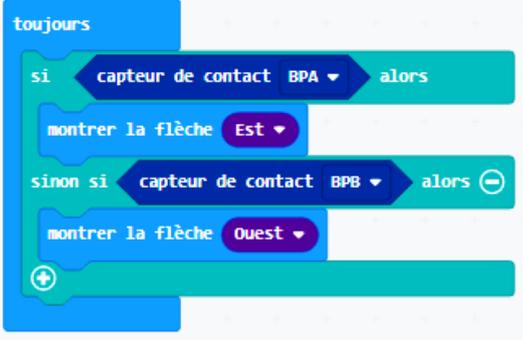
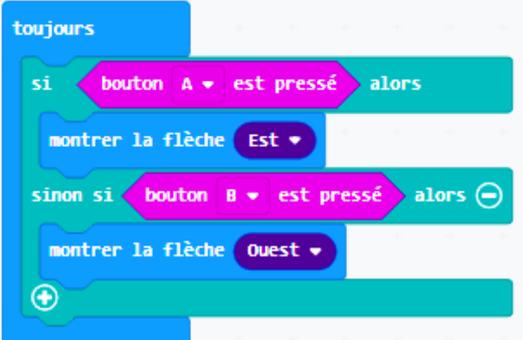
### Niveau 3 : dessiner un rond

Avec MakeCode : codo-STYLO-EX3.hex	Avec mu Python Editor : codo-STYLO-EX3.py
	<pre data-bbox="853 194 1305 479">1 from microbit import * 2 3 4 pin1.write_analog(70) 5 pin14.write_analog(255) 6 pin13.write_analog(0) 7 pin16.write_analog(0) 8 pin15.write_analog(255)</pre>

# PARE-CHOCS

Option pare-chocs	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> Base</li> <li> CODO</li> <li> Logique</li> </ul>	

**Niv1 : afficher une flèche vers l'est lorsqu'un obstacle est détecté par le pare-chocs à gauche, et une flèche vers l'ouest, lorsqu'il le détecte à droite.**

Avec MakeCode : codo-PARECHOC-EX1.hex	Avec mu Python Editor : codo-PARECHOC-EX1.py
 <p>Ou</p> 	<pre> 1  from microbit import * 2 3  while True: 4      pin14.write_analog(200) 5      pin13.write_analog(0) 6      pin16.write_analog(200) 7      pin15.write_analog(0) 8      if button_a.is_pressed(): 9          pin13.write_analog(200) 10         pin14.write_analog(0) 11         pin15.write_analog(200) 12         pin16.write_analog(0) 13         sleep(200) 14         pin14.write_analog(0) 15         pin13.write_analog(300) 16         pin16.write_analog(300) 17         pin15.write_analog(0) 18         sleep(200) 19     elif button_b.is_pressed(): 20         pin13.write_analog(200) 21         pin14.write_analog(0) 22         pin15.write_analog(200) 23         pin16.write_analog(0) 24         sleep(200) 25         pin13.write_analog(0) 26         pin14.write_analog(300) 27         pin16.write_analog(0) 28         pin15.write_analog(300) 29         sleep(200) 30 </pre>

## Niv 2 : faire une manœuvre d'évitement lorsqu'un obstacle est détecté par le pare-chocs

Avec MakeCode : codo-PARECHOC-EX2.hex

Avec mu : codo-PARECHOC-EX2.py

```

1 from microbit import *
2
3 while True:
4     if button_a.is_pressed():
5         display.show(Image("00900:"
6                             |"00090:"
7                             |"99999:"
8                             |"00090:"
9                             |"00900:"))
10
11     elif button_b.is_pressed():
12         display.show(Image("00900:"
13                             |"09000:"
14                             |"99999:"
15                             |"09000:"
16                             |"00900:"))

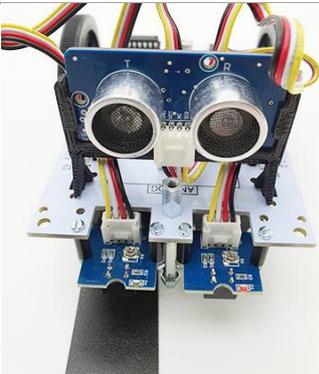
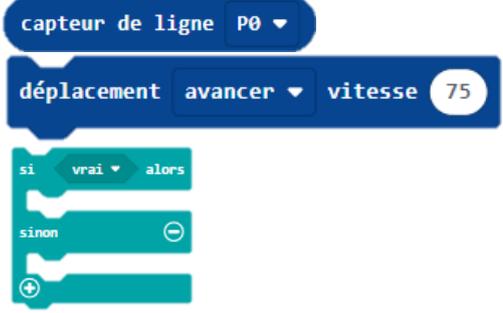
```

## Niv3 : quand le robot rencontre un obstacle à moins de 10 cm à l'aide du module ultrasons, la LED s'allume, le speaker sonne et le robot recule et tourne pour éviter l'obstacle détecté par le pare-chocs

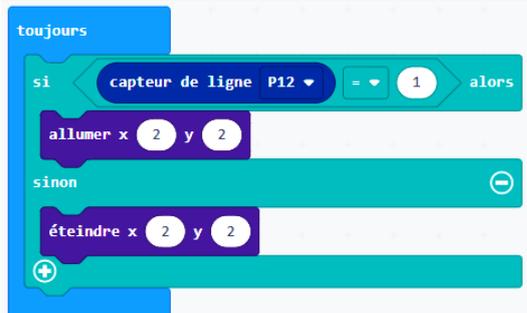
Avec MakeCode : codo-PARECHOC-EX3.hex

## DETECTION DE LIGNE

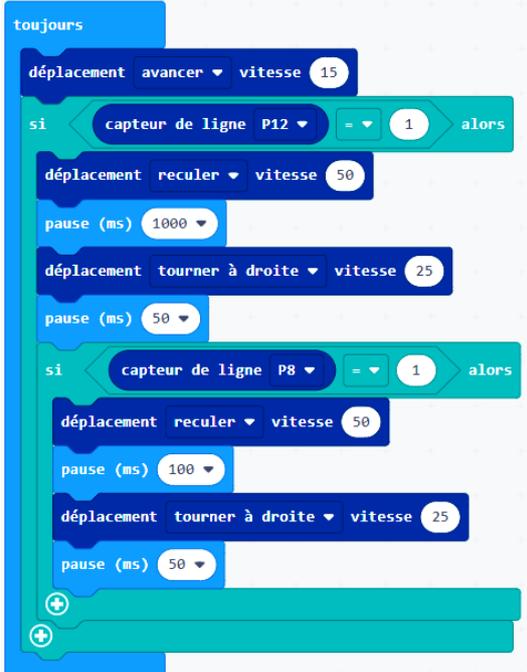
Module Grove de détection de ligne fonctionne lorsqu'il est placé entre 4mm et 15mm de la surface à analyser. Le capteur infrarouge renvoie 1 si du noir est détecté et 0 si du blanc est détecté. Il est à utiliser sur des bandes noires de 3 cm sur fond blanc.

<p><b>Détection de ligne</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li>Base</li> <li>CODO</li> <li>Logique</li> </ul>	
--	--	--

**Niv1 : allumer la LED centrale si le capteur détecte du noir et l'éteindre sinon**

<p>Avec MakeCode : codo-LIGNE-EX1.hex</p> 	<p>Avec mu Python Editor : codo-LIGNE-EX1.py</p> <pre> 1 from microbit import * 2 3 while True: 4     display.show(pin12.read_digital()) 5     sleep(1000) 6     display.clear() </pre>
--	---

**Niv2 : mettre le robot dans un circuit fermé noir. Lorsque le capteur détecte du noir il change de direction**

<p>Avec MakeCode : codo-LIGNE-EX2.hex</p> 	<p>Avec mu Python Editor : codo-LIGNE-EX2.py</p> <pre> 1 from microbit import * 2 3 4 while True: 5     if pin12.read_digital() == 0: 6         pin14.write_analog(250) 7         pin13.write_analog(0) 8         pin16.write_analog(0) 9         pin15.write_analog(250) 10    elif pin8.read_digital() == 0: 11        pin14.write_analog(0) 12        pin13.write_analog(250) 13        pin16.write_analog(250) 14        pin15.write_analog(0) 15    else: 16        pin14.write_analog(300) 17        pin13.write_analog(0) 18        pin16.write_analog(300) 19        pin15.write_analog(0) </pre>
---	---

### Niv3 : dessiner une large bande noire à la main et faire suivre le chemin au robot

Avec MakeCode : codo-LIGNE-EX3.hex

The diagram shows a Scratch-style block diagram for a robot navigation program. It starts with a 'toujours' (forever) loop block. Inside the loop, there are four conditional blocks: 1. 'si' (if) block: 'capteur de ligne P12' (line sensor P12) equals 1 and 'capteur de ligne P8' (line sensor P8) equals 1, then 'déplacement avancer' (move forward) with 'vitesse 50' (speed 50). 2. 'sinon si' (else if) block: 'capteur de ligne P12' equals 1 and 'capteur de ligne P8' equals 0, then 'déplacement tourner à droite' (turn right) with 'vitesse 50'. 3. 'sinon si' (else if) block: 'capteur de ligne P12' equals 0 and 'capteur de ligne P8' equals 1, then 'déplacement tourner à gauche' (turn left) with 'vitesse 50'. 4. 'sinon' (else) block: 'déplacement reculer' (move backward) with 'vitesse 50'. The loop ends with a '+' sign.

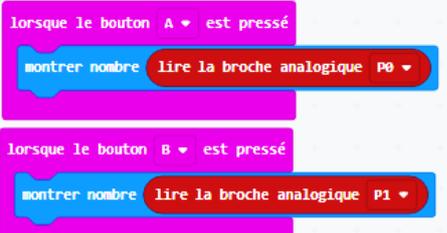
Avec mu Python Editor : codo-LIGNE-EX3.py

```
1 from microbit import *
2
3
4 while True:
5     if pin12.read_digital() == 1:
6         pin14.write_analog(250)
7         pin13.write_analog(0)
8         pin16.write_analog(0)
9         pin15.write_analog(250)
10    elif pin8.read_digital() == 1:
11        pin14.write_analog(0)
12        pin13.write_analog(250)
13        pin16.write_analog(250)
14        pin15.write_analog(0)
15    else:
16        pin14.write_analog(300)
17        pin13.write_analog(0)
18        pin16.write_analog(300)
19        pin15.write_analog(0)
```

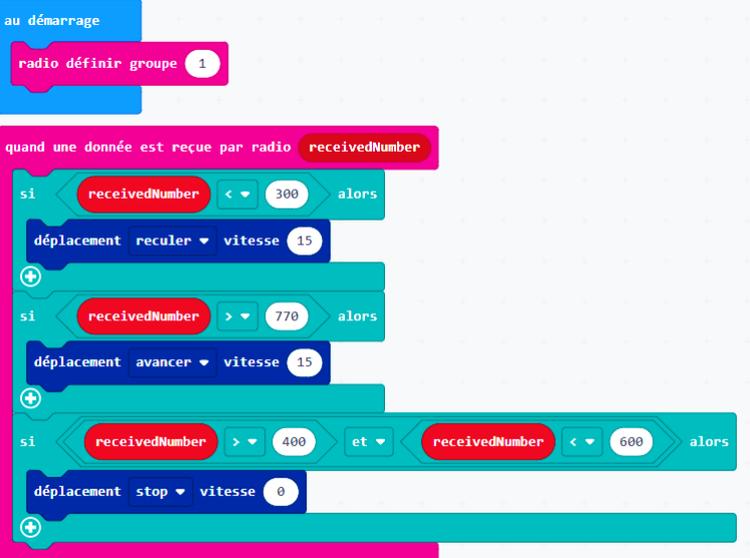
# JOYSTICK

<p><b>Joystick</b></p> 	<p><b>Instructions / Fonctions utilisées</b></p> <ul style="list-style-type: none"> <li>Base</li> <li>Broches</li> <li>Radio</li> <li>Logique</li> <li>CODO</li> </ul>	<p>montrer nombre 0</p> <p>lire la broche numérique P0</p> <p>radio définir groupe 1</p> <p>quand une donnée est reçue par radio receivedNumber</p>
--	--	---

**Niv1 : afficher les angles de rotation sur l'axe X lorsque le bouton A est pressé et afficher les angles de rotation sur l'axe Y lorsque le bouton B est pressé**

<p>Avec MakeCode : codo-JOYSTICK-EX1.hex</p> 	<p>Avec mu Python Editor : codo-JOYSTICK-EX1.py</p> <pre> 1 from microbit import * 2 3 while True: 4     if button_a.is_pressed(): 5         display.scroll(str(pin0.read_analog())) 6     elif button_b.is_pressed(): 7         display.scroll(str(pin1.read_analog())) </pre>
--	---

**Niv2 : par radio faire avancer ou reculer le robot (carte 1) à l'aide du joystick (carte 2)**

<p>Avec MakeCode : codo-JOYSTICK-EX2A.hex</p> 
<p>Avec MakeCode : codo-JOYSTICK-EX2B.hex</p> 

### Niv3 : diriger le robot dans toutes les directions par radio avec le joystick connecté à la carte 2

Avec MakeCode : codo-JOYSTICK-EX3A.hex

```
au démarrage
  radio définir groupe 1

toujours
  si lire la broche analogique P0 > 700 alors
    envoyer le nombre 1 par radio
  sinon si lire la broche analogique P0 < 300 alors
    envoyer le nombre 2 par radio
  sinon si lire la broche analogique P1 < 300 alors
    envoyer le nombre 3 par radio
  sinon si lire la broche analogique P1 > 700 alors
    envoyer le nombre 4 par radio
  sinon
    envoyer le nombre 5 par radio
```

Avec MakeCode : codo-JOYSTICK-EX3B.hex

```
au démarrage
  radio définir groupe 1

quand une donnée est reçue par radio receivedNumber
  si receivedNumber = 1 alors
    déplacement avancer vitesse 15
  si receivedNumber = 2 alors
    déplacement reculer vitesse 15
  si receivedNumber = 3 alors
    déplacement tourner à droite vitesse 15
  si receivedNumber = 4 alors
    déplacement tourner à gauche vitesse 15
  sinon
    déplacement stop vitesse 0
```

# Programmation avancée

## En blocs

Thèmes	Enoncés	Corrections
<b>Matrice LED</b>	Ex 1 : allumer les LED une par une de gauche à droite et de haut en bas.	codo-MATRICE3-EX1
	Ex 2 : reprendre l'exercice 1, en désactivant chaque LED 1 seconde après avoir été activée.	codo-MATRICE3-EX2
	Ex 3 : activer toutes les LED et éteindre certaines LED une par une de telle sorte qu'un L apparaisse.	codo-MATRICE3-EX3
<b>Minuteur</b>	Ex 1 : au lancement du programme déclencher un compte à rebours de 10 secondes et jouer un son lorsqu'il est à zéro.	codo-Minuteurs-EX1
	Ex 2 : déclencher un compte à rebours de 10 s lorsque le bouton A est pressé.	codo-Minuteurs-EX2
	Ex 3 : reprendre l'exercice 2, et ajouter la possibilité de réinitialiser le compte à rebours lorsque le bouton B est pressé.	codo-Minuteurs-EX3

## En Python

Les exercices proposés ci-dessous permettent de tracer des graphes de la température et de l'accéléromètre. **Attention** ! Ils fonctionnent uniquement avec Windows 10.

Thème	Enoncés	Corrections
<b>Graphiques</b>	Ex 1 : tracer le graphique de la T° en fonction du temps.	graph temp
	Ex 2 : tracer le graphique d'accélération suivant les 3 axes en fonction du temps.	graph accel
	Ex 3 : tracer le graphique de l'accélération suivant l'axe X en fonction du temps.	accel suivant x

Pour les exercices suivants, il s'agit d'une programmation en Python standard.

Ce sont les 6 premiers exercices de la carte micro:bit seule, adaptés en python standard.

Les motifs ne s'affichent plus sur la carte micro:bit mais à l'écran de l'ordinateur.

Ces corrections permettent d'étudier la librairie « turtle » de Python mais également de découvrir les bases du langage python standard.

<b>Matrice LED 1 python standard</b>	Ex 1 : afficher un carré.	codo-MOTIFS-EX1
	Ex 2 : afficher un carré et une seconde après un triangle.	codo-MOTIFS-EX2
	Ex 3 : répéter l'affichage d'un carré, d'un triangle, puis d'une croix avec un intervalle de 1 seconde.	codo-MOTIFS-EX3
<b>Bouton A et B python standard</b>	Exercice 1 : afficher un triangle quand le bouton A est pressé.	codo-BOUTONS-EX1
	Exercice 2 : afficher un triangle quand le bouton A est pressé et effacer l'écran après 3 secondes.	codo-BOUTONS-EX2
	Exercice 3 : afficher une croix lorsque le bouton A est pressé, le nombre 15 lorsque le bouton B est pressé et la première lettre de votre prénom lorsque les boutons A et B sont pressés	codo-BOUTONS-EX3

# MATRICE LED

Module	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> Base</li> <li> Variables</li> <li> LED</li> <li> Boucles</li> <li> Logique</li> </ul>	

Ex 1 : allumer les LED une par une de gauche à droite et de haut en bas.

Avec MakeCode : codo-MATRICE3-EX1		

**Ex 2 : reprendre l'exercice 1, en désactivant chaque LED 1 seconde après avoir été activée.**

Avec MakeCode : codo-MATRICE3-EX2

```
au démarrage
définir i à 0
définir N à 0

toujours
  tant que i < 5
    faire
      allumer x i y N
      pause (ms) 1000
      éteindre x i y N
      modifier i de 1
    si i = 5 alors
      modifier N de 1
      définir i à 0
```

**Ex 3 : activer toutes les LED et éteindre certaines LED une par une de telle sorte qu'un L apparaisse.**

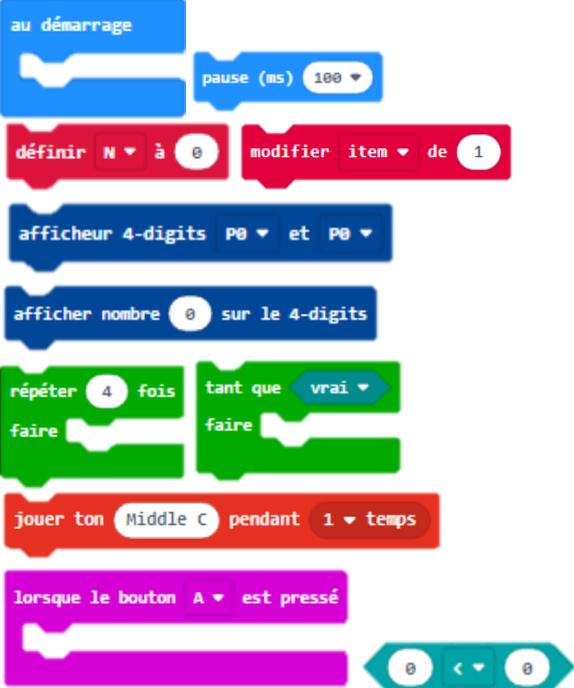
Avec MakeCode : codo-MATRICE3-EX3

The code is written in Scratch and is organized into three main sections:

- au démarrage (When green flag clicked):**
  - A "montrer LEDs" block with a 5x5 grid of LED indicators.
  - Two "définir" blocks: "i" à 0 and "N" à 0.
- toujours (Forever loop):**
  - Loop 1:** "tant que" condition:  $i < 5$  et  $N = 0$ .
    - "faire" loop:
      - "éteindre x" block: x = i, y = N.
      - "pause (ms)" block: 500.
      - "modifier i" block: de 1.
      - "si" block:  $i \geq 5$  alors:
        - "modifier N" block: de 1.
        - "définir i" block: à 0.
  - Loop 2:** "tant que" condition:  $i = 0$  et  $N < 4$ .
    - "faire" loop:
      - "éteindre x" block: x = i, y = N.
      - "pause (ms)" block: 500.
      - "modifier N" block: de 1.
      - "si" block:  $N \geq 4$  alors:
        - "modifier i" block: de 1.
        - "définir N" block: à 1.

Etc.

# MINUTEUR

Minuteur	Instructions / Fonctions utilisées	
	<ul style="list-style-type: none"> <li> Base</li> <li> Variables</li> <li> LED</li> <li> Boucles</li> <li> Logique</li> <li> Musique</li> <li> Entrées</li> </ul>	

Ex 1 : au lancement du programme déclencher un compte à rebours de 10 secondes et jouer un son lorsqu'il est à zéro.

Avec MakeCode : codo-Minuteurs-EX1	
	

**Ex 2 : déclencher un compte à rebours de 10 s lorsque le bouton A est pressé.**

Avec MakeCode : codo-Minuteurs-EX2

```
au démarrage
  luminosité du 4-digits à 7
  afficheur 4-digits P1 et P2
  définir i à 10
  afficher nombre i sur le 4-digits
  tant que i > 0
    faire
      pause (ms) 1000
      modifier i de -1
      afficher nombre i sur le 4-digits
  jouer ton Middle C pendant 1 temps
```

**Ex 3 : reprendre l'exercice 2 et ajouter la possibilité de réinitialiser le compte à rebours lorsque le bouton B est pressé**

Avec MakeCode : codo-Minuteurs-EX3

```
au démarrage
  luminosité du 4-digits à 7
  afficheur 4-digits P1 et P2
  définir i à 10
  afficher nombre i sur le 4-digits

lorsque le bouton A est pressé
  tant que i > 0
    faire
      pause (ms) 1000
      afficher nombre i sur le 4-digits
      modifier i de -1
  jouer ton Middle C pendant 1 temps

lorsque le bouton B est pressé
  définir i à 10
  afficher nombre i sur le 4-digits
```

### Ex 1 : tracer le graphique de la T° en fonction du temps

En mu Python Editor : graph temp.py

```
1 from microbit import *
2
3 while True:
4     temp = temperature()
5     print((temp,))
6     sleep(1000)
```

### Ex 2 : tracer le graphique d'accélération suivant les 3 axes en fonction du temps

En mu Python Editor : graph accel

```
1 from microbit import *
2
3 flag = accelerometer.get_values()
4 while True:
5     print(flag)
6     sleep(20)
```

### Ex 3 : tracer le graphique de l'accélération suivant l'axe X en fonction du temps

En mu Python Editor : accel suivant x

```
1 from microbit import *
2
3 flag = (accelerometer.get_x(),)
4 while True:
5     print(flag)
6     sleep(20)
```

## Ex 1 : afficher un carré

### codo-MOTIFS-EX1

```
>>> from turtle import * #on importe turtle

def carre():
    color("red")           #on définit le carré
    begin_fill()          #on choisit la couleur du carré
    #cette fonction commence à dessiner une forme
    for i in range (4) :
        down()            #on définit la forme à dessiner, on veut que cette forme se répète 4 fois
        forward(50)       # le stylo va en bas
        left(90)          # il se déplace de 50
        # tourne à gauche en angle droit
    end_fill()           # la forme est finie

def ligne():
    for i in range (1):   #on définit notre motif
        carre()           #on veut un carré donc on réalise 1 itération
        up()              # le carré se dessine
        # le stylo se lève

x=0                       #on choisit les coordonnées
y=0

for i in range (1) :      #le carré se forme
    goto(x,y)
    ligne()
    x=0
```

## Ex 2 : afficher un carré et une seconde après un triangle

### codo-MOTIFS-EX2

```
def carre():
    color("red")
    begin_fill()
    for i in range (4) :
        down()
        forward(50)
        left(90)
    end_fill()

def triangle():
    color("red")
    begin_fill()
    down()
    forward(50)
    left(135)
    forward(70.72)
    left(135)
    forward(50)
    end_fill()

def ligne1():
    for i in range (1):
        carre()
        up()

def ligne2():
    for i in range (1):
        triangle()
        up()

x=0
y=0

goto(x,y)
ligne1()
x=60
y=0
up()
time.sleep(1)
goto(x,y)
down()
ligne2()
```

### Exercice 1 : afficher un triangle quand vous appuyez sur le bouton A

codo-BOUTONS-EX1.py

```
def triangle():
    color("red")
    begin_fill()
    down()
    forward(50)
    left(135)
    forward(70.72)
    left(135)
    forward(50)
    end_fill()

def ligne2():
    for i in range (1):
        triangle()
        up()

x=0
y=0

print("oui = 1, non = 0")
bouton = int(input("voulez-vous presser le bouton A ?"))
print(bouton)

if bouton == 1:
    goto(x,y)
    ligne2()
```

### Exercice 2 : afficher un triangle quand vous appuyez sur le bouton A et effacer l'écran après 3 secondes

codo-BOUTONS-EX2.py

```
def croix():
    color("red")
    begin_fill()
    down()
    left(45)
    forward(35.36)
    up()
    x=25
    y=0
    goto(x,y)
    down()
    left(90)
    forward(35.36)
    up()
    end_fill

def ligne3():
    croix()
    up()

x=0
y=0

print("A=1, B=2, A+B=3")
bouton = int(input("Quel(s) bouton(s) voulez-vous presser?"))

if bouton == 1:
    goto(x,y)
    ligne3()
    up()

if bouton == 2:
    print("15")

if bouton == 3:
    print("L")
```

**Exercice 3 : afficher une croix lorsque vous appuyez sur le bouton A, le nombre 15 lorsque vous appuyez sur B et la première lettre de votre prénom lorsque vous appuyez sur les boutons A et B**

codo-BOUTONS-EX3.py

```
def croix() :
    color("red")
    begin_fill()
    down()
    left(45)
    forward(35.36)
    up()
    x=25
    y=0
    goto(x,y)
    down()
    left(90)
    forward(35.36)
    up()
    end_fill

def ligne3():
    croix()
    up()

x=0
y=0

print("A=1, B=2, A+B=3")
bouton = int(input("Quel(s) bouton(s) voulez-vous presser?"))

if bouton == 1:
    goto(x,y)
    ligne3()
    up()
    time.sleep(3)

if bouton == 2:
    print("15")

if bouton == 3:
    print("L")
```

# Montage des options et modules sur la CODO

## MOTORISATION

Pour transformer la carte CODO en robot qui se déplace dans toutes les directions.

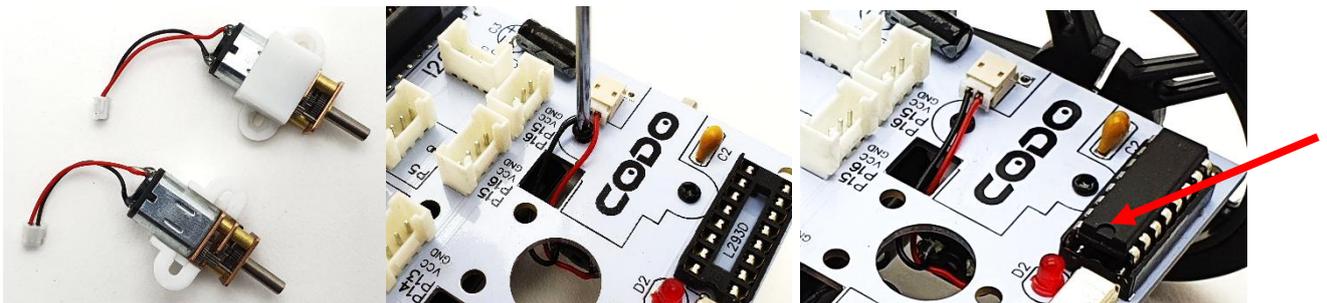
Cette option permet de mettre la carte CODO en mouvement grâce à des moteurs atteignant une vitesse de 200 tours par minute.

Elle est constituée de 2 motoréducteurs 200 tr/min 100 g.cm, 2 roues Ø 60 mm, 2 supports pour moteurs et 1 boîtier 16 pattes L293 (à fixer sur son support).

Pour la fixation : 4 vis, 4 écrous, 1 entretoise MF Ø 3 x h. 10mm, 1 entretoise MF Ø 3 x h 20 mm, 1 écrou borgne Ø3 mm et 1 entretoise nylon Ø3 x h 4 mm.

### Montage des moteurs et du circuit intégré L293D

Positionner chaque moteur dans sa bride de fixation. Mettre en place les écrous dans leurs logements hexagonaux. (Au besoin, utiliser un tournevis). Fixer et connecter chaque moteur sur la carte CODO. Enfiler le circuit intégré L293D dans son support. ATTENTION au sens ! Faire coïncider le méplat.



### Montage des roues

Mettre en place les roues. Positionner la roue à plat sur l'établi, la faire coïncider avec l'axe moteur (méplat), l'enfiler à force en appuyant sur le moteur. Vérifier que chaque roue tourne librement.



### Montage de la roulette avant

Roulette avant pivotante qui permet au robot CODOK de franchir des marches et des obstacles et d'évoluer sur des surfaces difficiles ou accidentées.

Constitué de deux pièces imprimables en 3D (fourche et roue), d'un tube laiton 3x4 coupé à L 16 mm, d'une vis TC M3 x 8, d'une vis TC M3 x 20 et d'un écrou M3.

Fichiers 3D pour impression 3D de la fourche et de la roue à télécharger sur [www.a4.fr](http://www.a4.fr)



## Montage des 2 capteurs de ligne (réf S-101020009) et carters infrarouge imprimés en 3D

Les carters infrarouges permettent d'améliorer les performances des capteurs de ligne. Il suffit de les enficher sur chaque module.



### Réglage des capteurs de ligne

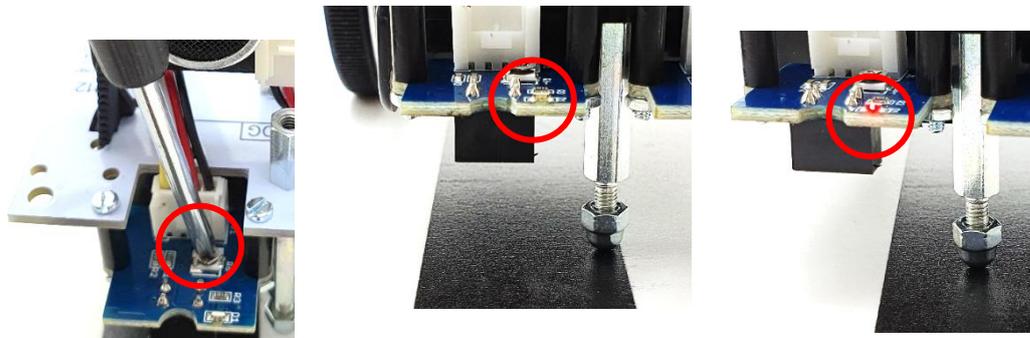
Tourner le potentiomètre 5 fois dans le sens antihoraire.

Positionner le capteur au-dessus d'une ligne noire et tourner le potentiomètre dans le sens horaire jusqu'à ce que la LED s'éteigne.

Une fois cette condition validée, positionner le capteur au-dessus d'une surface blanche et tourner le potentiomètre dans le sens horaire, jusqu'à ce que la LED témoin s'allume.

Vérifier de nouveau que les 2 conditions sont validées, sinon continuer à tourner dans le sens horaire pour affiner le réglage.

Faire la même procédure pour l'autre capteur.



Capteur correctement réglé : LED témoin est rouge quand le capteur détecte une zone claire **ET** LED témoin est éteinte quand le capteur détecte une zone sombre.

Note : les rayons infrarouges incidents peuvent perturber le fonctionnement du capteur. Eviter réglage et utilisation en plein soleil.

## PLATEFORME ROBOTIQUE

Elle est composée de 6 plaques de PMMA épaisseur 3 mm pour la fixation des différentes options et des modules Grove utiles pour aller plus loin.

### Montage des microrupteurs et du pare-chocs

2 plaques PMMA et 2 microrupteurs.

Pour la fixation des microrupteurs : 4 vis Ø2 x h12 mm, 4 écrous Ø2 mm.

Pour la fixation du pare-chocs : 1 vis Ø3 x h10 mm, 1 écrou Ø3 mm, 1 rondelle Ø12mm, 1 entretoise nylon Ø 2,2 x h 3 mm.



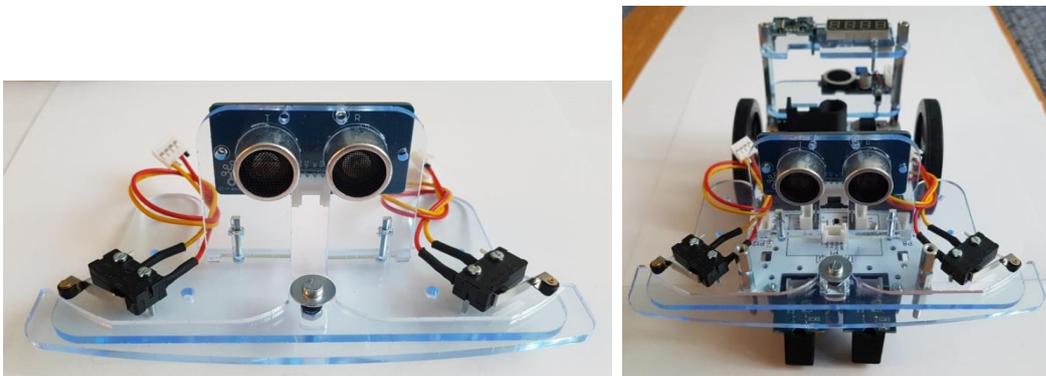
### Montage du module Ultrasons (S-101020010)

Plaque PMMA et 1 télémètre à ultrasons (non fourni).

Pour la fixation du module : 3 vis Ø2 x h 12 mm, 3 écrous Ø2 mm.

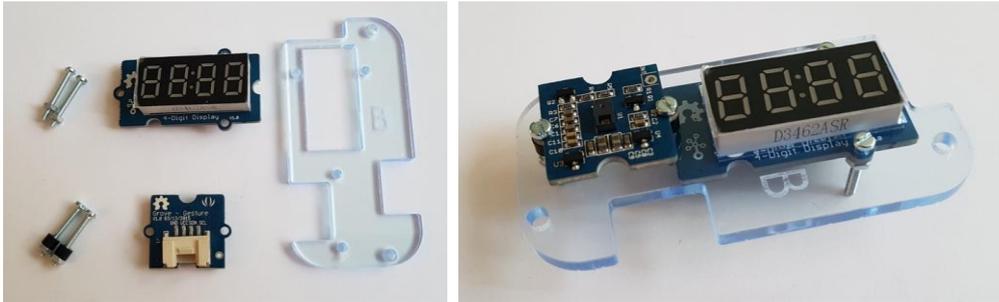


Fixation de la plaque : 2 vis Ø2 x h16 mm, 2 écrous Ø 2 mm.



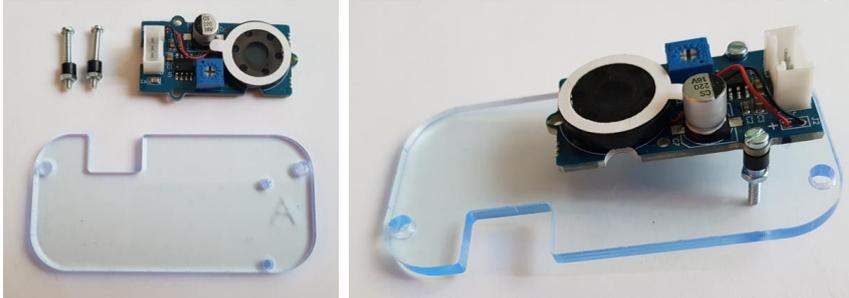
### Montage afficheur 4 digits et capteur de mouvement

Plaque B, 1 afficheur 4 digits (non fourni) et 1 capteur de mouvement (non fourni).  
Pour la fixation : 2 vis de 12 mm, 2 vis de 8 mm, 4 écrous, 2 entretoises nylon.

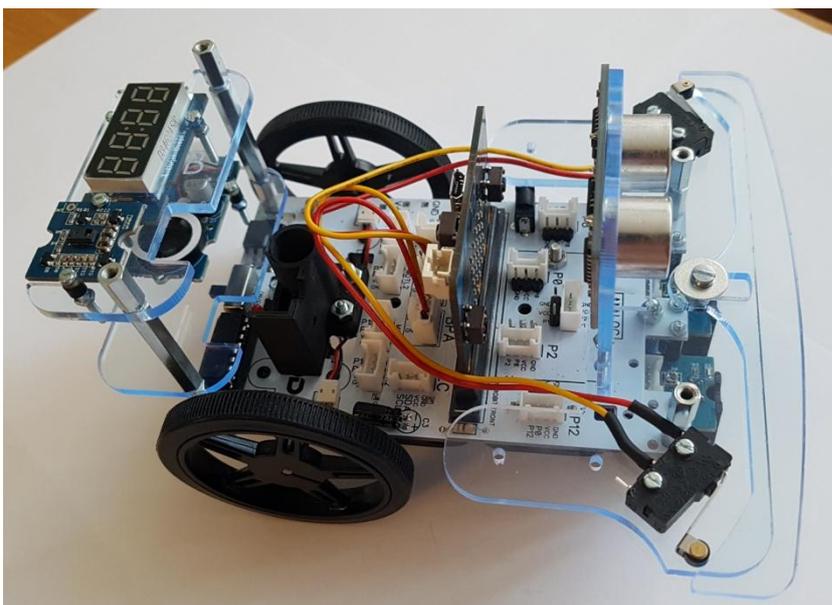
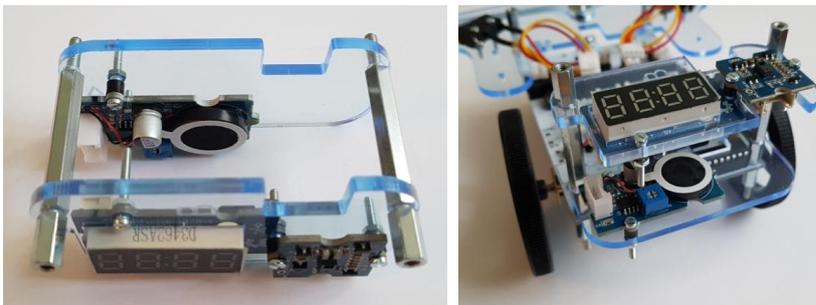


### Montage speaker

Plaque A et 1 module Speaker (non fourni).  
Pour la fixation du module : 2 vis de 12 mm, 2 écrous, 2 entretoises nylon.



Pour la fixation de l'ensemble sur la plateforme : 2 entretoises MF h30 mm, 4 entretoises MF h5 mm.



## OPTION PORTE-STYLO

---

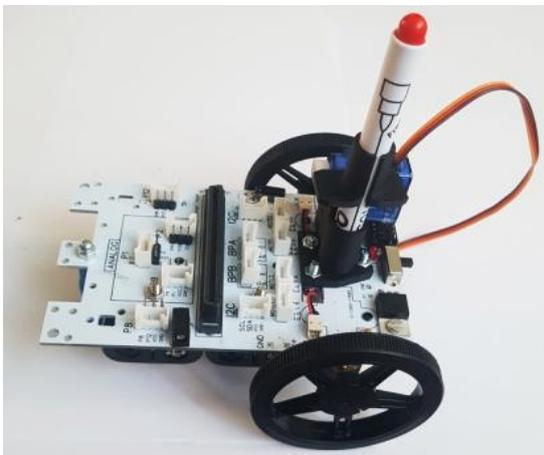
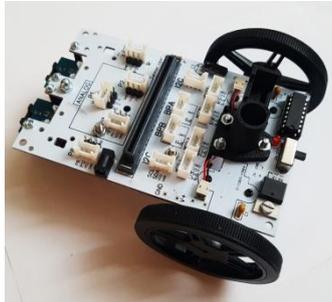
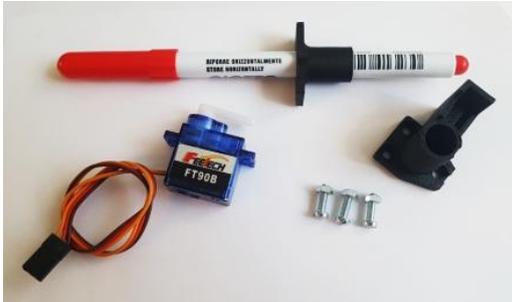
Pour transformer la carte CODO en robot dessinateur

Cette option découle de l'option avec moteur. Avec les roues, le robot peut être en mouvement.

Le stylo, piloté par un servomoteur, peut écrire et dessiner.

Elle est composée de 1 servomoteur, 2 pièces imprimées en 3D et 1 stylo.

Pour la fixation : 3 vis Ø3 x h10 mm, 3 écrous Ø3 mm et 1 vis Ø2 x h 12mm.

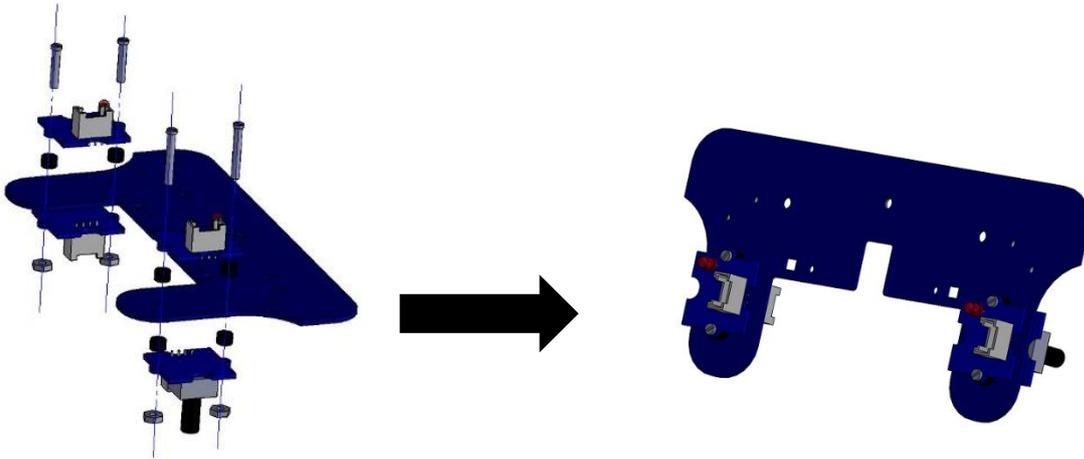


## AUTRES MODULES GROVE POUR ALLER PLUS LOIN

### LED, potentiomètre et bouton-poussoir

1 bouton-poussoir (non fourni) et 1 potentiomètre (non fourni), 2 LED (non fournis)

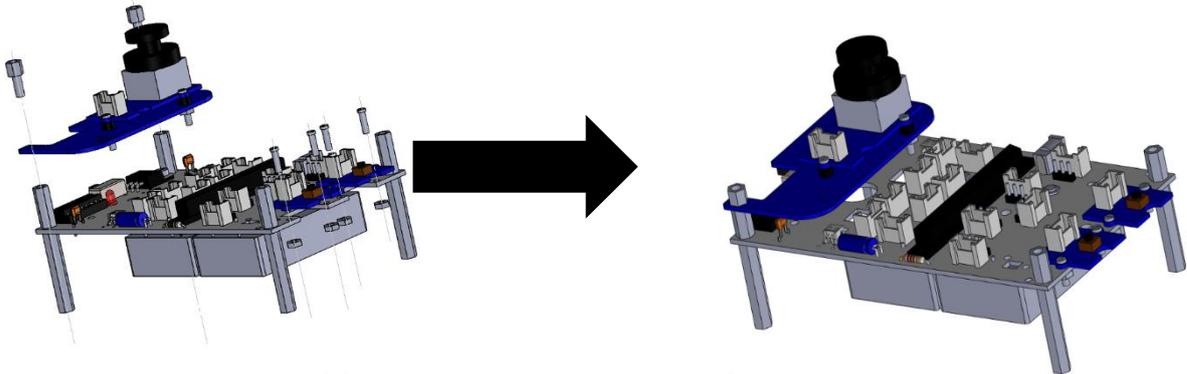
Pour la fixation : 2 vis de 12mm, 2 vis de 16 mm, 2 vis de 12mm, 4 écrous, 6 entretoises nylon.



### Joystick

La plaque C et 1 joystick (non fourni)

Pour la fixation : 2 entretoises MF de 5mm.



Note : pour concevoir une télécommande un peu plus complète deux boutons-poussoirs peuvent être fixés à l'avant avec 4 vis de 12mm et 4 écrous de diamètre 2mm.





Concepteur et fabricant de matériels pédagogiques  
[www.a4.fr](http://www.a4.fr)