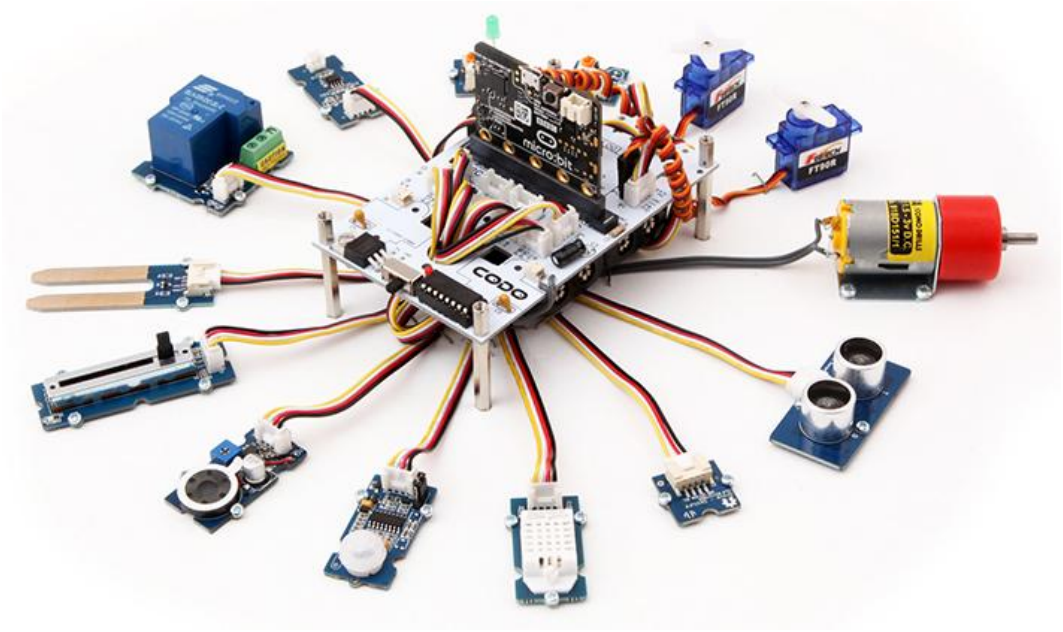
Activités de programmation
avec MakeCode et Mu Python Editor

CODO

Grove micro:bit project board



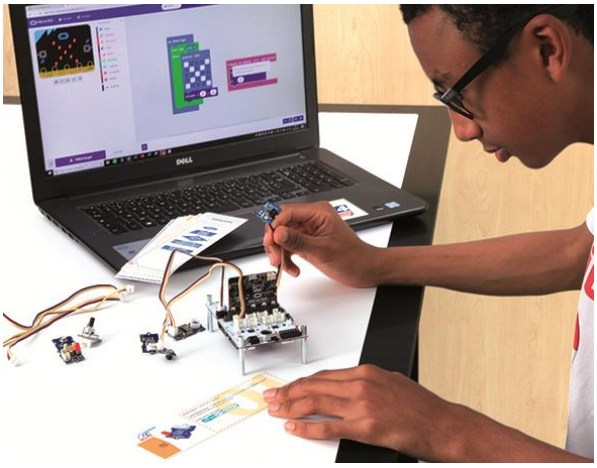
MakeCode

```
from microbit import *

k = 0

led = Image("00000:"
            "00900:"
            "00000:"
            "00000:"
            "00000:")

while True:
    if button_b.is_pressed():
        for k in range(0, 8, 1):
            display.show(led)
            sleep(500)
            display.clear()
            sleep(500)
```

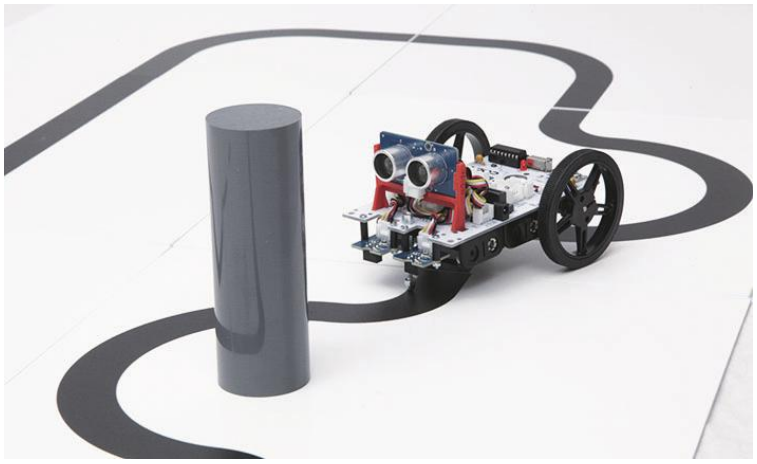
A4 TECHNOLOGIE

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** 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.

SOMMAIRE

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

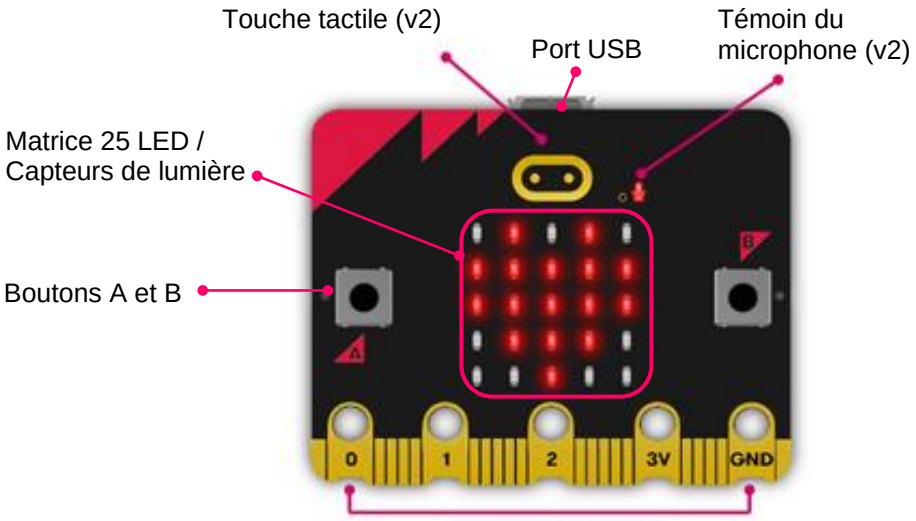
Pour l'option Bluetooth :

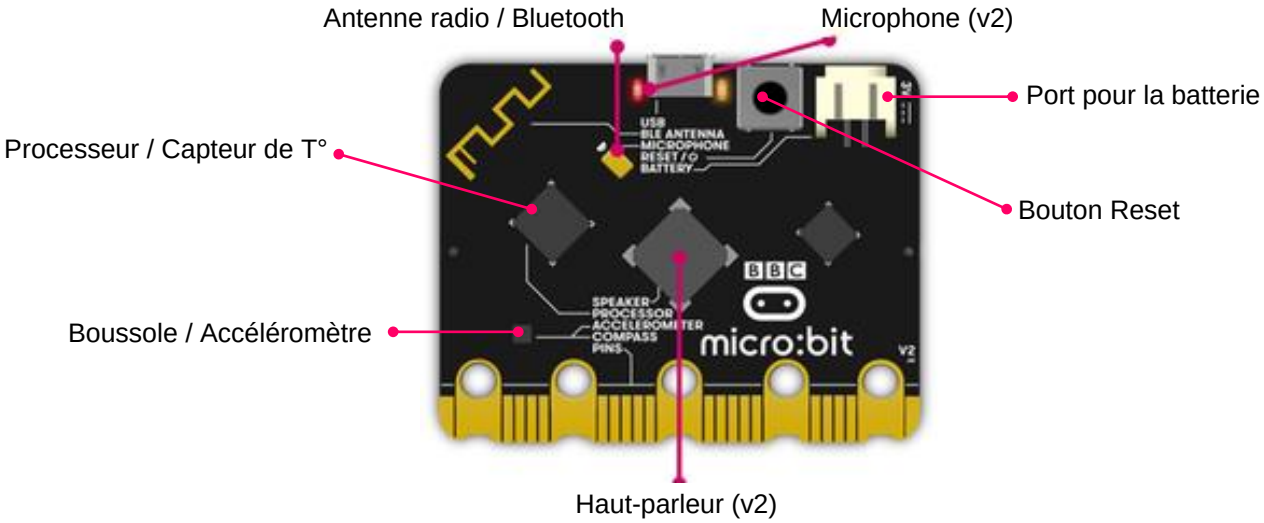
- Tablette ou smartphone Android 5 ou + équipés de Bluetooth V3.
- Connexion internet pour accéder à App Inventor **:** http://ai2.appinventor.mit.edu/
- Compte Gmail.

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.

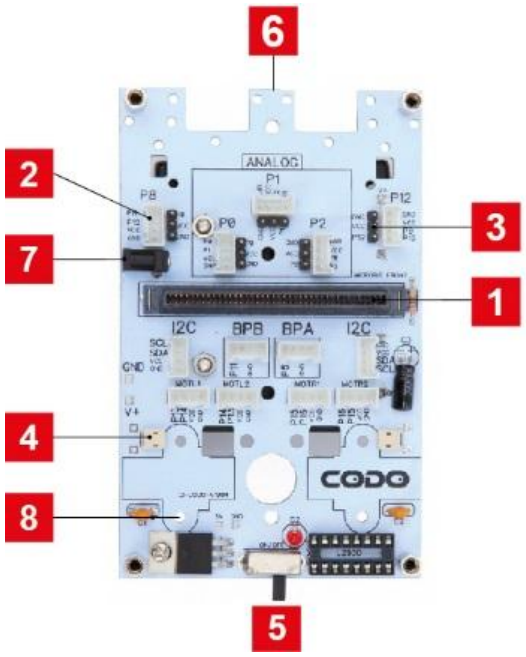




LA CARTE CODO

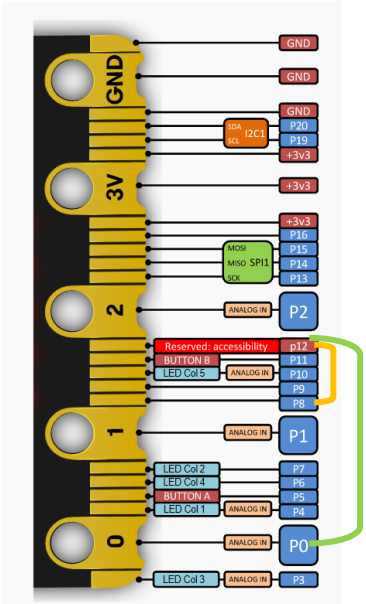
La carte CODO a été réalisée pour connecter simultanément différents modules Grove à la carte micro:bit. Elle est équipée de 13 entrées/ sorties pour modules Grove.
Elle est conçue pour être alimentée en 6V grâce à 4 emplacements pour piles 1,5V.
La carte micro:bit peut être fixée sur la carte CODO via un connecteur prévu à cet effet.
Cet assemblage est idéal pour être codé en blocs et en python.

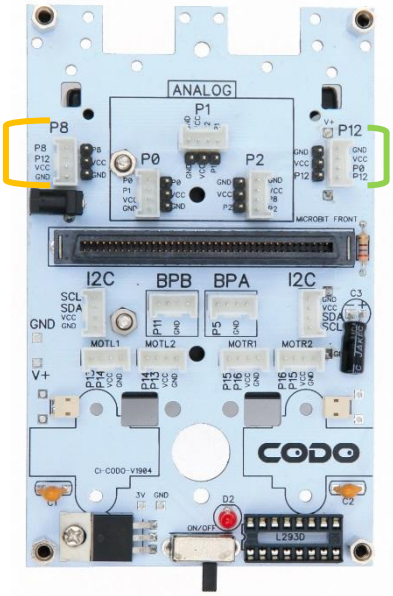
Attention toutefois, ce dossier n'explique pas l'utilisation de la carte micro:bit ni celle des modules Grove. Il est donc recommandé d'avoir des notions de bases sur ces éléments avant d'utiliser la carte CODO.



1. Connecteur pour carte micro:bit.
2. 13 connecteurs pour modules Grove.
3. 5 connecteurs pour servomoteurs.
4. 2 connecteurs pour option motorisation
5. Interrupteur M/A avec LED de mise sous tension.
6. Support de piles embarqué pour alimentation 4 piles ou accus AA.
7. Entrée d'alimentation annexe pour bloc secteur 5 à 9V.
8. Points de fixation multiples pour éléments mécaniques (motorisation, porte-stylo, etc.).

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.





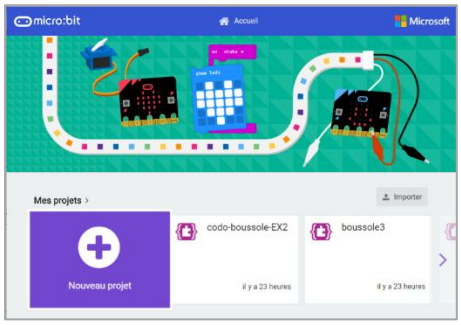
LES ENVIRONNEMENTS DE PROGRAMMATION

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 AppInventor2 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.

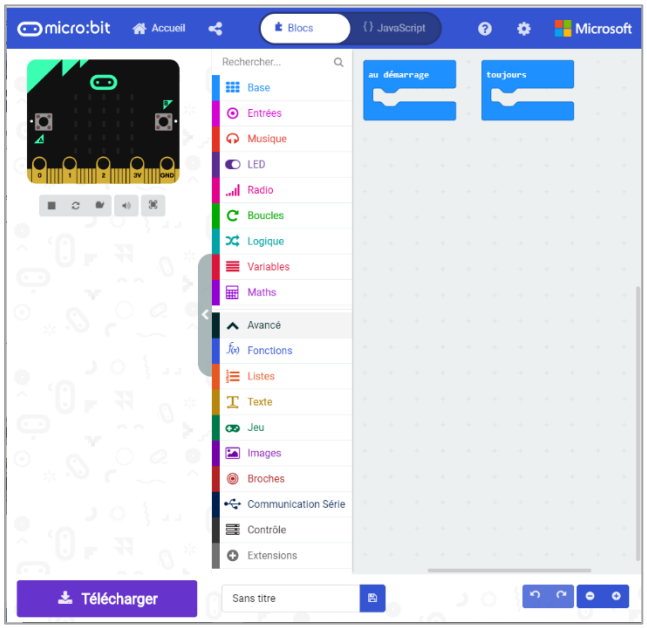
Pour accéder à ce logiciel disponible gratuitement sur internet, copier le lien suivant :
https://makecode.microbit.org



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.



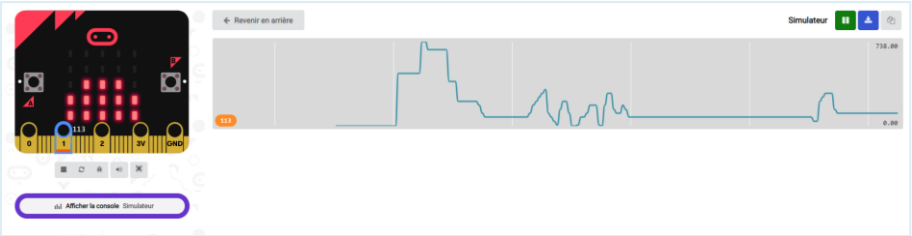
Vous pouvez coder en blocs et en JavaScript.
Ici, nous nous concentrons sur le mode blocs.
Sur la gauche de l'écran, vous trouvez les différents blocs disponibles.
En haut, les blocs de bases sont affichés.

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.

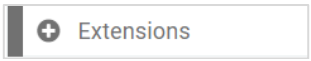
Mode simulation



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

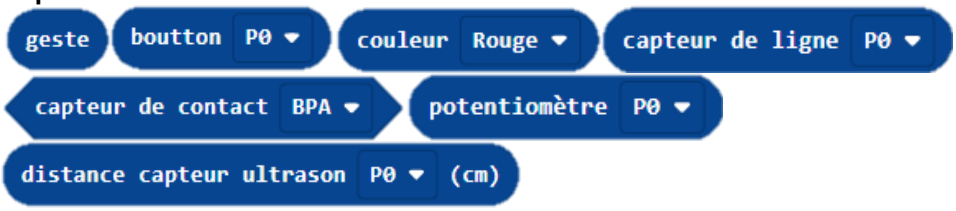
Vous pouvez ajouter des blocs supplémentaires.
A partir de la rubrique **Avancé**, cliquez sur **Extensions**.



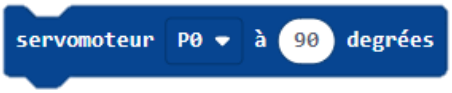
Tapez « CODO » dans le moteur de recherche.
Pour obtenir les blocs correspondant aux moteurs du robot, aux modules Grove abordés dans les cartes ExoProg.

Moteurs

Capteurs



Actionneurs



Afficheurs

Plus

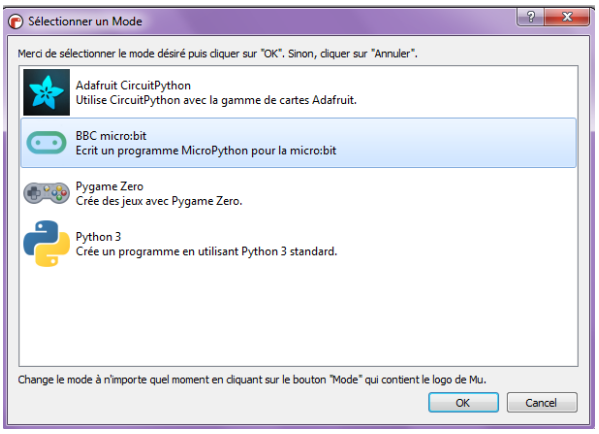
Autres extensions : Néopixel, Bluetooth, Grove …

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.

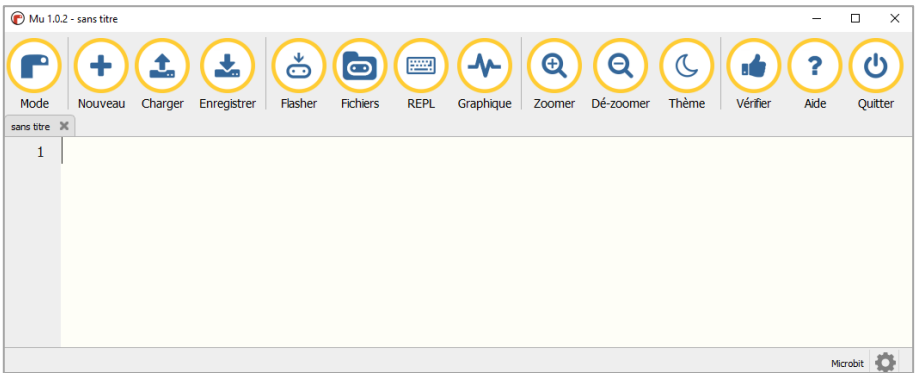
Mu Python Editor

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 ».



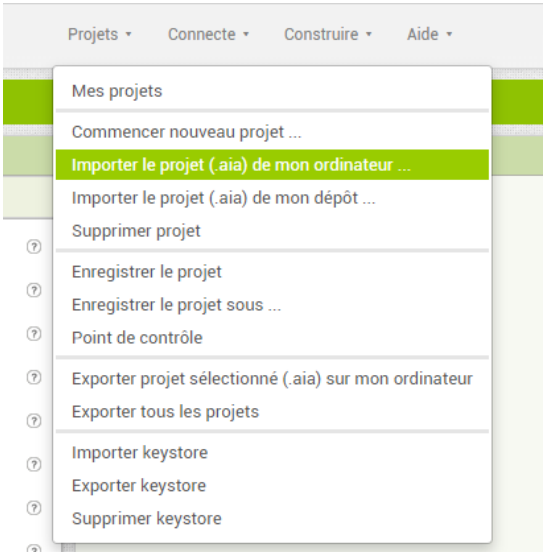
Guide les librairies :
Vous trouverez de nombreuses librairies sur le site suivant :
https://microbit-micropython.readthedocs.io/fr/latest/tutorials/introduction.html

App Inventor 2

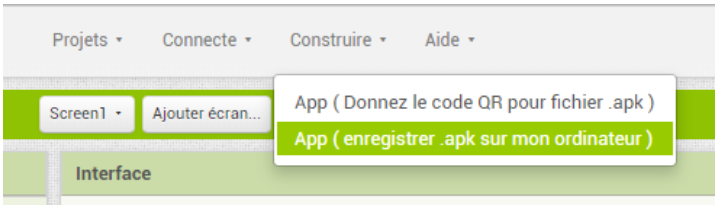
MIT App Inventor

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 **codo-bluetoothbase.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.

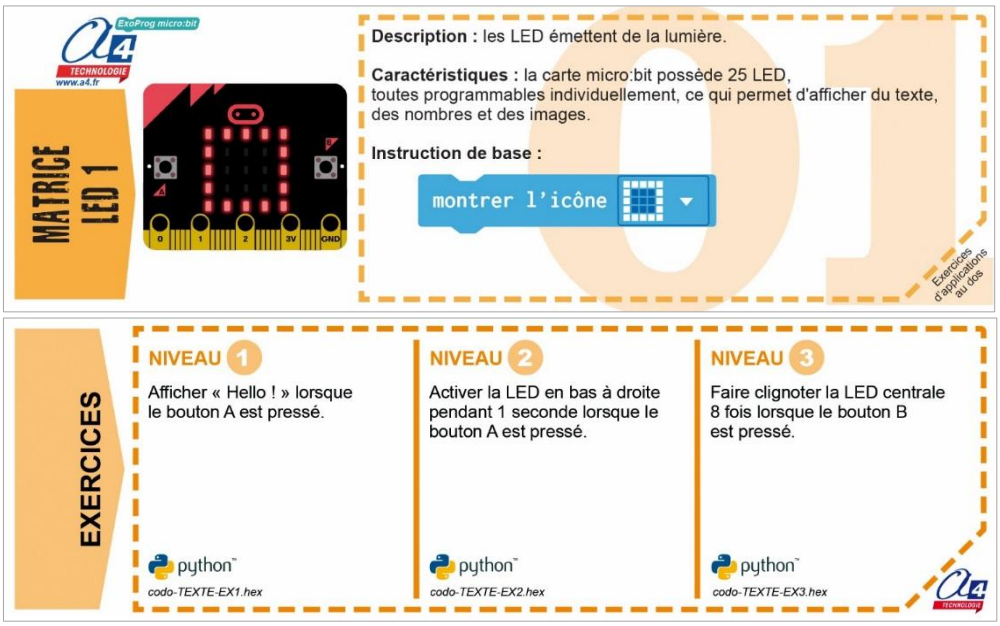
Note : pour pouvoir installer les applications d'A4, il faut autoriser l'installation d'applications de sources inconnues dans les paramètres de sécurité de votre smartphone.

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.



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 apparait à coté cela signifie qu'une correction en python sous mu est disponible.

Deux approches :

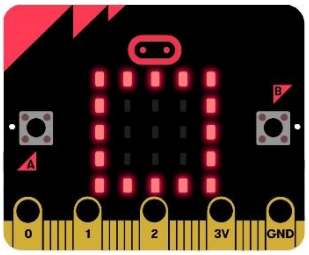
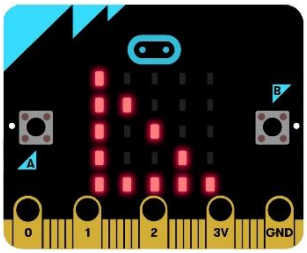
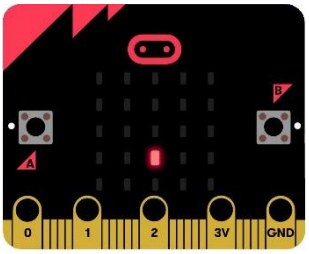
- Avec les corrections de programmes, les utilisateurs découvrent les principes de la programmation en blocs : chargement d'un programme, modification d'un programme et vérification sur le matériel (ex : modification des temps d'attente, etc.).
- Les utilisateurs conçoivent eux-mêmes le programme pour 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

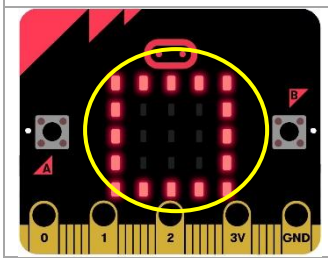
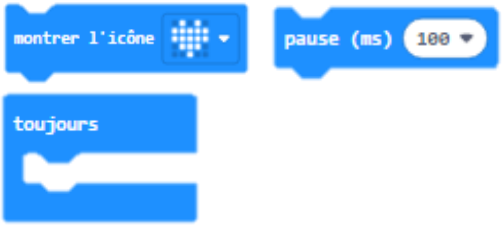
Module	Référence
Bouton-poussoir	S-101020003
Potentiomètre	S-101020017
Détection de ligne	S-101020009
Télémètre à ultrasons	S-101020010
LED rouge	S-104030005
Afficheur 4 digits	S-104030003
Ruban de LED	S-104020108
Joystick	S-101020028
Haut-parleur miniature	S-107020001
Capteur de mouvement	S-101020083

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

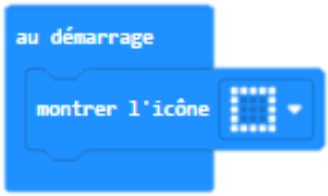
Programmer avec la carte micro:bit

Modules	Enoncés	Corrections
Matrice LED 1 	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
Bouton A et 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
Matrice LED 2 	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
Capteur de température 	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

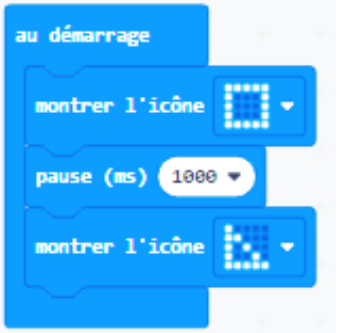
MATRICE LED 1

Module	Instructions / Fonctions utilisées	
	Base	

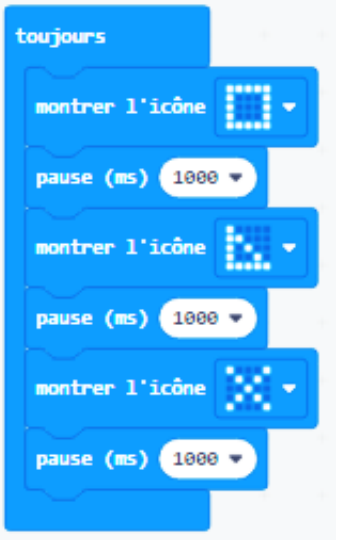
Niv1 : Afficher un carré

Avec MakeCode : codo-MOTIFS-EX1.hex	Avec mu Python Editor : codo-MOTIFS-EX1.py
	```1 from microbit import * 2 3 display.show(Image.SQUARE)```

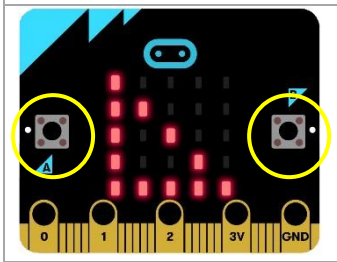
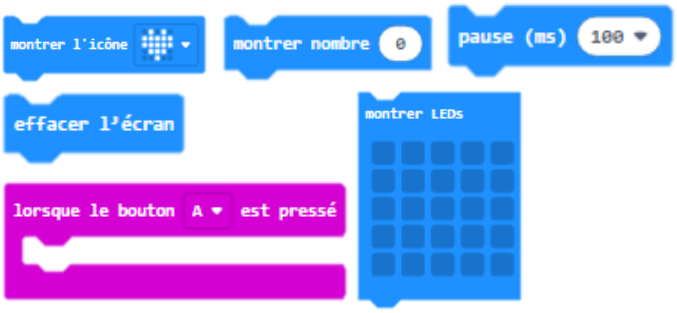
Niv2 : afficher un carré et une seconde après un triangle

Avec MakeCode : codo-MOTIFS-EX2.hex	Avec mu Python Editor : codo-MOTIFS-EX2.py
	```1 from microbit import * 2 3 display.show(Image.SQUARE) 4 sleep(1000) 5 display.show(Image.TRIANGLE)```

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
	```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)```

BOUTONS A ET B

Module	Instructions / Fonctions utilisées	
		

Niv1 : afficher un triangle quand le bouton A est pressé

Avec MakeCode : codo-BOUTONS-EX1.hex	Avec mu Python Editor : codo-BOUTONS-EX1.py
	

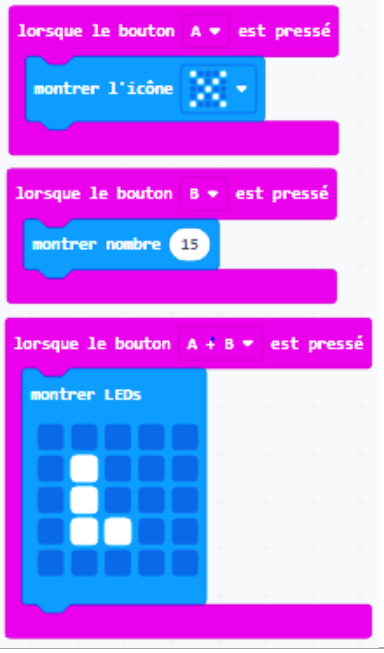

```python
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.TRIANGLE_LEFT)
```

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
	

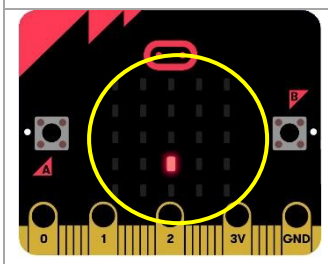
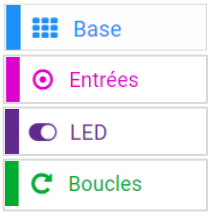
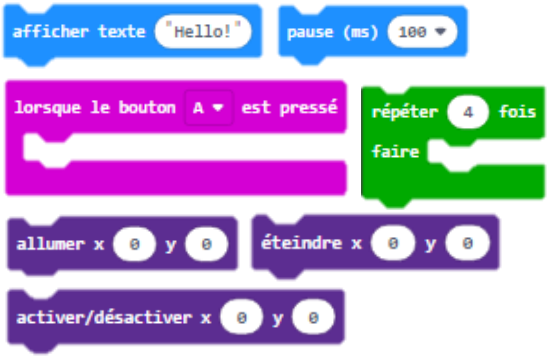

```python
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.TRIANGLE_LEFT)
        sleep(3000)
        display.clear()
```

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.

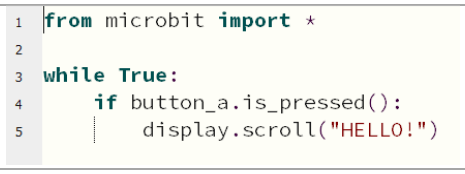
Avec MakeCode : codo-BOUTONS-EX3.hex	Avec mu Python Editor : codo-BOUTONS-EX3.py
	


```python
from microbit import *

lettre = Image("00000:"
               "00900:"
               "00900:"
               "00990:"
               "00000:")

while True:
    if button_a.is_pressed():
        display.show(Image.NO)
        if button_b.is_pressed():
            display.show(lettre)
    elif button_b.is_pressed():
        display.scroll(15)
    display.clear()
```

MATRICE LED 2

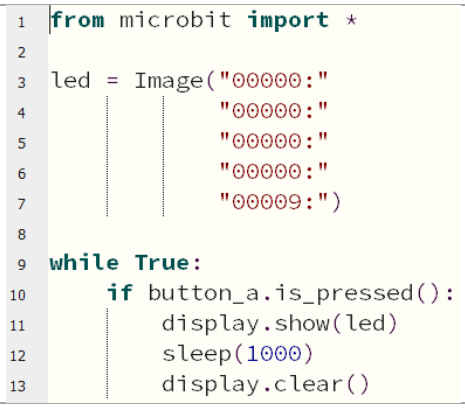
Module	Instructions / Fonctions utilisées	
		

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

Avec MakeCode : codo-TEXTE-EX1.hex	Avec mu Python Editor : codo-TEXTE-EX1.py
	

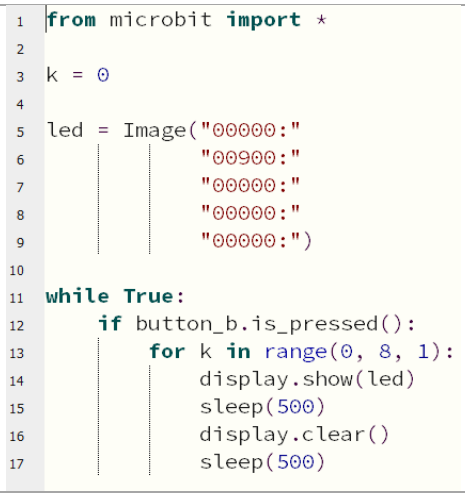

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.scroll("HELLO!")
```

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
	

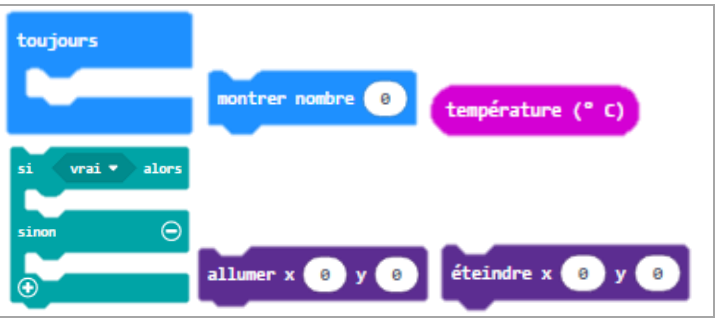

```
from microbit import *

led = Image("00000:"
            "00000:"
            "00000:"
            "00000:"
            "00009:")

while True:
    if button_a.is_pressed():
        display.show(led)
        sleep(1000)
        display.clear()
```

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
	


```
from microbit import *

k = 0

led = Image("00000:"
            "00900:"
            "00000:"
            "00000:"
            "00000:")

while True:
    if button_b.is_pressed():
        for k in range(0, 8, 1):
            display.show(led)
            sleep(500)
            display.clear()
            sleep(500)
```

CAPTEUR DE TEMPERATURE

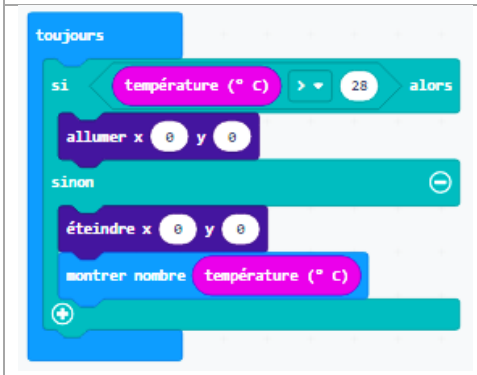
Module	Instructions / Fonctions utilisées	
	Base Entrées Logique LED	

Niv1 : afficher la valeur de la température

Avec MakeCode : codo-TEMP-EX1.hex	Avec mu Python Editor : codo-TEMP-EX1.py
	

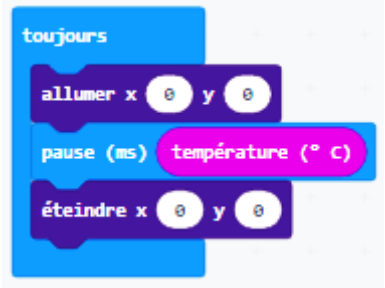

```
from microbit import *

t = temperature()

while True:
    display.scroll(t)
    sleep(1000)
```

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

Avec MakeCode : codo-TEMP-EX2.hex	Avec mu Python Editor : codo-TEMP-EX2.py
	

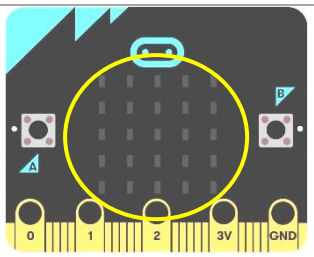
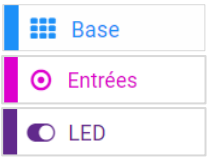
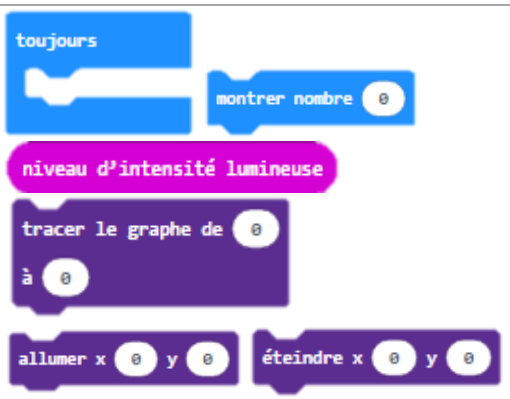

```
from microbit import *

t = temperature()

i = Image("90000:"
          "00000:"
          "00000:"
          "00000:"
          "00000:")

while True:
    if t > 28:
        display.show(i)
    else:
        display.scroll(t)
```

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

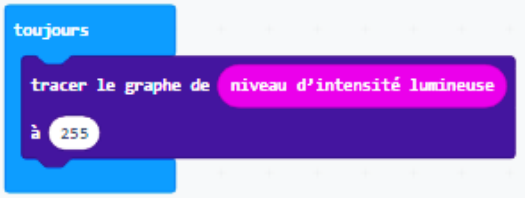
Avec MakeCode : codo-TEMP-EX3.hex	Avec mu Python Editor : codo-TEMP-EX3.py
	


```
from microbit import *

i = Image("90000:"
          "00000:"
          "00000:"
          "00000:"
          "00000:")

while True:
    if temperature() > 0:
        display.show(i)
        sleep(temperature())
        display.clear()
        sleep(temperature())
```

CAPTEUR DE LUMIERE

Module	Instructions / Fonctions utilisées	
		

Niv1 : tracer le graphe du niveau d'intensité lumineuse

Avec MakeCode : codo-LUM-EX1.hex	Avec mu Python Editor : codo-LUM-EX1.py
	

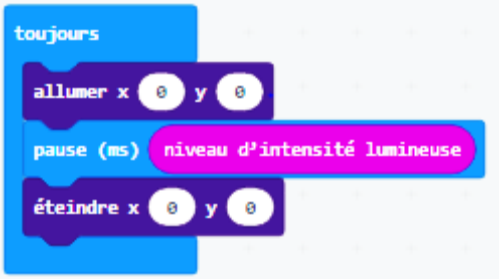

```
from microbit import *

lumière = display.read_light_level()
while True:
    display.scroll(lumière)
    sleep(1000)
```

Niv2 : afficher la valeur du niveau d'intensité lumineuse

Avec MakeCode : codo-LUM-EX2.hex	Avec mu Python Editor : codo-LUM-EX2.py
	

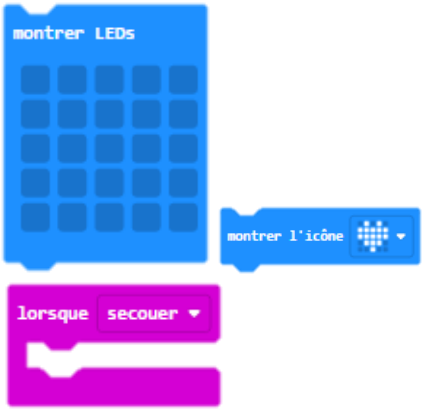

```
from microbit import *

lumière = display.read_light_level()
intensité = (lumière/255)*100
while True:
    display.scroll(intensité)
    sleep(1000)
```

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

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


```
from microbit import *

LED = Image ("00000:"
             "00000:"
             "00900:"
             "00000:"
             "00000:")

lumière = display.read_light_level()

while True:
    display.show(LED)
    sleep(lumière)
    display.clear()
    sleep(lumière)
```

ACCELEROMETRE

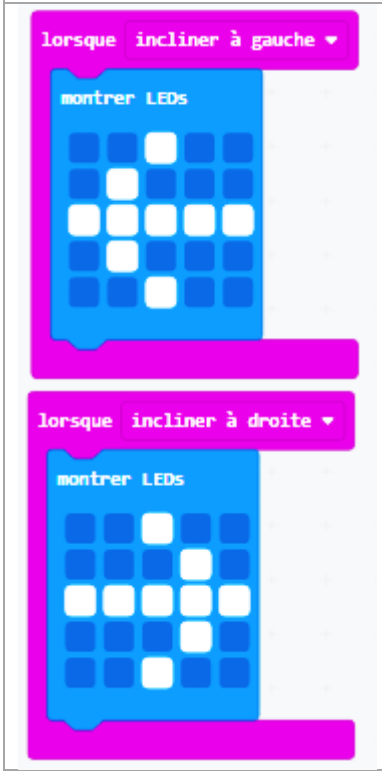
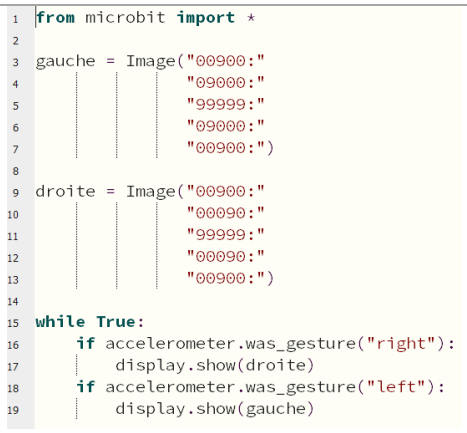
Module	Instructions / Fonctions utilisées	
	Base Entrées	

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

Avec MakeCode : codo-accel-EX1.hex	Avec mu Python Editor : codo-accel-EX1.py
	

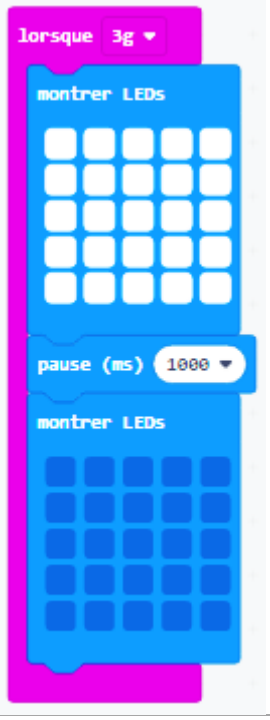
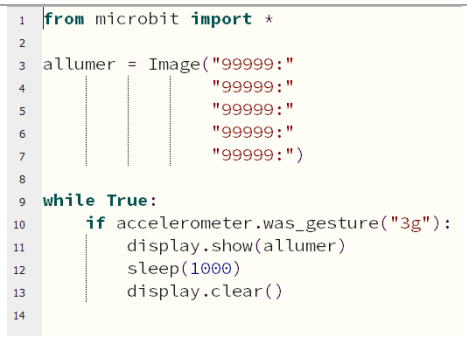

```
from microbit import *

while True:
    if accelerometer.was_gesture("shake"):
        display.show(Image.ANGRY)
```

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

Avec MakeCode : codo-accel-EX2.hex	Avec mu Python Editor : codo-accel-EX2.py
	

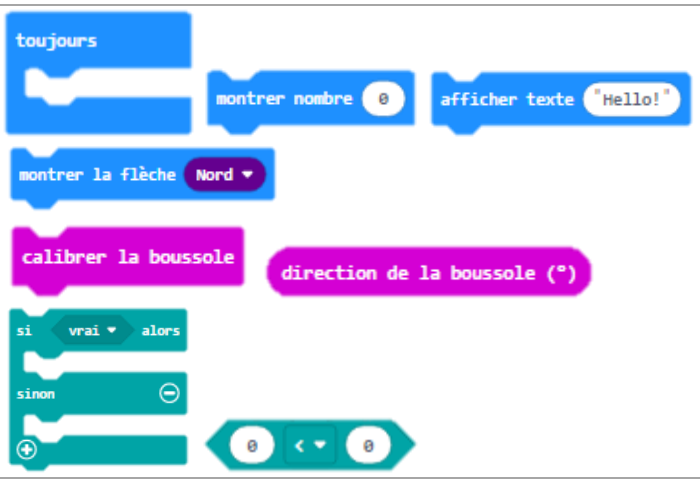

```
from microbit import *

gauche = Image("00900:"
               "09000:"
               "99999:"
               "09000:"
               "00900:")

droite = Image("00900:"
               "00090:"
               "99999:"
               "00090:"
               "00900:")

while True:
    if accelerometer.was_gesture("right"):
        display.show(droite)
    if accelerometer.was_gesture("left"):
        display.show(gauche)
```

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
	

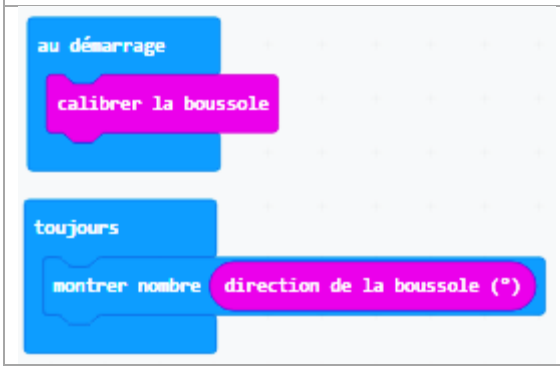

```python
from microbit import *

allumer = Image("99999:"
                "99999:"
                "99999:"
                "99999:"
                "99999:")

while True:
    if accelerometer.was_gesture("3g"):
        display.show(allumer)
        sleep(1000)
        display.clear()

```

BOUSSOLE

Module	Instructions / Fonctions utilisées	
		

Niv1 : afficher la direction de la boussole, la faire pivoter et observer les variations.

Avec MakeCode : codo-boussole-EX1.hex

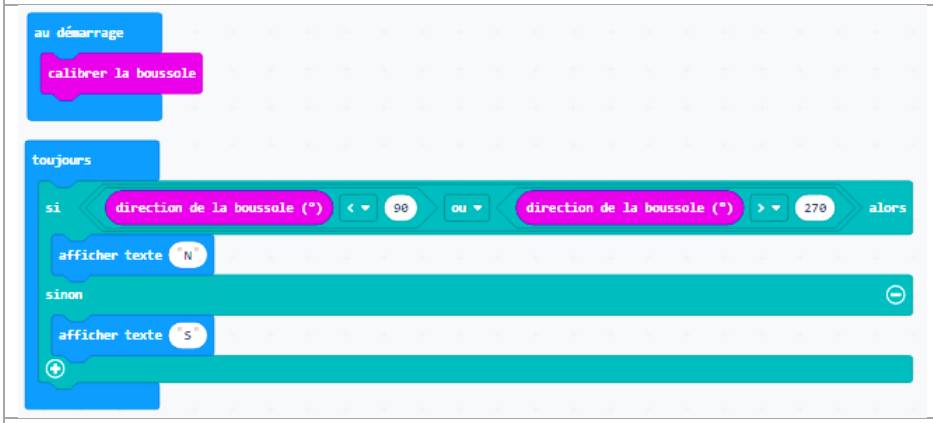


Avec mu Python Editor : codo-boussole-EX1.py

```
from microbit import *

compass.calibrate()

while True:
    display.scroll(compass.heading())
    sleep(1000)
    display.clear()
```

Niv2 : si la boussole est pointée vers le nord, afficher N ; si elle est pointée vers le sud, afficher S

Avec MakeCode : codo-boussole-EX2.hex

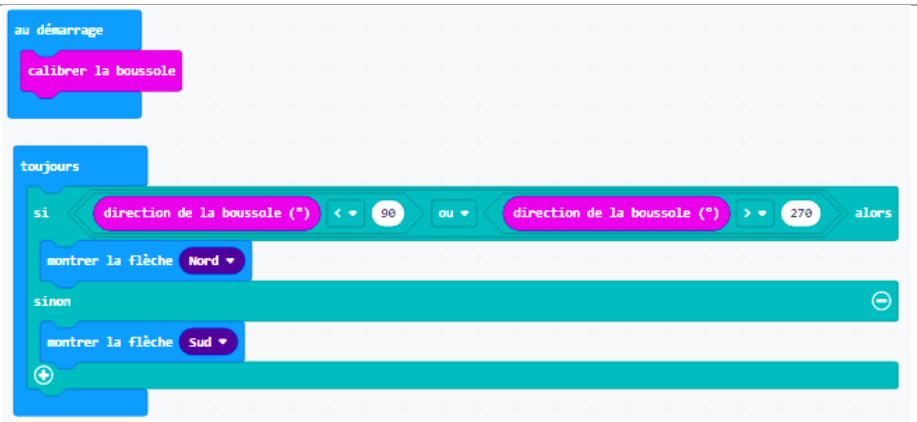


Avec mu Python Editor : codo-boussole-EX2.py

```
from microbit import *

compass.calibrate()

while True:
    if compass.heading() < 90 or compass.heading() > 270:
        display.show("N")
        sleep(1000)
        display.clear()
    else:
        display.show("S")
        sleep(1000)
        display.clear()
```

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

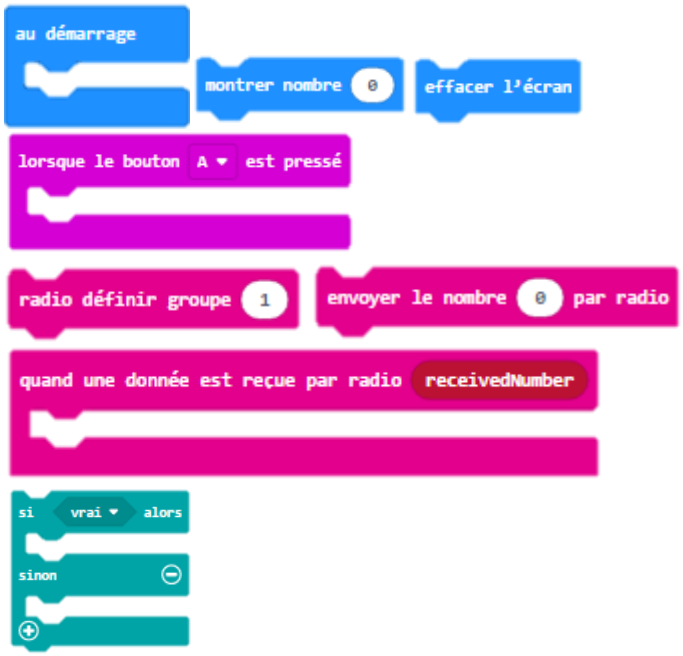


Avec mu Python Editor : codo-boussole-EX3.py

```
from microbit import *

compass.calibrate()

N = Image("00900:"
          "09990:"
          "90909:"
          "00900:"
          "00900:")

S = Image("00900:"
          "00900:"
          "90909:"
          "09990:"
          "00900:")

while True:
    if compass.heading() < 90 or compass.heading() > 270:
        display.show(N)
        sleep(1000)
        display.clear()
    else:
        display.show(S)
        sleep(1000)
        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.

Radio	**Instructions / Fonctions utilisées**	
	Base Entrées Radio Variables Logique	

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.

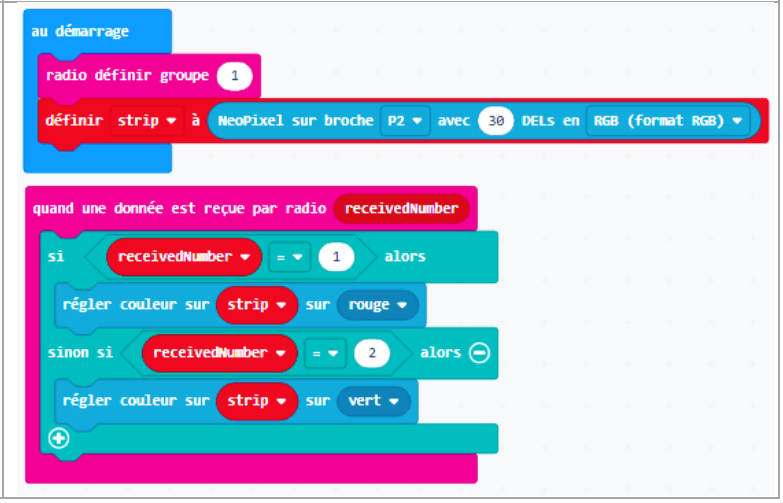
Avec MakeCode : codo-RADIO-EX1A.hex	Avec MakeCode : codo-RADIO-EX1B.hex
	

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 disparait 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
	

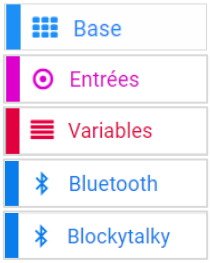
COMMUNICATION BLUETOOTH

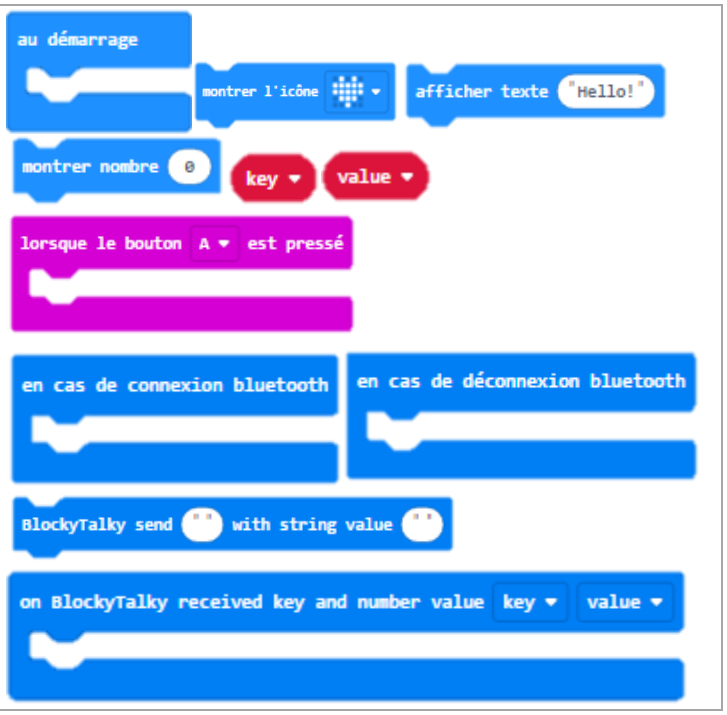
Permet de transmettre des données au travers d'une liaison sans fil entre un smartphone Android et la carte micro:bit. Les applications proposées sont réalisées sous App Inventor 2.
Le fichier « Documentdebase.aia » sert de base pour coder une application sur AppInventor. Il contient toute la partie 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

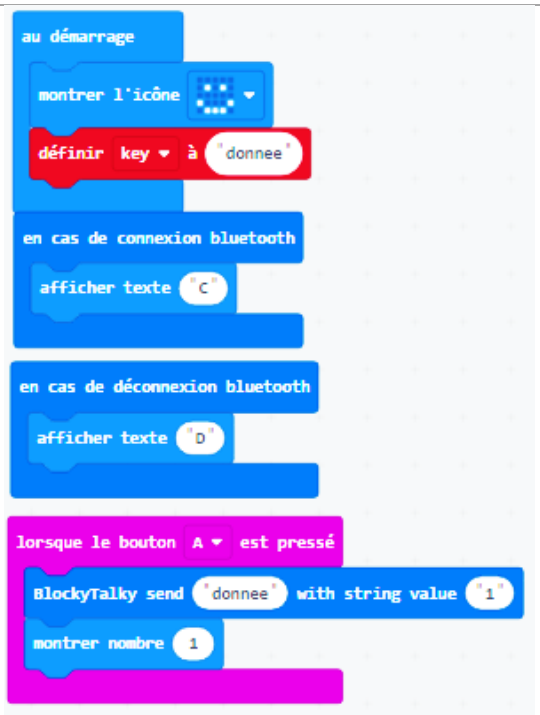
Instructions / Fonctions utilisées





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

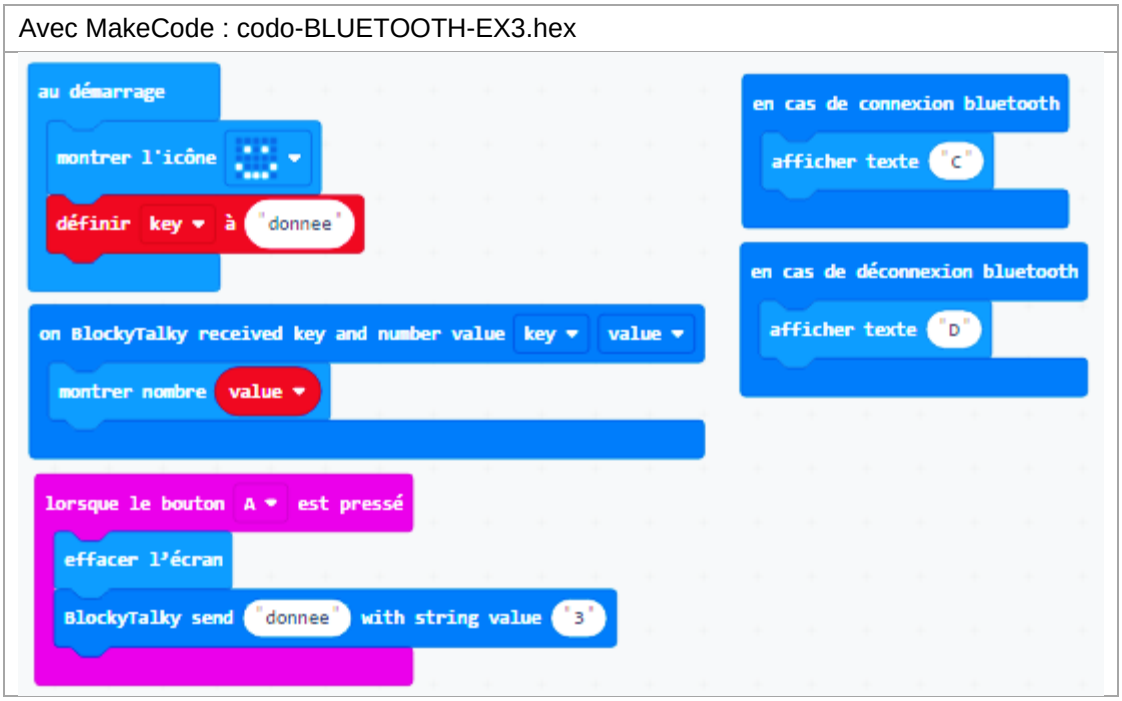


Niveau 2 : Communication de l'application vers la carte micro:bit - Lorsque vous appuyez sur un bouton dans l'application, un chiffre est envoyé et affiché sur la carte.

Avec MakeCode : codo-BLUETOOTH-EX2.hex

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

Avec MakeCode : codo-BLUETOOTH-EX3.hex



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.

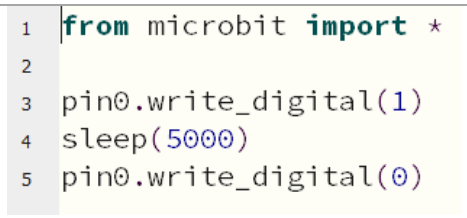
Modules	Enoncés	Corrections
LED 	Niv1 : activer la LED pendant 5 secondes puis l'éteindre.	codo-led-EX1
	Niv2 : faire clignoter la LED toutes les 0,5 secondes.	codo-led-EX2
	Niv3 : faire varier la vitesse de clignotement de la LED.	codo-led-EX3
Bouton-poussoir 	Niv1 : lorsque l'on appuie sur le bouton 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
Afficheur 4 digits 	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
Télémètre à ultrasons 	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
Speaker 	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

Potentiomètre 	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
Capteur de mouvement 	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
Neopixel 	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
Neopixel (suite) 	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 apparaitre un arc en ciel des LED et faire varier les couleurs à l'aide du potentiomètre.	codo-Neo2-EX3

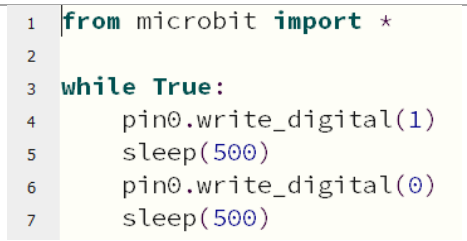
LED

Module	Instructions / Fonctions utilisées	
	Base Broches Variables	

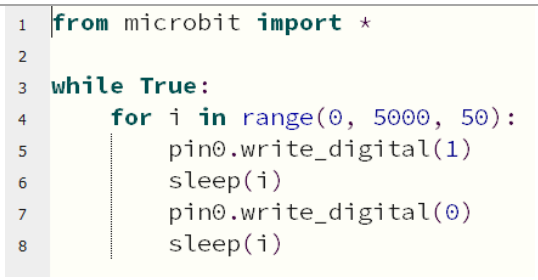
Niv1 : activer la LED pendant 5 secondes, puis l'éteindre

Avec MakeCode : codo-led-EX1.hex	Avec mu Python Editor : codo-led-EX1.py
	

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

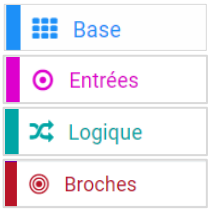
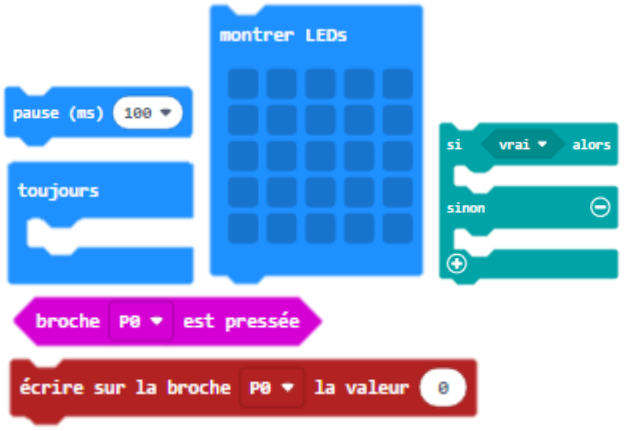
Avec MakeCode : codo-led-EX2.hex	Avec mu Python Editor : codo-led-EX2.py
	

Niv3 : faire varier la vitesse de clignotement de la LED

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


```
from microbit import *

while True:
    for i in range(0, 5000, 50):
        pin0.write_digital(1)
        sleep(i)
        pin0.write_digital(0)
        sleep(i)
```

BOUTON-POUSSOIR

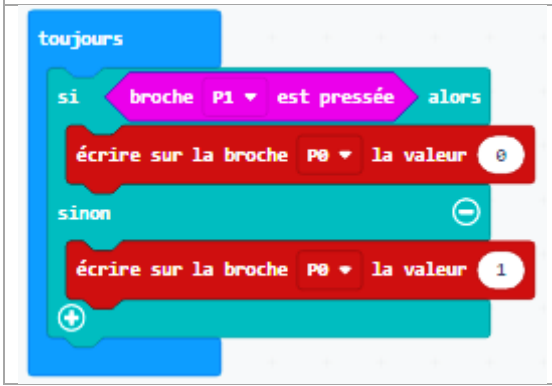
Module	Instructions / Fonctions utilisées	
		

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
	


```
from microbit import *

while True:
    if pin12.read_digital()==1:
        display.show(Image.PITCHFORK)
    sleep(1000)
    display.clear()
```

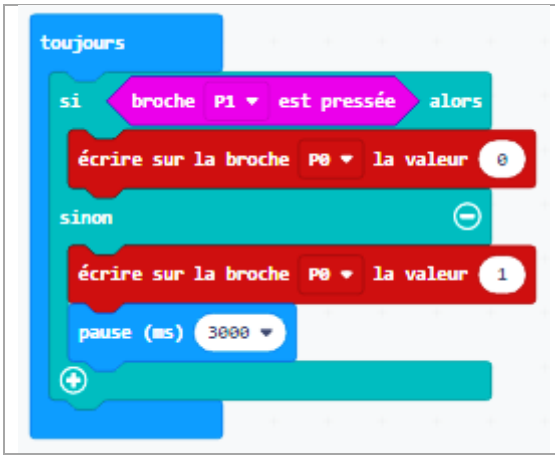
Niv2 : lorsque l'on appuie sur le bouton-poussoir, allumer la LED et l'éteindre lorsqu'il est relâché

Avec MakeCode : codo-BP-EX2.hex	Avec mu Python Editor : codo-BP-EX2.py
	


```
from microbit import *

while True:
    if pin12.read_digital()==1:
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)
```

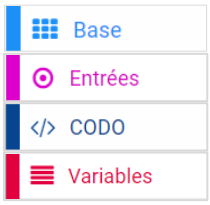
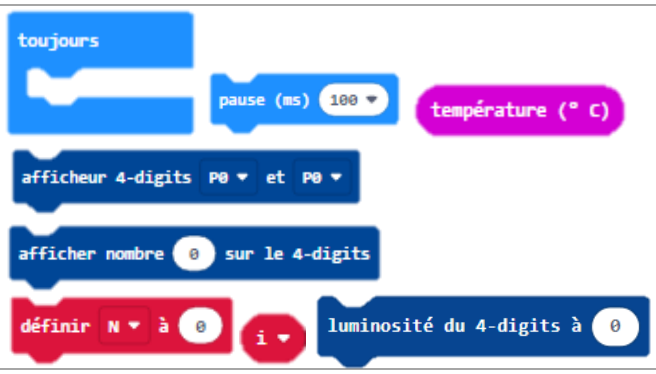
Niv3 : éteindre la LED 3 secondes après avoir relâché le bouton

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




```
from microbit import *

while True:
    if pin12.read_digital()==1:
        pin0.write_digital(1)
    else:
        sleep(3000)
        pin0.write_digital(0)
```

AFFICHEUR 4 DIGITS

Module	Instructions / Fonctions utilisées	
		

Niv1 : afficher "1234"

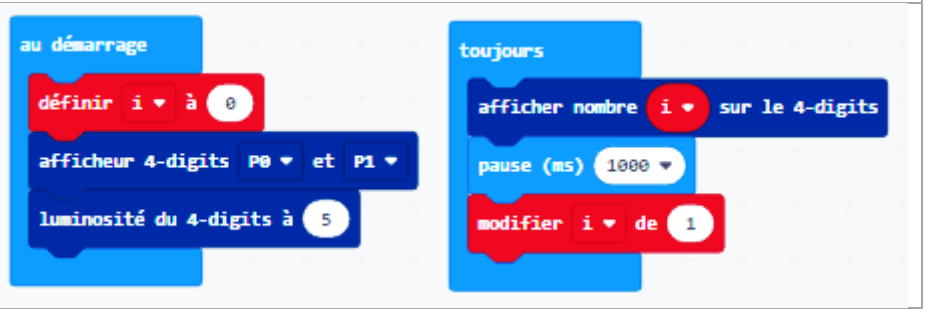
Avec MakeCode : codo-digits-EX1.hex

Niv2 : afficher la valeur du capteur de température

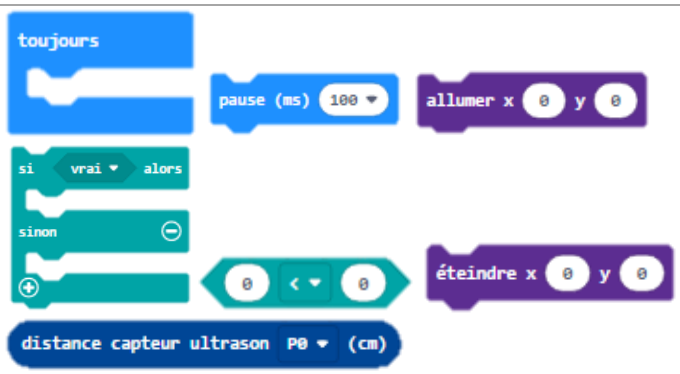
Avec MakeCode : codo-digits-EX2.hex

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

Avec MakeCode : codo-digits-EX3.hex

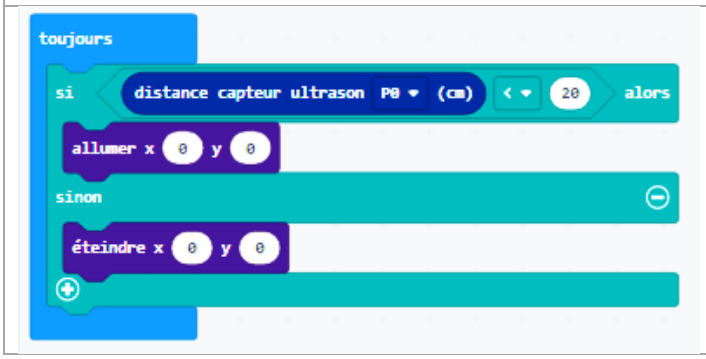


TELEMETRE A ULTRASONS

Module	Instructions / Fonctions utilisées	
	Base LED Logique CODO	

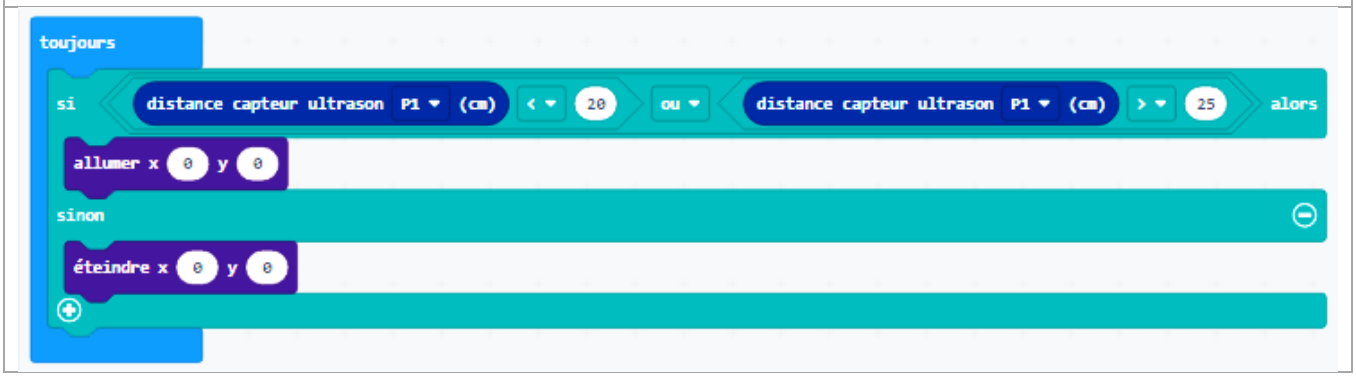
Niv1 : activer le module LED si la distance mesurée est inférieure à 20 cm sinon la désactiver

Avec MakeCode : codo-ULTRA-EX1.hex



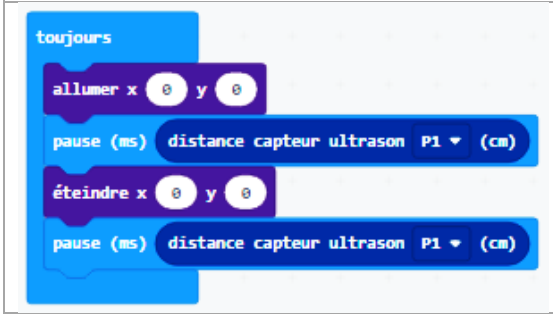
Niv2 : activer le module LED si la distance mesurée est inférieure à 20 cm et la réactiver quand l'objet détecté s'est éloigné de plus de 25 cm

Avec MakeCode : codo-ULTRA-EX2.hex

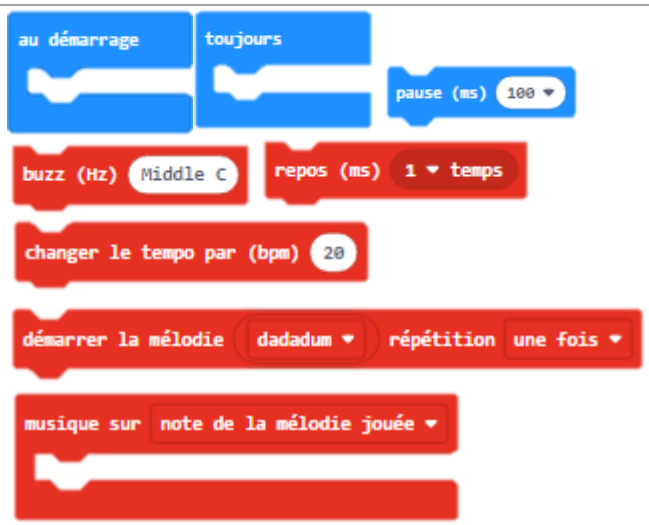


Niv3 : faire clignoter le module LED à une fréquence variant avec la distance mesurée.

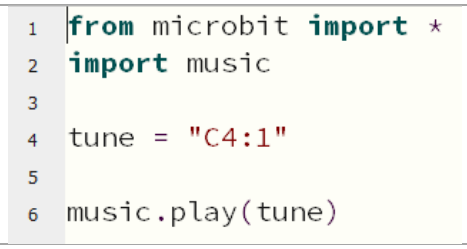
Avec MakeCode : codo-ULTRA-EX3.hex



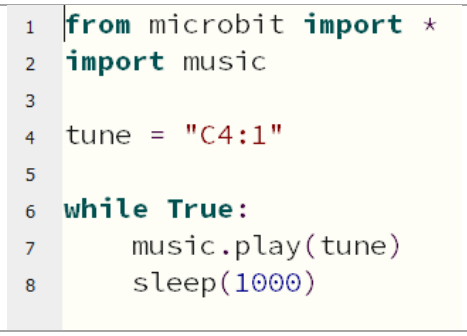
SPEAKER

Module	Instructions / Fonctions utilisées	
	Base Musique	

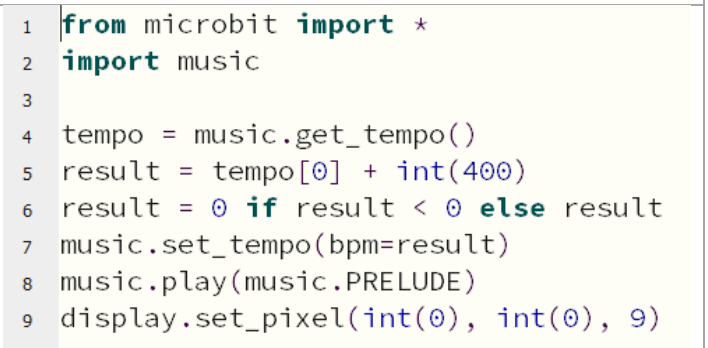
Niv1 : actionner le buzzer

Avec MakeCode : codo-speaker-EX1.hex	Avec mu Python Editor : codo-speaker-EX1.py
	

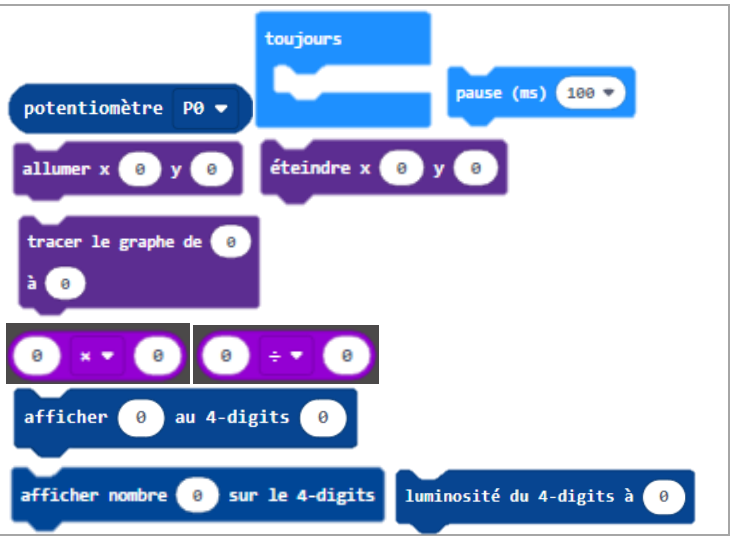
Niv2 : actionner le buzzer toutes les secondes

Avec MakeCode : codo-speaker-EX2.hex	Avec mu Python Editor : codo-speaker-EX2.py
	

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

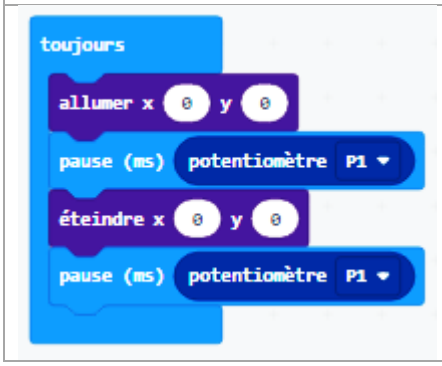
Avec MakeCode : codo-speaker-EX3.hex	Avec mu Python Editor : codo-speaker-EX3.py
	

POTENTIOMETRE

Module	Instructions / Fonctions utilisées	
	CODO Base LED Maths	

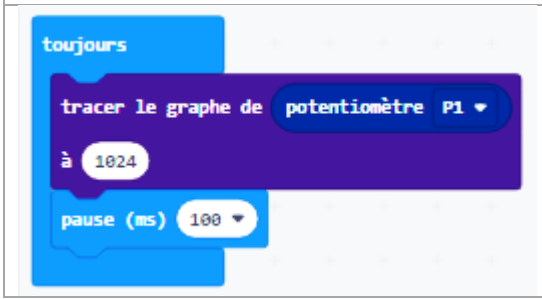
Niv1 : faire clignoter une LED à une fréquence dépendant de la valeur du potentiomètre

Avec MakeCode : codo-POT-EX1_v2.hex



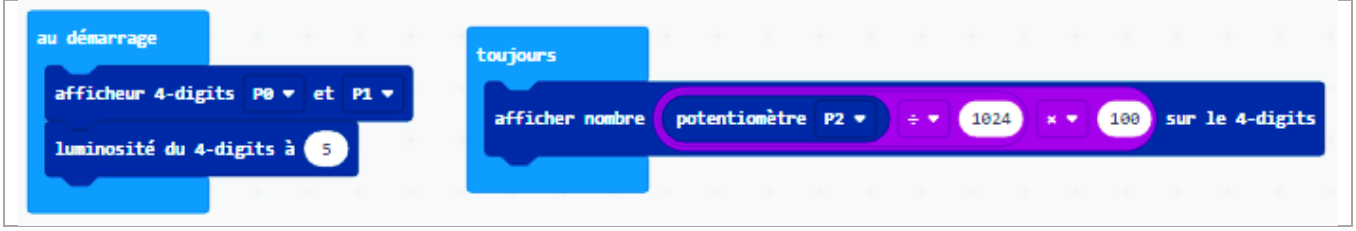
Niv2 : tracer le graphe de la valeur du potentiomètre

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

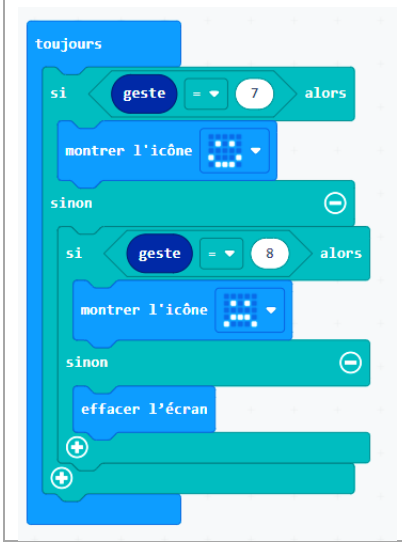
Avec MakeCode : codo-POT-EX3.hex



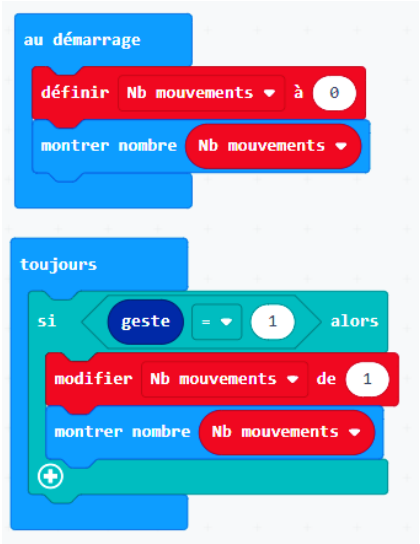
CAPTEUR DE MOUVEMENT

Module	Instructions / Fonctions utilisées	
		 **Valeurs renvoyées par le capteur :** Aucun : 0 Droite : 1 Gauche : 2 Vers le haut : 3 Vers le bas : 4 Tout droit : 5 En arrière : 6 Sens horaire : 7 Sens antihoraire : 8 Vague : 9

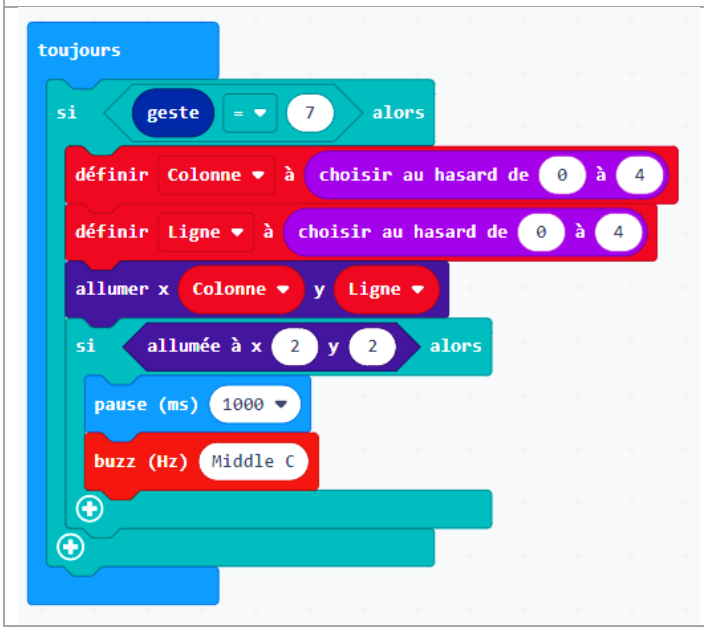
Niv1 : lorsque l'on tourne dans le sens des aiguilles d'une montre afficher ☺ et dans le sens antihoraire afficher ☹

Avec MakeCode : codo-CM-EX1.hex	
	

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

Avec MakeCode : codo-CM-EX2.hex	
	

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	
	

NEOPIXEL

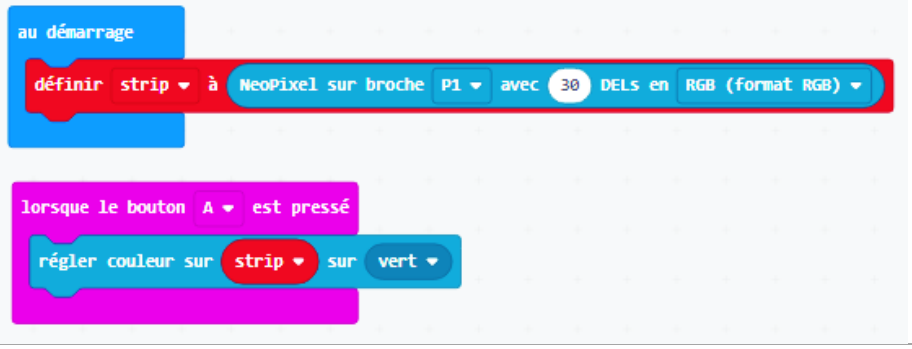
Module	Instructions / Fonctions utilisées
	Base Variables Neopixel Entrées

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

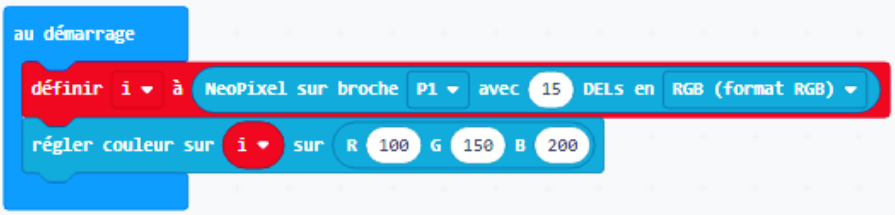


Avec mu Python Editor : codo-Neo-EX2.py

```
from microbit import *
import neopixel
np = neopixel.NeoPixel(pin0, 30)
i = 0

while True:
    if button_a.is_pressed():
        for k in range(0, 29, 1):
            np[i] = (0, 255, 0)
            np.show()
            i = i + 1

```

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

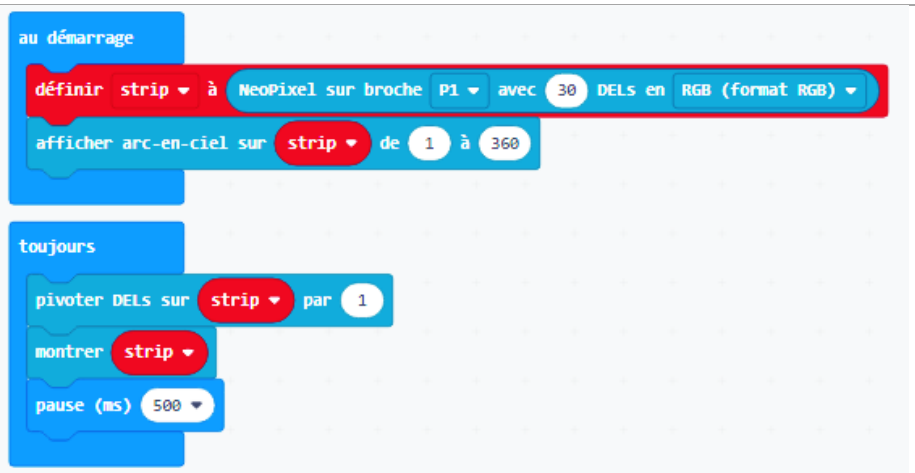
```
from microbit import *
import neopixel
np = neopixel.NeoPixel(pin0, 15)
i = 0

for k in range(0, 15, 1):
    np[i] = (100, 150, 200)
    np.show()
    i = i + 1
```

NEOPIXEL (suite)

Module	Instructions / Fonctions utilisées	
	Base Variables Neopixel	au démarrage définir strip à NeoPixel sur broche P0 avec 24 DELs en RGB (format RGB) afficher arc-en-ciel sur strip de 1 à 360 régler couleur sur strip sur rouge

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

Avec MakeCode : codo-Neo2-EX1.hex

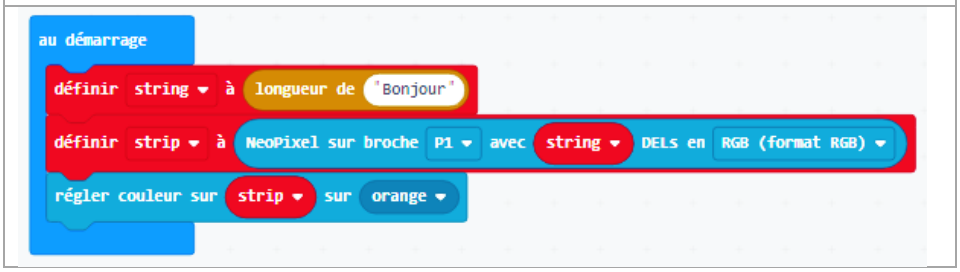


Avec mu Python Editor : codo-Neo2-EX1.py

```
from microbit import *
import neopixel

i = 0
np = neopixel.NeoPixel(pin0, 30)

while True:
    for k in range(0, 29, 1):
        np[i] = (255, 0, 0)
        np.show()
        sleep(500)
        i = i + 1
    for n in range(29, 0, -1):
        np[i] = (255, 130, 70)
        np.show()
        sleep(500)
        i = i - 1
    np.clear()
```

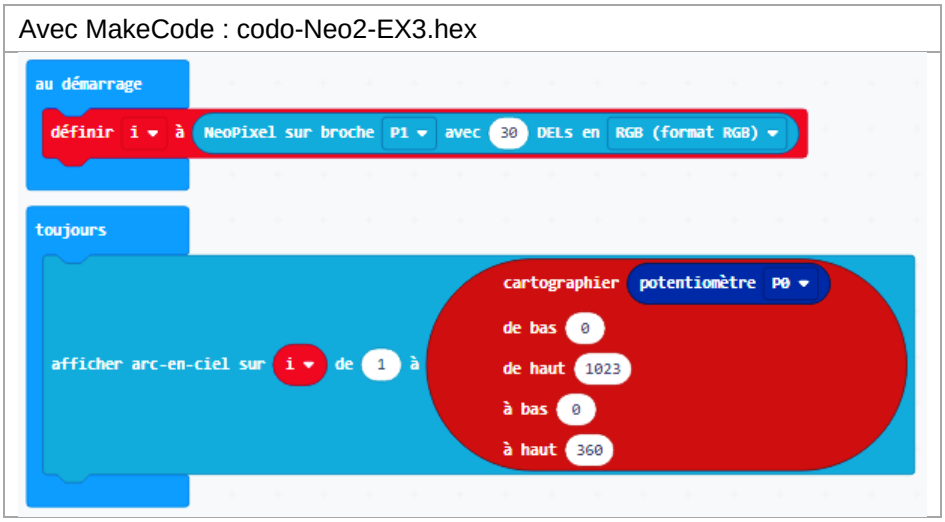
Niv2 : allumer en orange autant de LED qu'il y a de lettres dans le mot « bonjour »

Avec MakeCode : codo-Neo2-EX2.hex



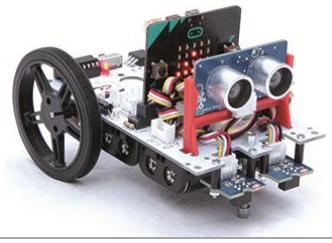
Niv3 : faire apparaitre un arc en ciel des LED et faire varier les couleurs à l'aide du potentiomètre

Avec MakeCode : codo-Neo2-EX3.hex



Programmer avec les options CODO

MOTORISATION

Option motorisation	Instructions / Fonctions utilisées	
	Base CODO	au démarrage pause (ms) 100 déplacement avancer vitesse 75 stopper les deux moteurs

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

au démarrage
déplacement avancer vitesse 50
pause (ms) 1000
stopper les deux moteurs
	`from microbit import *`
	
	`pin14.write_analog(512)`
	`pin13.write_analog(0)`
	`pin16.write_analog(512)`
	`pin15.write_analog(0)`
	`sleep(1000)`
	`pin14.write_analog(0)`
	`pin13.write_analog(0)`
	`pin16.write_analog(0)`
	`pin15.write_analog(0)`

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

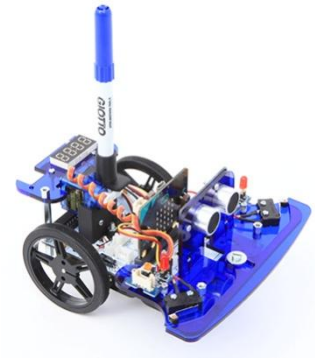
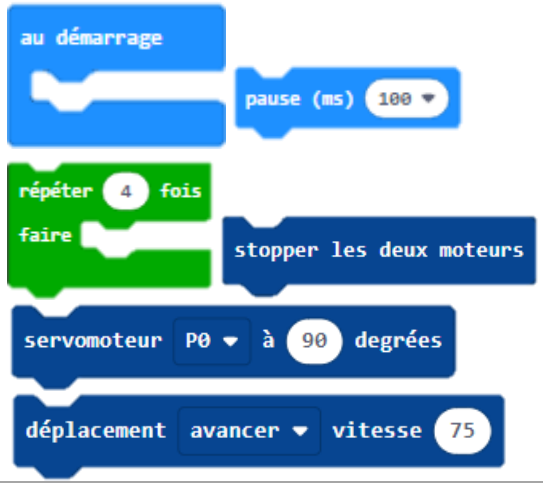
au démarrage
déplacement reculer vitesse 50
pause (ms) 1000
stopper les deux moteurs
	`from microbit import *`
	
	`pin13.write_analog(512)`
	`pin14.write_analog(0)`
	`pin15.write_analog(512)`
	`pin16.write_analog(0)`
	`sleep(1000)`
	`pin13.write_analog(0)`
	`pin14.write_analog(0)`
	`pin15.write_analog(0)`
	`pin16.write_analog(0)`

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

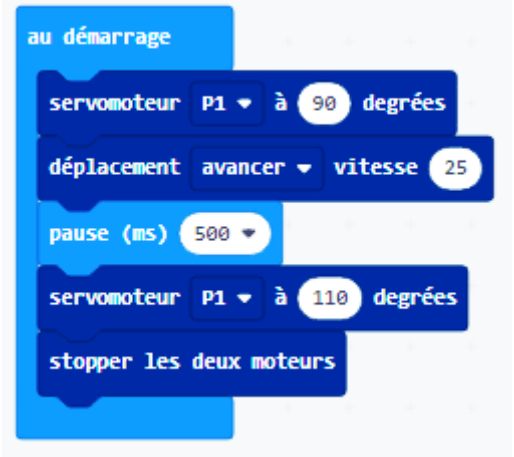
Avec MakeCode : codo-ROBOT-EX3	Avec mu Python Editor : codo-ROBOT-EX3.py
	


```
from microbit import *

pin14.write_analog(512)
pin13.write_analog(0)
pin16.write_analog(512)
pin15.write_analog(0)
sleep(500)
pin14.write_analog(512)
pin13.write_analog(0)
pin16.write_analog(0)
pin15.write_analog(512)
sleep(300)
pin14.write_analog(0)
pin13.write_analog(0)
pin16.write_analog(0)
pin15.write_analog(0)
```

PORTE-STYLO

Option porte-stylo	Instructions / Fonctions utilisées	
	Base CODO	

Niveau 1 : dessiner un segment

Avec MakeCode : codo-STYLO-EX1.hex	Avec mu Python Editor : codo-STYLO-EX1.py
	

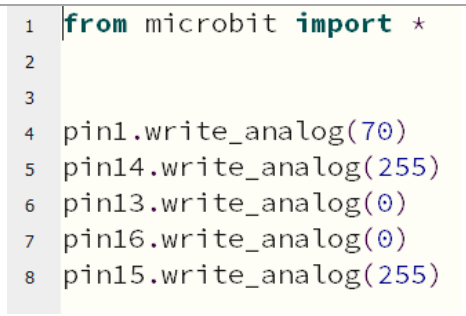

```
from microbit import *


pin1.write_analog(70)
pin14.write_analog(512)
pin13.write_analog(0)
pin16.write_analog(512)
pin15.write_analog(0)
sleep(500)
pin1.write_analog(90)
pin14.write_analog(0)
pin13.write_analog(0)
pin16.write_analog(0)
pin15.write_analog(0)
```

Niveau 2 : dessiner des traits en pointillés

Avec MakeCode : codo-STYLO-EX2.hex	Avec mu Python Editor : codo-STYLO-EX2.py
	

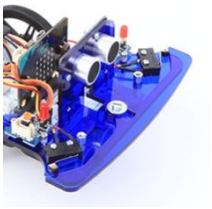

```
from microbit import *

while True:
    pin1.write_analog(70)
    pin14.write_analog(125)
    pin13.write_analog(0)
    pin16.write_analog(125)
    pin15.write_analog(0)
    sleep(200)
    pin1.write_analog(90)
    pin14.write_analog(125)
    pin13.write_analog(0)
    pin16.write_analog(125)
    pin15.write_analog(0)
    sleep(100)
```

Niveau 3 : dessiner un rond

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

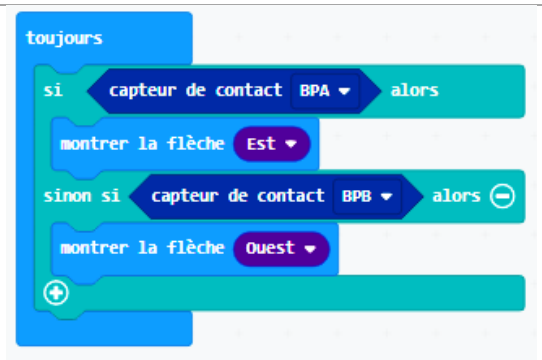

```
from microbit import *


pin1.write_analog(70)
pin14.write_analog(255)
pin13.write_analog(0)
pin16.write_analog(0)
pin15.write_analog(255)
```

PARE-CHOCS

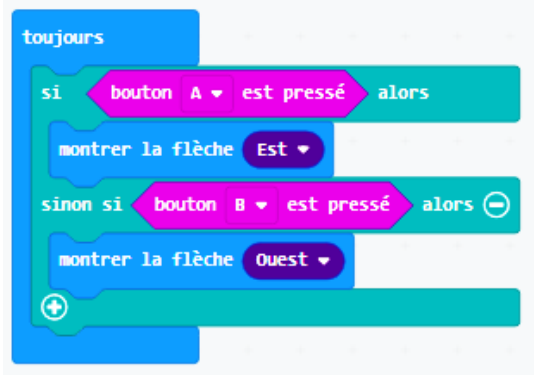
Option pare-chocs	Instructions / Fonctions utilisées	
	Base CODO Logique	capteur de contact BPA montrer la flèche Nord si vrai alors sinon

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



Ou

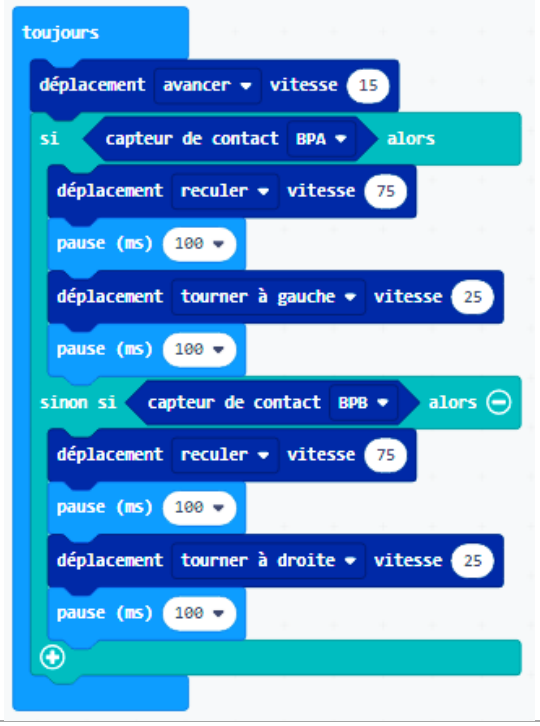


Avec mu Python Editor : codo-PARECHOC-EX1.py

```
from microbit import *

while True:
    pin14.write_analog(200)
    pin13.write_analog(0)
    pin16.write_analog(200)
    pin15.write_analog(0)
    if button_a.is_pressed():
        pin13.write_analog(200)
        pin14.write_analog(0)
        pin15.write_analog(200)
        pin16.write_analog(0)
        sleep(200)
        pin14.write_analog(0)
        pin13.write_analog(300)
        pin16.write_analog(300)
        pin15.write_analog(0)
        sleep(200)
    elif button_b.is_pressed():
        pin13.write_analog(200)
        pin14.write_analog(0)
        pin15.write_analog(200)
        pin16.write_analog(0)
        sleep(200)
        pin13.write_analog(0)
        pin14.write_analog(300)
        pin16.write_analog(0)
        pin15.write_analog(300)
        sleep(200)
```

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



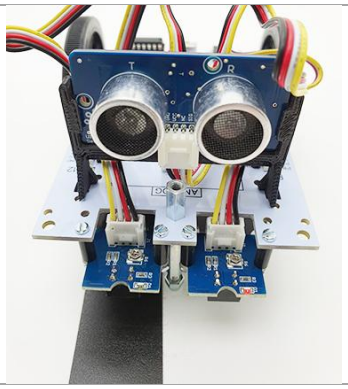

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image("00900:"
                           "00090:"
                           "99999:"
                           "00090:"
                           "00900:"))
    elif button_b.is_pressed():
        display.show(Image("00900:"
                           "09000:"
                           "99999:"
                           "09000:"
                           "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.

Détection de ligne	**Instructions / Fonctions utilisées**	
	Base CODO Logique	

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

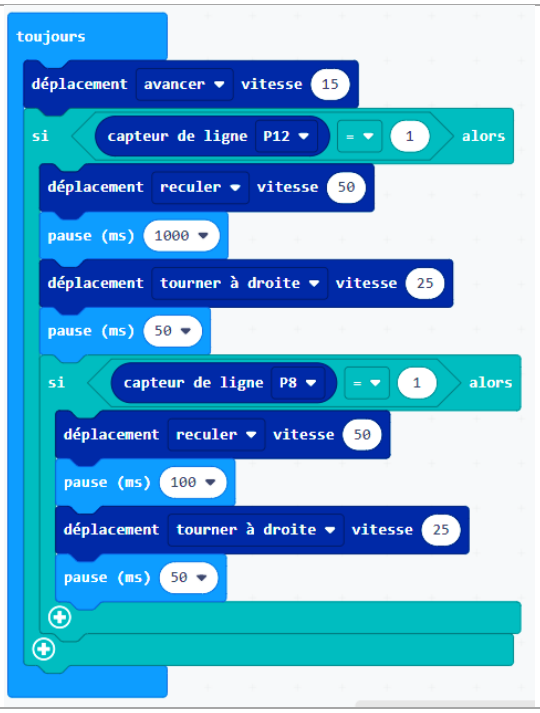
Avec MakeCode : codo-LIGNE-EX1.hex

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

```
1  from microbit import *
2
3  while True:
4      display.show(pin12.read_digital())
5      sleep(1000)
6      display.clear()
```

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

Avec MakeCode : codo-LIGNE-EX2.hex

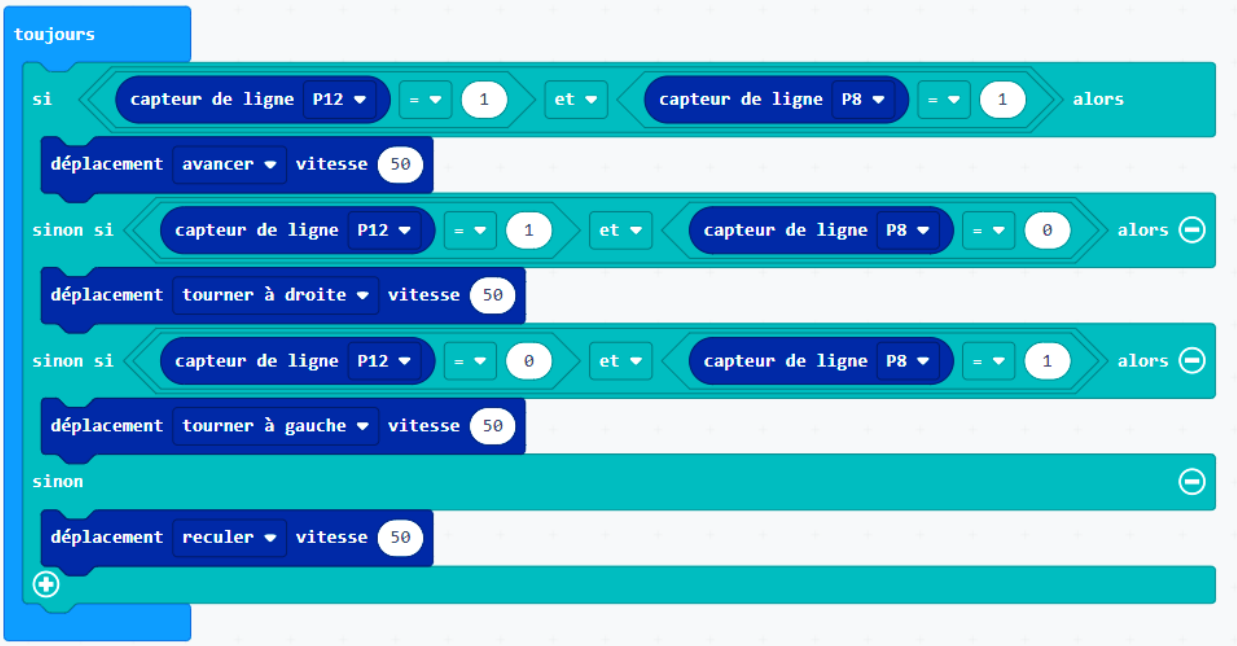


Avec mu Python Editor : codo-LIGNE-EX2.py

```
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)
```

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

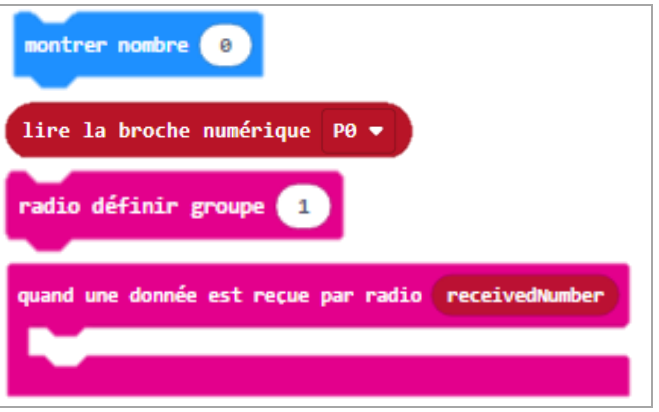
Avec MakeCode : codo-LIGNE-EX3.hex



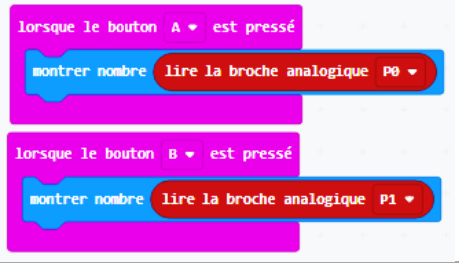
Avec mu Python Editor : codo-LIGNE-EX3.py

```
from microbit import *


while True:
    if pin12.read_digital() == 1:
        pin14.write_analog(250)
        pin13.write_analog(0)
        pin16.write_analog(0)
        pin15.write_analog(250)
    elif pin8.read_digital() == 1:
        pin14.write_analog(0)
        pin13.write_analog(250)
        pin16.write_analog(250)
        pin15.write_analog(0)
    else:
        pin14.write_analog(300)
        pin13.write_analog(0)
        pin16.write_analog(300)
        pin15.write_analog(0)
```

JOYSTICK

Joystick	Instructions / Fonctions utilisées	
		

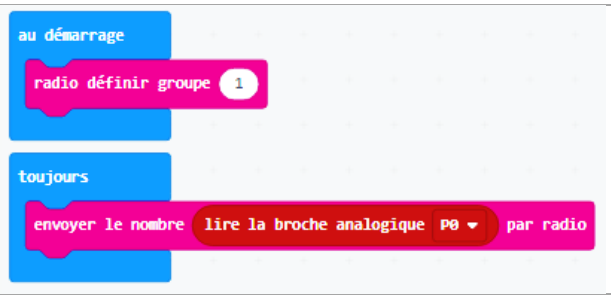
Niv1 : afficher les angles de rotation sur l'axe X lorsque le bouton A est pressé et afficher les angles de rotation sur l'axe Y lorsque le bouton B est pressé

Avec MakeCode : codo-JOYSTICK-EX1.hex	Avec mu Python Editor : codo-JOYSTICK-EX1.py
	

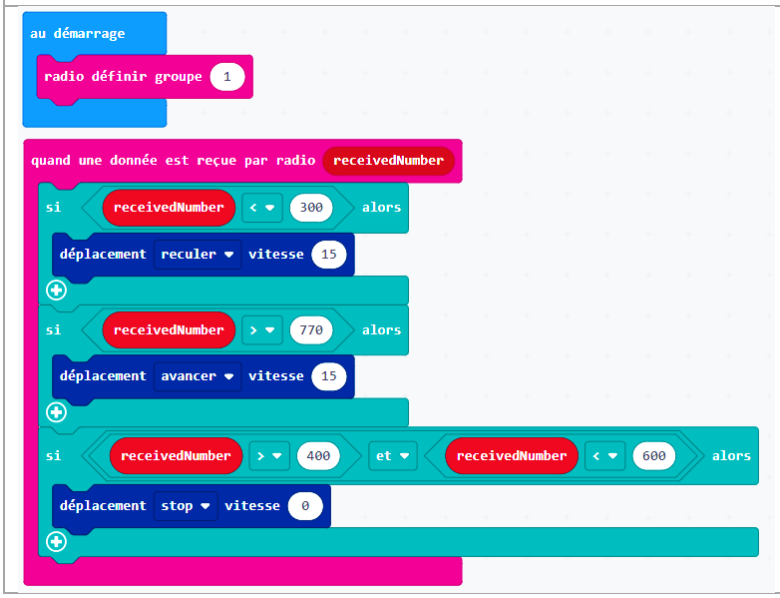
```
from microbit import *

while True:
    if button_a.is_pressed():
        display.scroll(str(pin0.read_analog()))
    elif button_b.is_pressed():
        display.scroll(str(pin1.read_analog()))
```

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

Avec MakeCode : codo-JOYSTICK-EX2A.hex

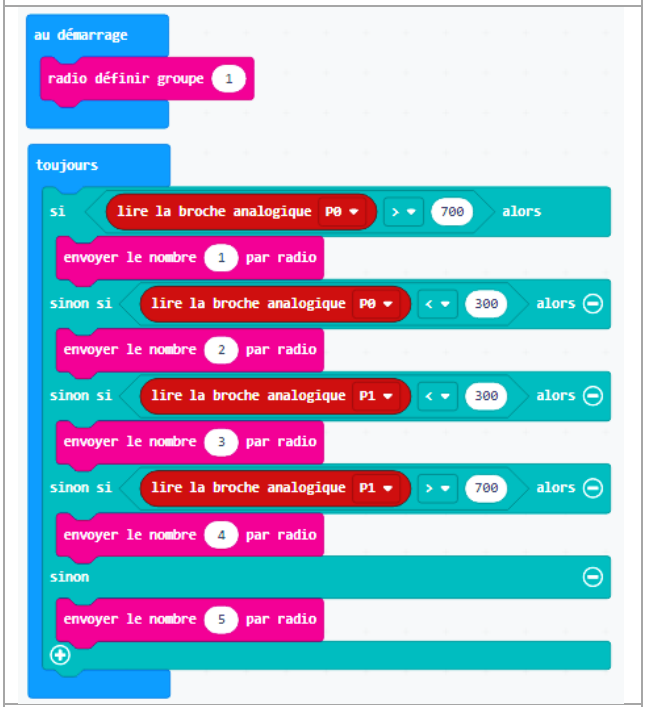


Avec MakeCode : codo-JOYSTICK-EX2B.hex



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

Avec MakeCode : codo-JOYSTICK-EX3A.hex



Avec MakeCode : codo-JOYSTICK-EX3B.hex

Programmation avancée

En blocs

Thèmes	Enoncés	Corrections
Matrice LED	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
Minuteur	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

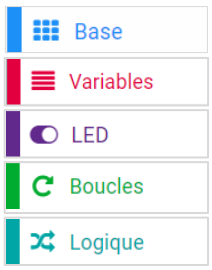
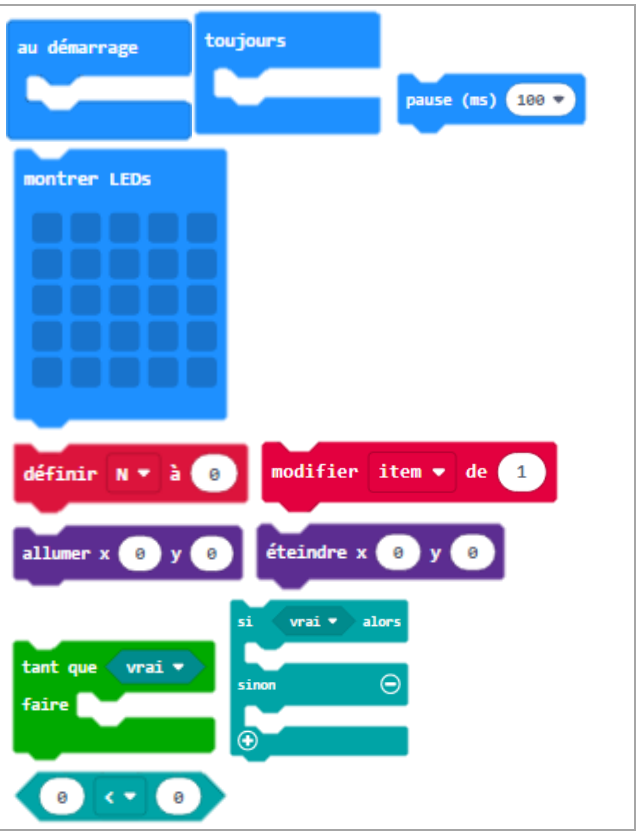
En Python

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

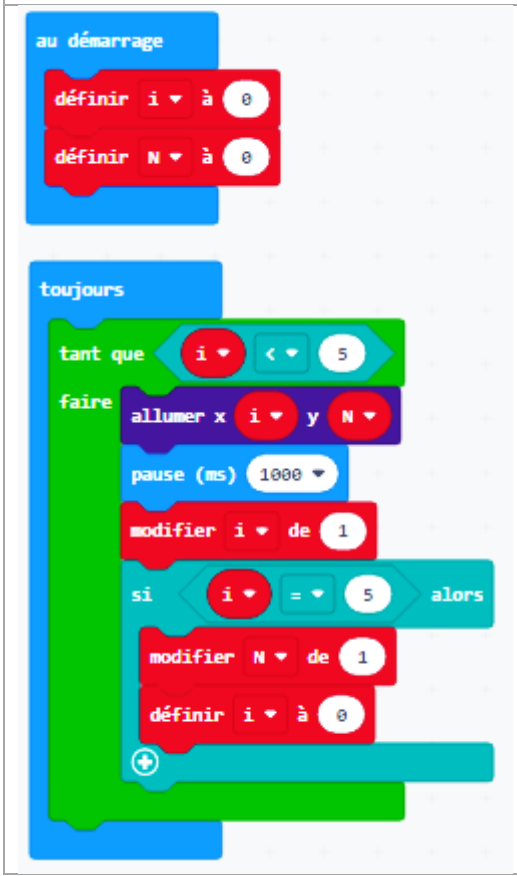
Thème	Enoncés	Corrections
Graphiques	Ex 1 : tracer le graphique de la T° en fonction du temps.	graph temp
	Ex 2 : tracer le graphique d'accélération suivant les 3 axes en fonction du temps.	graph accel
	Ex 3 : tracer le graphique de l'accélération suivant l'axe X en fonction du temps.	accel suivant x

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.

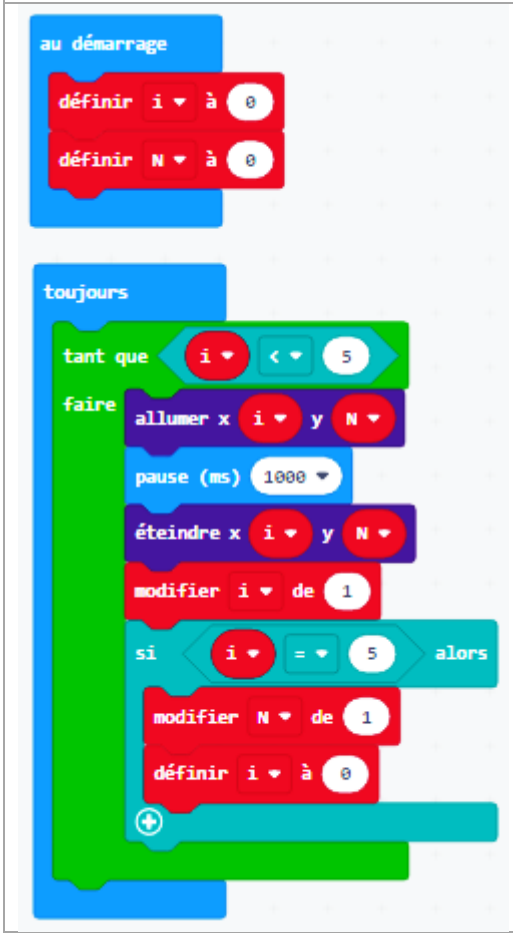
Matrice LED 1 python standard	Ex 1 : afficher un carré.	codo-MOTIFS-EX1
	Ex 2 : afficher un carré et une seconde après un triangle.	codo-MOTIFS-EX2
	Ex 3 : répéter l'affichage d'un carré, d'un triangle, puis d'une croix avec un intervalle de 1 seconde.	codo-MOTIFS-EX3
Bouton A et B python standard	Exercice 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

Module	Instructions / Fonctions utilisées	
		

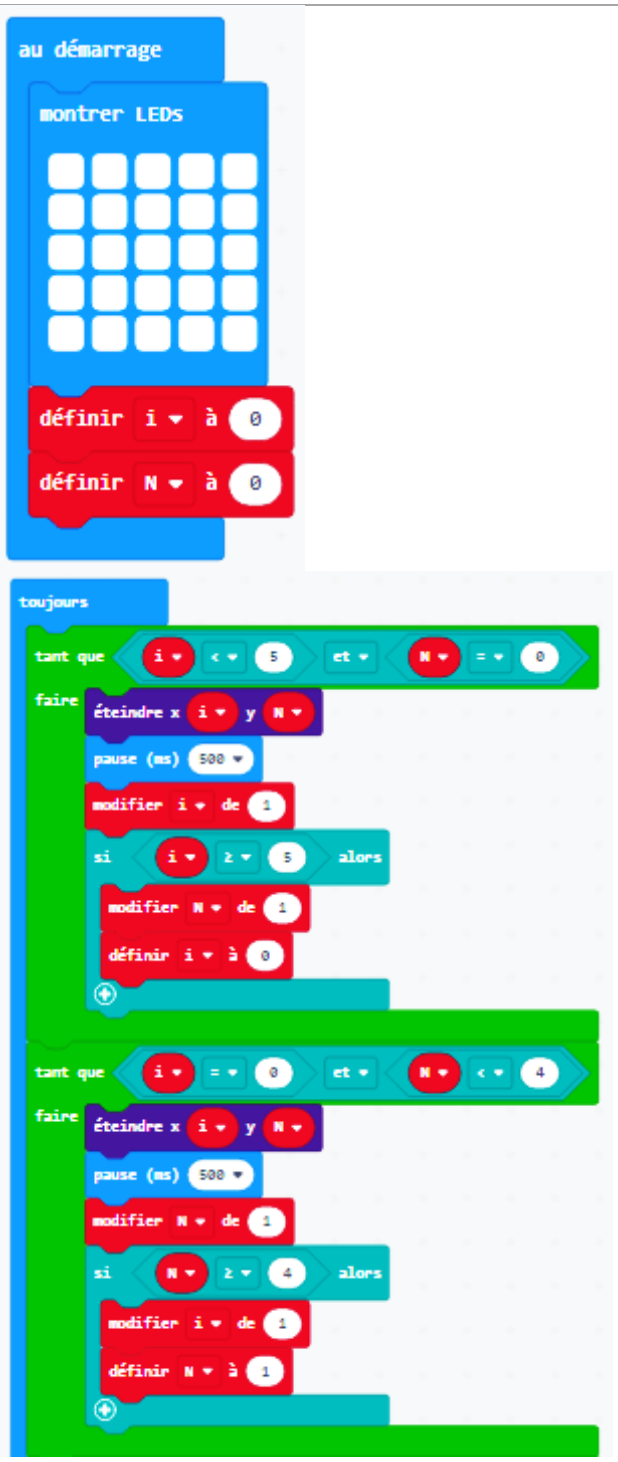
Ex 1 : allumer les LED une par une de gauche à droite et de haut en bas.

Avec MakeCode : codo-MATRICE3-EX1	
	

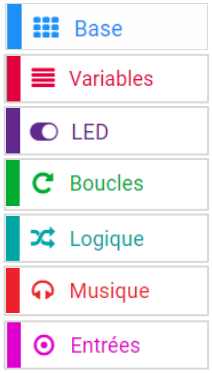
Ex 2 : reprendre l'exercice 1, en désactivant chaque LED 1 seconde après avoir été activée.

Avec MakeCode : codo-MATRICE3-EX2	
	

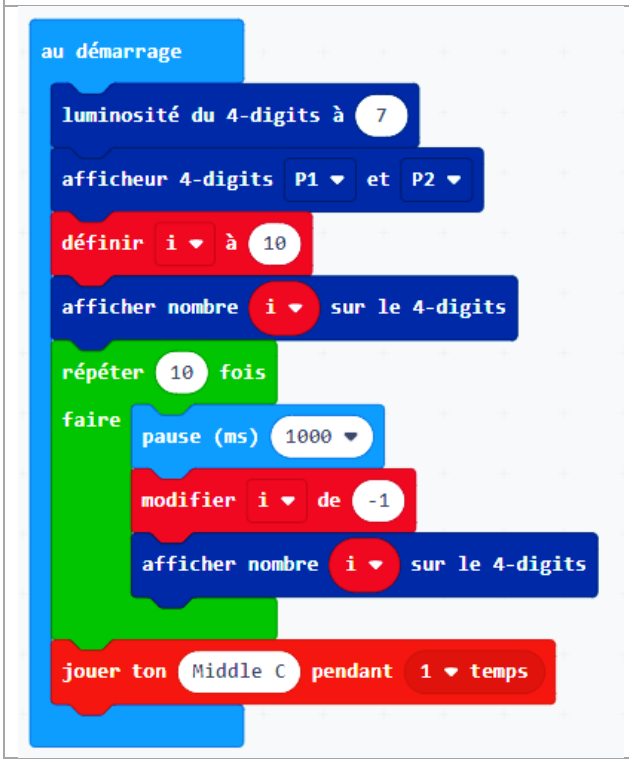
Ex 3 : activer toutes les LED et éteindre certaines LED une par une de telle sorte qu'un L apparaisse.

Avec MakeCode : codo-MATRICE3-EX3	
 Etc.	

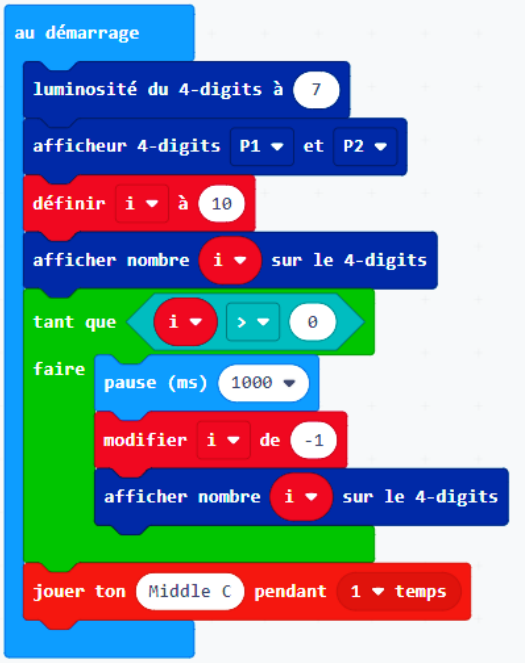
MINUTEUR

Minuteur	Instructions / Fonctions utilisées	
		

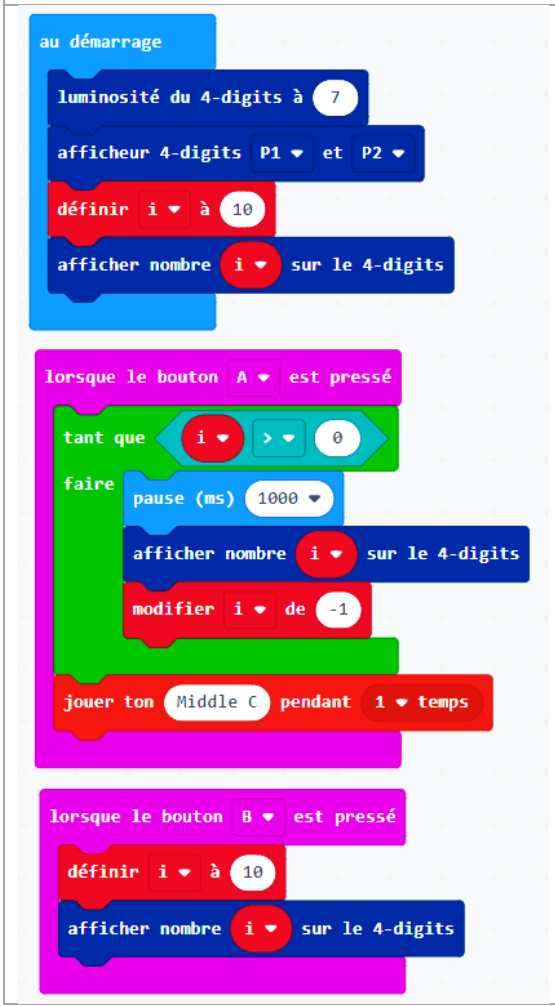
Ex 1 : au lancement du programme déclencher un compte à rebours de 10 secondes et jouer un son lorsqu'il est à zéro.

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	
	

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	
	

GRAPHIQUES

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

En mu Python Editor : graph temp.py

```
from microbit import *

while True:
    temp = temperature()
    print((temp,))
    sleep(1000)
```

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

En mu Python Editor : graph accel

```
from microbit import *

flag = accelerometer.get_values()
while True:
    print(flag)
    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

```
from microbit import *

flag = (accelerometer.get_x(),)
while True:
    print(flag)
    sleep(20)
```

MATRICE LED1 EN PYTHON STANDARD

Ex 1 : afficher un carré

codo-MOTIFS-EX1

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

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

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

BOUTONS A ET B EN PYTHON STANDARD

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

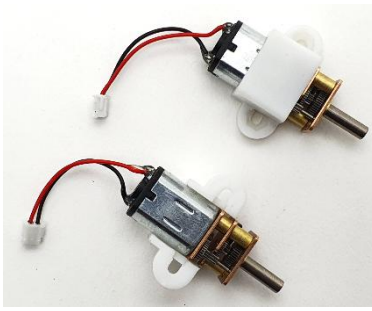
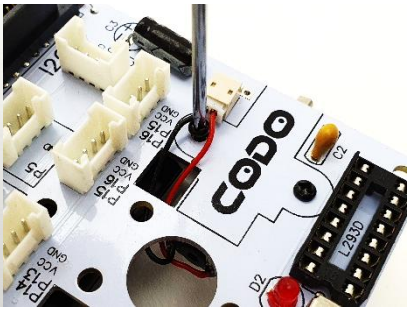
```
def croix():
    color("red")
    begin_fill()
    down()
    left(45)
    forward(35.36)
    up()
    x=25
    y=0
    goto(x,y)
    down()
    left(90)
    forward(35.36)
    up()
    end_fill

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")
```

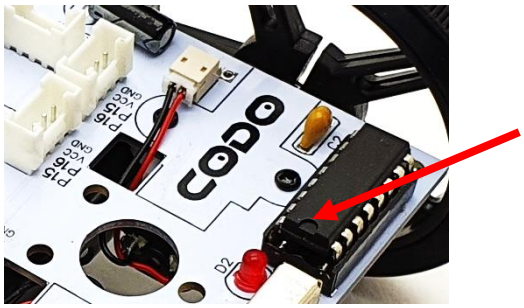
Montage des options et modules sur la CODO

MOTORISATION

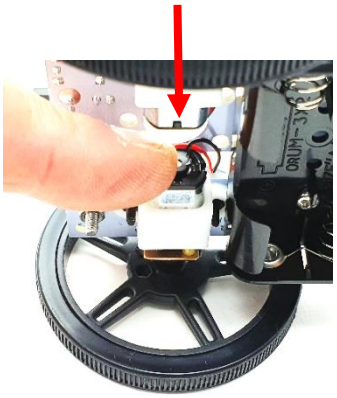
Pour transformer la carte CODO en robot qui se déplace dans toutes les directions.
Cette option permet de mettre la carte CODO en mouvement grâce à des moteurs atteignant une vitesse de 200 tours par minute.
Elle est constituée de 2 motoréducteurs 200 tr/min 100 g.cm, 2 roues Ø 60 mm, 2 supports pour moteurs et 1 boitier 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 L239D
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. Enficher le circuit intégré L293D dans son support. ATTENTION au sens ! Faire coïncider le méplat.

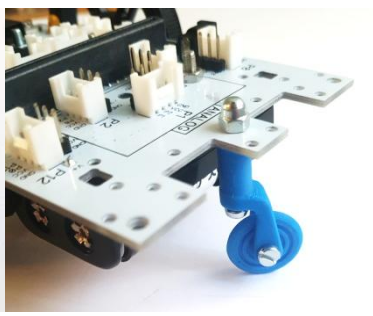





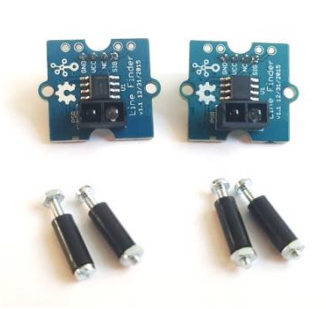
Montage des roues
Mettre en place les roues. Positionner la roue à plat sur l'établi, la faire coïncider avec l'axe moteur (méplat), l'enficher à force en appuyant sur le moteur. Vérifier que chaque roue tourne librement.

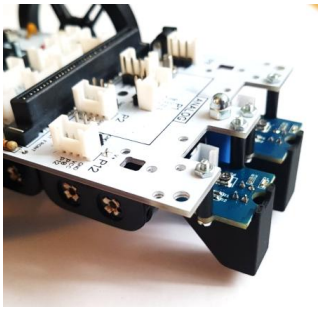



Montage de la roulette avant
Roulette avant pivotante qui permet au robot CODOK de franchir des marches et des obstacles et d'évoluer sur des surfaces difficiles ou accidentées.
Constitué de deux pièces imprimables en 3D (fourche et roue), d'un tube laiton 3x4 coupé à L 16 mm, d'une vis TC M3 x 8, d'une 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

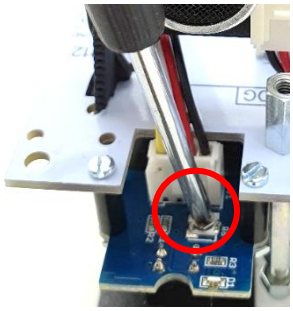
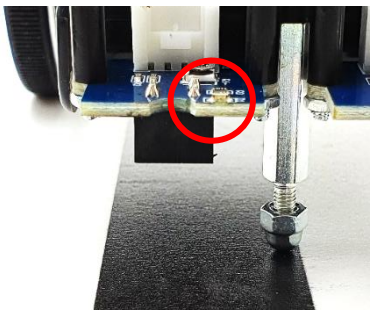
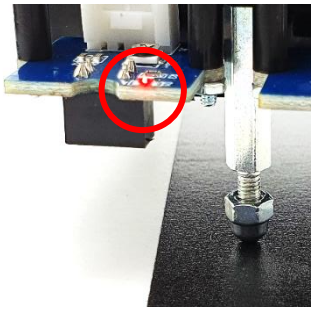



Montage des 2 capteurs de ligne (réf S-101020009) et carters infrarouge imprimés en 3D
Les carters infrarouges permettent d'améliorer les performances des capteurs de ligne. Il suffit de les enficher sur chaque module.





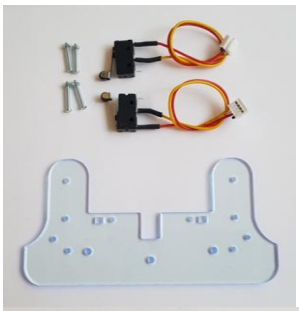
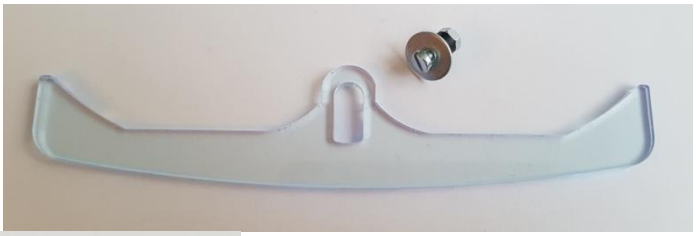
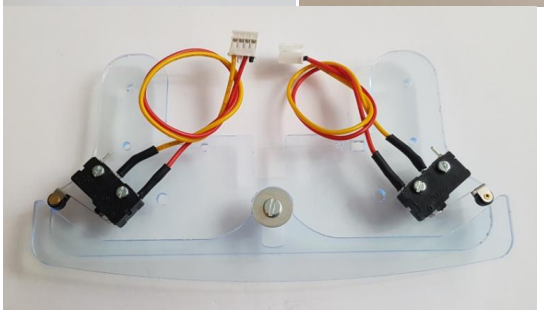
Réglage des capteurs de ligne
Tourner le potentiomètre 5 fois dans le sens antihoraire.
Positionner le capteur au-dessus d'une ligne noire et tourner le potentiomètre dans le sens horaire jusqu'à ce que la LED s'éteigne.
Une fois cette condition validée, positionner le capteur au-dessus d'une surface 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.

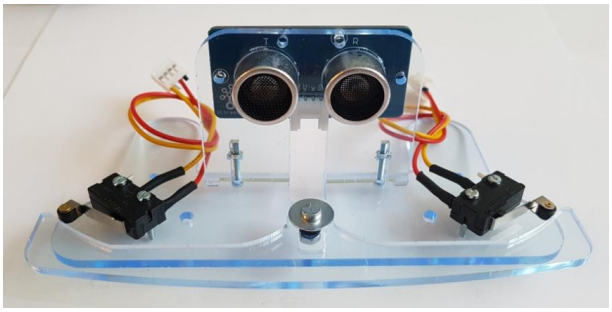
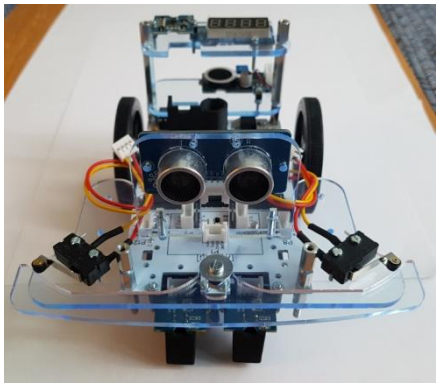
Elle est composée de 6 plaques de PMMA épaisseur 3 mm pour la fixation des différentes options et des modules Grove utiles pour aller plus loin.

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.

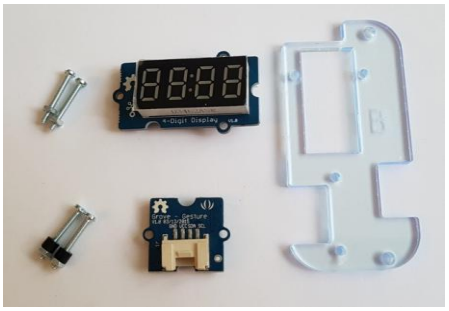
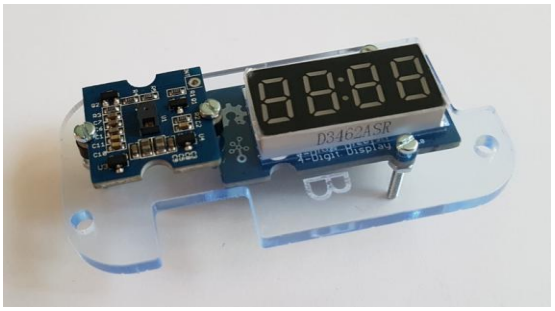




Montage du module Ultrasons (S-101020010)
Plaque PMMA et 1 télémètre à ultrasons (non fourni).
Pour la fixation du module : 3 vis Ø2 x h 12 mm, 3 écrous Ø2 mm.

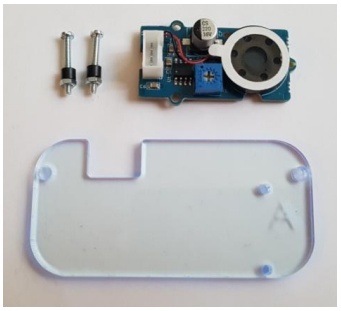
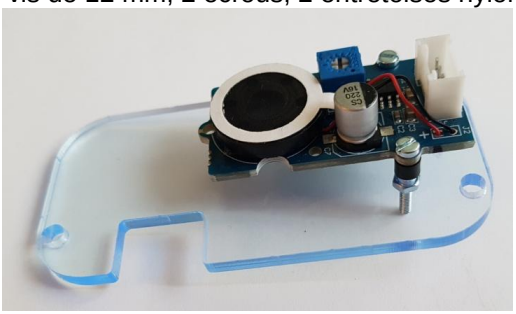

Fixation de la plaque : 2 vis Ø2 x h16 mm, 2 écrous Ø 2 mm.

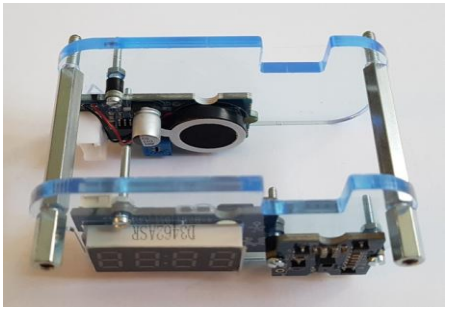
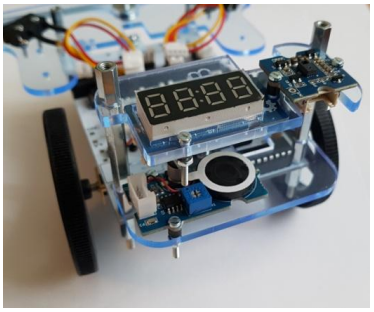
Montage afficheur 4 digits et capteur de mouvement
Plaque B, 1 afficheur 4 digits (non fourni) et 1 capteur de mouvement (non fourni).
Pour la fixation : 2 vis de 12 mm, 2 vis de 8 mm, 4 écrous, 2 entretoises nylon.

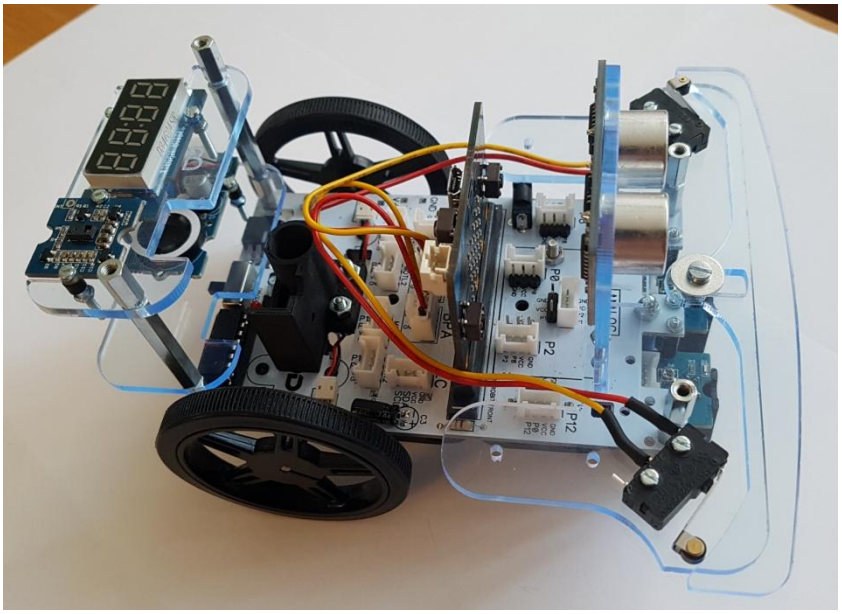



Montage speaker
Plaque A et 1 module Speaker (non fourni).
Pour la fixation du module : 2 vis de 12 mm, 2 écrous, 2 entretoises nylon.

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

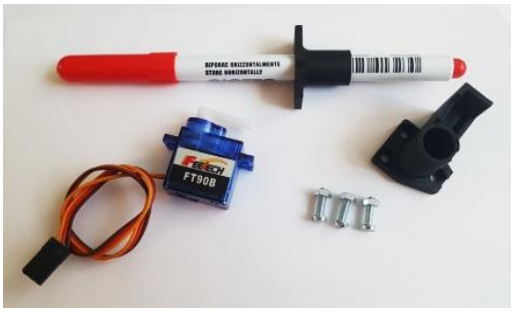



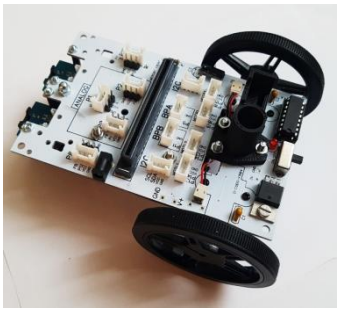


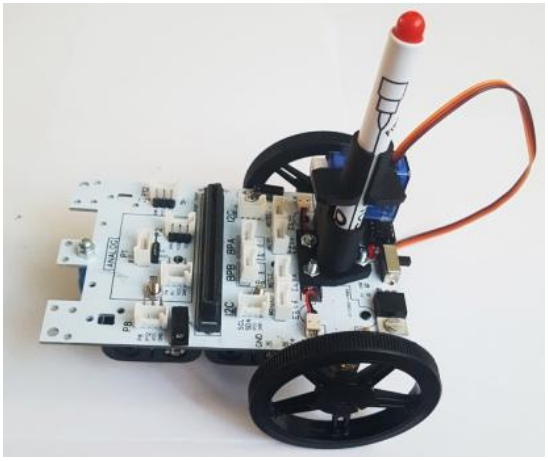
OPTION PORTE-STYLO

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.

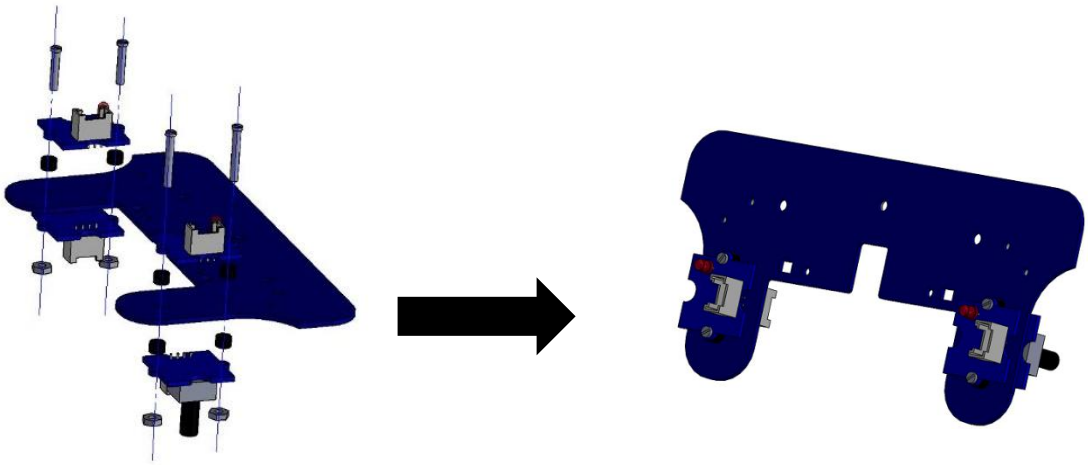






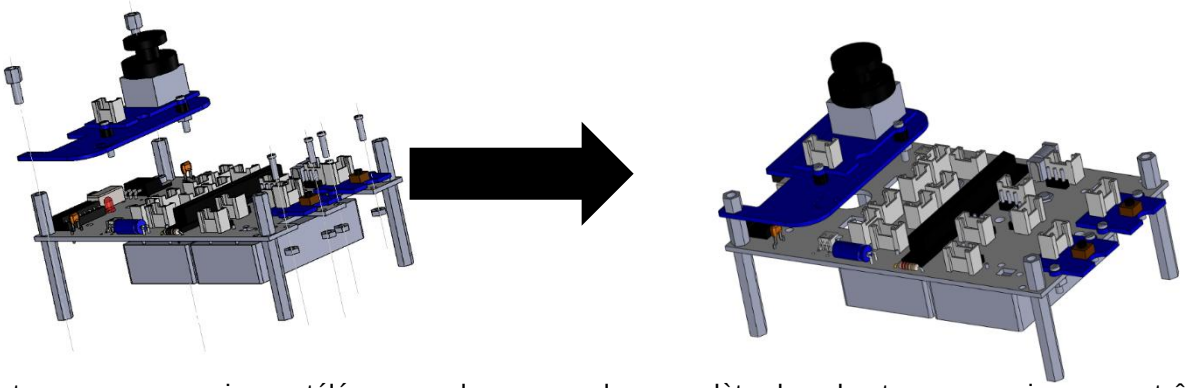
LED, potentiomètre et bouton-poussoir

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