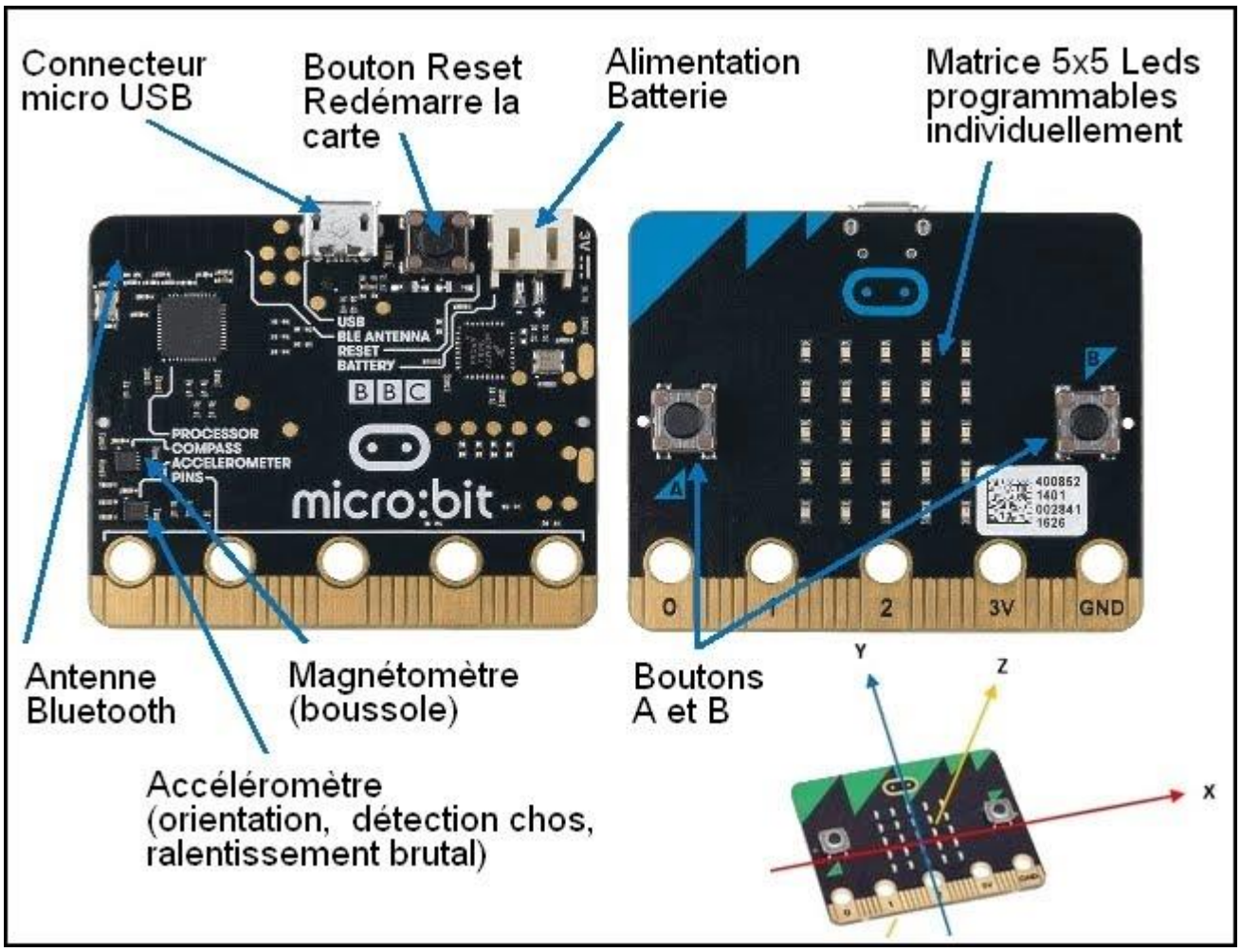
Réaliser un compteur

➢ PRÉSENTATION DE LA CARTE

L'objectif de cette activité est de réaliser un prototype de compteur de personnes pour un supermarché. Un vigile, situé à l'entrée du magasin aura pour mission de comptabiliser les personnes entrant dans le commerce. À chaque fois qu'un client entre, le vigile doit appuyer sur le bouton « A » du prototype de compteur.

Le prototypage s'articulera autour d'un nano-ordinateur « **Micro:bit** ».



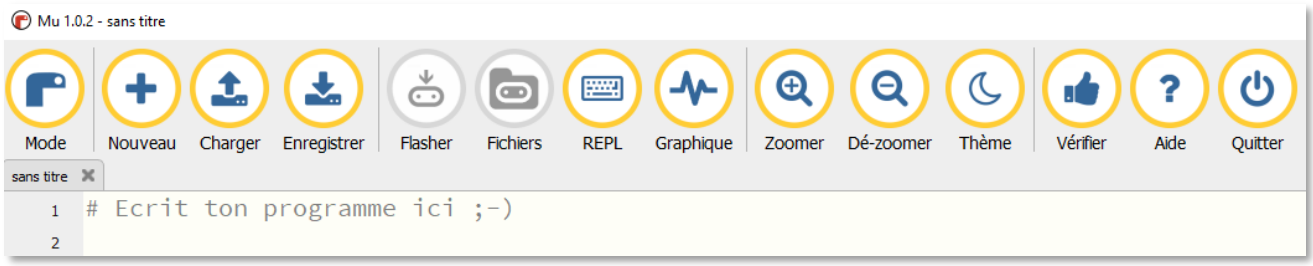
La carte « **Micro:bit** » est équipée d'interfaces d'entrée (bouton poussoirs, accéléromètre, boussole,…) et de sortie (matrice à leds) et de communication Bluetooth entre autre.

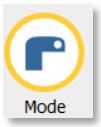
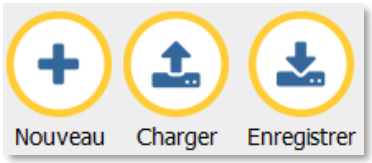
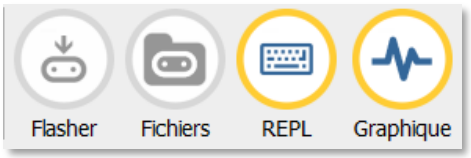
Voici un aide mémoire pour programmer votre Micro:Bit :

Clic droit puis ouvrir dans un nouvel onglet

➢ PROGRAMMATION

Vous allez programmer la carte Micro:bit à l'aide de l'éditeur « **MU** ».
Exécutez le logiciel « **MU** » depuis le raccourci du bureau « *Autre raccourcis/SNT* »
L'interface « **MU** » se présente ainsi :



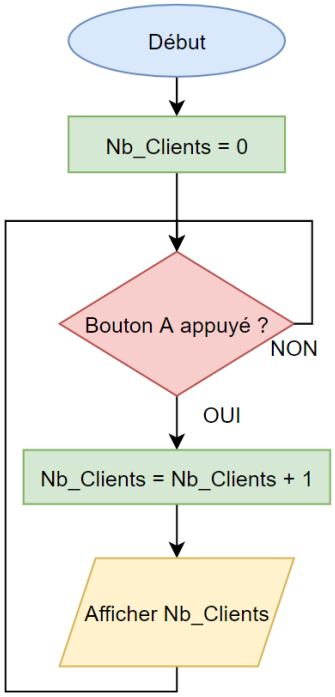
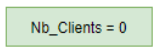
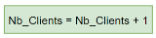
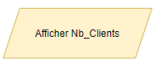
	Ce bouton permet de changer de mode de programmation. **Vous devez choisir le mode « BBC micro:bit ».**
	Ces trois boutons permettent de gérer les fichiers programme réalisés vis-à-vis du support de stockage. Il est possible de charger plusieurs programmes ; chacun s'ouvrira dans un onglet spécifique.
	Le bouton « **Flasher** » permet de transférer le programme en cours d'édition dans la mémoire du **Micro:bit** si ce dernier est connecté à l'ordinateur via USB. Le bouton « **Fichiers** » permet de visualiser les fichiers contenu sur la carte et sur le disque dur. Le bouton « **REPL** » permet d'afficher la console (Read, Evaluate, Print, Loop) Le bouton « **Graphique** » permet d'afficher des courbes lorque le programme en génère.
	Zone de saisie du code du programme. Plusieurs onglets peuvent être affichés si plusieurs programmes ont été ouverts.

➢STRUCTURE DU PROGRAMME

Le déroulement du programme (algorithme) sera donc le suivant :

Stocker la valeur 0 dans la variable Nb_Clients ;
Répeter indéfiniment :
    Si le bouton « A » a été pressé :
        Incrémenter d'une unité la valeur stockée dans la variable Nb_Clients
        Afficher la valeur contenue dans la variable Nb_Clients
    Fin Si
Fin Répéter

À lire attentivement !

Algorigramme de la structure du programme	Symbole	Signification	Code associé
	 1 flèche en sortie	Indique le début du programme	from microbit import * while True: ***Lignes de code programmées un seule fois et en début de programme***
	 1 entrée et une sortie	**CALCUL** Affecte la valeur 0 à la variable Nb_Clients	Nb_Clients = 0
	 1 entrée et 2 sorties (OUI/NON)	**TEST** Teste **SI** le bouton A à été pressé	**if** button_a.was_pressed()**:**
		**CALCUL** Augmente la valeur contenue dans la variable Nb_Clients	Nb_Clients=Nb_Clients+1
		**AFFICHAGE** Afficher la valeur contenue dans la variable Nb_Clients	display.show(str(Nb_Clients)

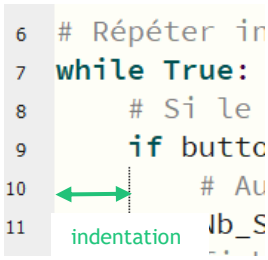
➢CODAGE DU PROGRAMME

Voici le code du programme que vous devez saisir :

```
1  from microbit import *
2
3  # Stocker la valeur zéro dans la variable Nb_Clients
4  Nb_Clients = 0
5  # Afficher la valeur contenue dans la variable Nb_Clients
6  # Cette valeur doit être convertie en texte : str(....)
7  display.show(str(Nb_Clients))
8
9  # Répéter indéfiniment
10 while True:
11     # Si le bouton "button_a" a été appuyé
12     if button_a.was_pressed():
13         # Augmenter la valeur de Nb_Clients de 1 unité
14         Nb_Clients = Nb_Clients + 1
15         display.show(str(Nb_Clients))
```

Les lignes qui commencent par le symbole « # » sont des lignes de commentaire ; elles ne sont ni analysées ni exploitées par le programme.

Attention aux minuscules et majuscules, espaces, **indentation**, etc…
L'indentation se fait en appuyant sur « tabulation » du clavier :




```
12     if button_a.was_pressed():
13         # Augmenter la valeur de Nb_Clients de 1 unité
14         Nb_Clients = Nb_Clients + 1
15         display.show(str(Nb_Clients))
```

Dans le programme, les lignes 14 et 15 ne seront exécutées que si la condition « *button_a.was_pressed()* » est vraie car ces deux lignes sont **INDENTÉES** par rapport à la ligne 12 (ligne de test).

Si la ligne 15 n'était pas indentée, elle serait exécutée même si l'on a pas appuyé sur le bouton A !

L'indentation est donc très importante lors du codage de votre programme.

➢TEST DE VOTRE PROGRAMME

Avant de flasher votre programme dans la mémoire de la carte **Micro:bit**, il est préférable de cliquer sur le bouton « **Vérifier** ». Ceci permet de vérifier les erreurs de syntaxe et les erreurs d'indentation.

Lorsque tout semble correct, il est grand temps d'enregistrer votre programme dans le dossier « Document/SNT/THEME_6 ». Donnez-lui un nom significatif ; je vous propose un nom de ce type : « ***a63-votrenom.py*** ».
Pas de prénom !

Vous devez, dès à présent, utiliser le bouton « **Flasher** » afin de transférer votre code dans la mémoire de l'ordinateur vers celle du **Micro:bit**.

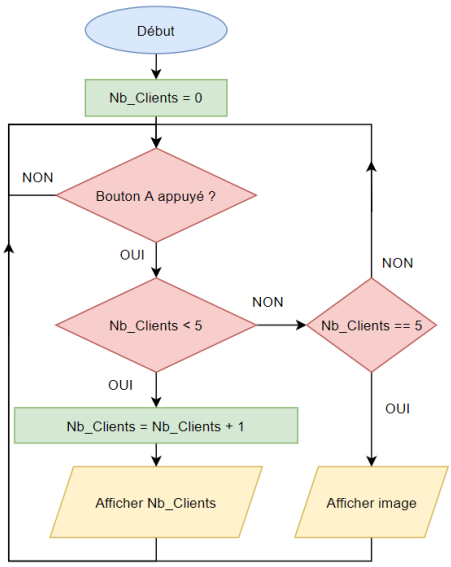
Il ne vous reste plus qu'à tester le fonctionnement : lorsque vous appuyez sur le bouton « **A** » la valeur affichée sur la matrice à LEDs doit augmenter.

Vous pouvez remettre votre compteur à 0 en appuyant sur « **Reset** » (voir page 1).

Tout fonctionne à merveille ? Excellent, vous avez franchi la première étape.

➢ AMÉLIORATIONS

1. Gestion du nombre maximal de clients :



En raison d'une épidémie, le nombre de maximal de clients est fixé à 5 (pour réduire le temps de comptage), il serait intéressant d'en informer le vigile. Je vous propose de modifier ce dernier programme afin que lorsque le 5ème client vient d'être compté, une image s'affiche sur la matrice à LED (image de votre choix).

Modifiez votre programme afin de mettre en place cette limite. Les nouvelles lignes de code doivent être saisies juste après la dernière ligne tapée précédemment.

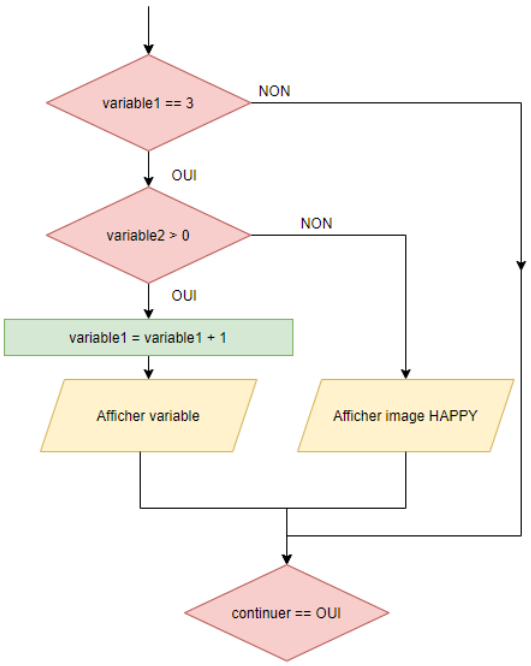
Une petite aide par ici :
https://lecluseo.scenari-community.org/CircuitPython/co/g_leds.html
(clic droit puis ouvrir dans un nouvel onglet)

Lorsqu'un test () est réalisé, l'instruction à coder est if.

Toutes les lignes indentées seront exécutées si la condition du test est vérifiée.

Pour continuer le programme sans condition, il suffit de supprimer l'indentation.

Exemple :



Le code ci-dessous n'est pas à saisir dans votre programme !

```
# Si la variable "variable1" contient la valeur 3
if variable1 == 3:
    # Si la variable "variable2" contient une valeur > 0
    if variable2 > 0:
        # Augmenter la valeur de "variable1" de 1 unité
        variable1 = variable1 + 1
        # Afficher la valeur contenue dans "variable1"
        display.show(str(variable1))
    if variable2 <= 0:
        # Afficher l'image "HAPPY"
        display.show(Image.HAPPY)
# Si la variable "continuer" vaut "OUI"
if continuer == "OUI":
    # Poursuite du programme...
```

Si « variable1 » vaut 3, le bloc vert est exécuté sinon le programme continue à la ligne « if continuer == « OUI » ;

Le code jaune est exécuté si « variable1 » vaut 3 **ET** que « variable2 » est supérieure à 0 ;

Le code bleu est exécuté si « variable1 » vaut 3 **ET** que « variable2 » est inférieure ou égale à 0.

Testez votre système et lorsque tout semble correct, il est grand temps d'enregistrer votre programme.

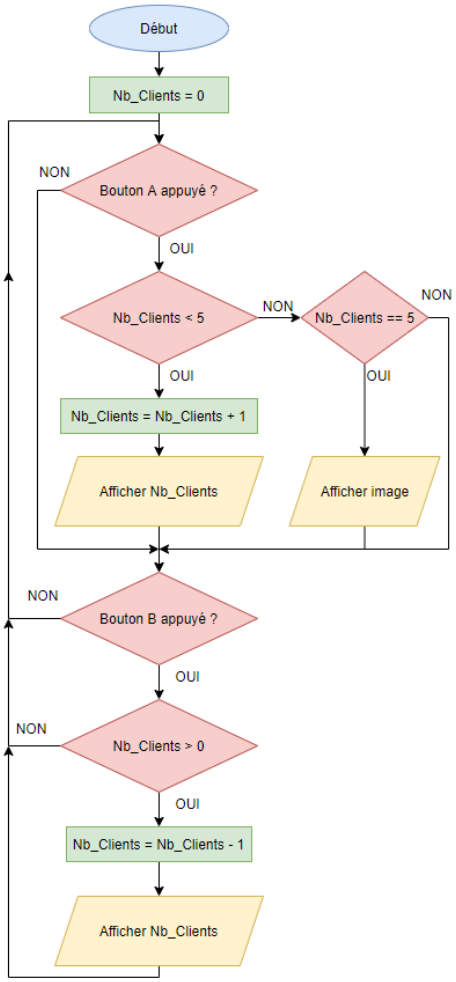
Tout fonctionne à merveille ? Excellent, vous avez franchi la troisième étape.

2. Gestion des erreurs et des sorties :

Si le vigile chargé d'effectuer le comptage des clients entrés se trompe en appuyant une fois de trop sur le bouton « A » ou bien si des clients comptés ressortent lorsqu'ils ont terminé leurs achats, le compte n'est plus correct.

Le programme initial ne permet pas de gérer cette situation ; vous devez donc compléter votre code afin de permettre la réduction de la valeur contenue dans la variable *Nb_Clients* lorsque c'est utile. Bien entendu, il ne faut pas compter un nombre négatif de clients !

Voici l'algorigramme correspondant à cette amélioration →



Complétez votre code afin de gérer cette situation.

Testez votre système et lorsque tout semble correct, il est grand temps d'enregistrer votre programme.

Tout fonctionne à merveille ? Excellent, vous avez franchi la seconde étape.

Faites valider le fonctionnement par votre professeur.

Enregistrez votre programme : 

puis choisissez de créer un nouveau programme : 

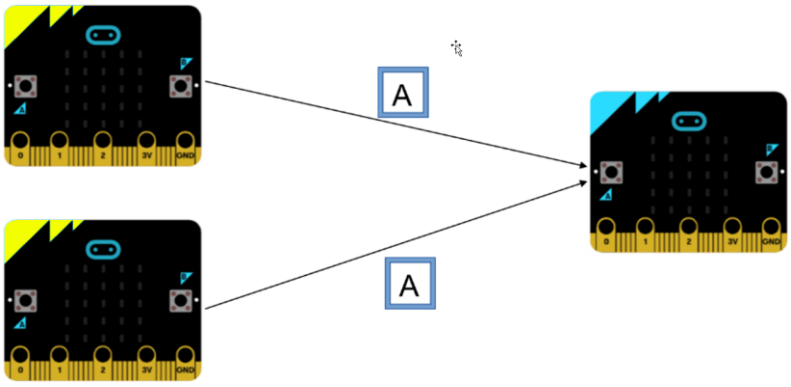
3. Il existe plusieurs entrés/sorties.

Le supermarché dispose de plusieurs issues ; il faut donc utiliser plusieurs cartes Micro:Bit pour compter les allées/venues des clients et une carte Micro:Bit pour centraliser le nombre total de clients présents dans le supermarché. Vous allez, pour cela, utiliser la fonction « **radio** » de la carte **Micro:Bit** entre les différentes cartes.

Utilisation de la fonction radio :

- Définir un protocole de communication ;
- Concevoir le programme sur les cartes émettrices ;
- Concevoir le programme sur la carte réceptrice ;
- Tester le fonctionnement du système.

Définir le protocole de communication :



- La carte réceptrice BLEUE augmente la valeur du compteur à chaque réception du message « A ».
  Vous disposez d'une carte BLEUE à utiliser par l'un d'entre-vous.

- Les cartes émettrices JAUNES envoie le message « A » à la carte réceptrice lorsque le bouton « A » est pressé.
  Vous disposez de trois cartes JAUNES à partager en fonction du nombre d'élèves restant.

Algorithmes :

Cartes émettrices	Carte réceptrice

Configurer la radio
Répéter indéfiniment :
  Si bouton_A pressé :
  Envoyer le message 'A'
	Configurer la radio
	Nb_Clients ← 0
	Répéter indéfiniment :
	  Si on reçoit le message 'A' :
	  Nb_Clients ← Nb_Clients + 1
	  Si bouton_A pressé :
	  Afficher Nb_Clients

Programmes :

Cartes émettrices	Carte réceptrice

from microbit **import** *
import radio
# Configuration radio 	
radio.config(group=1)
# Remplacer le 1 par votre numéro de Pôle 	
radio.on()
 	
while True:
  if button_a.was_pressed():
  radio.send("A")
	from microbit **import** *
	import radio
	# Configuration radio
	radio.config(group=1)
	# Remplacer le 1 par votre numéro de Pôle
	radio.on()
	
	Nb_Clients = 0
	while True:
	  reception = radio.receive()
	  if reception == "A":
	  Nb_Clients = Nb_Clients + 1
	  if button_a.was_pressed():
	  display.scroll(str(Nb_Clients))
	  sleep(2000)
	  display.clear()

Testez votre système et lorsque tout semble correct, il est grand temps d'enregistrer vos programmes. Donnez-leur un nom significatif ; je vous propose un nom de ce type : « ***a63e-votrenom.py*** » pour les cartes émettrices et « ***a63r-votrenom.py*** » pour la carte réceptrice.

Tout fonctionne à merveille ? Excellent, vous avez franchi une nouvelle étape.

Faites valider cette amélioration par votre professeur.

4. Les clients sortent du supermarché.

Améliorez vos programmes afin de gérer la sortie des clients. Le décomptage s'effectue sur la carte réceptrices lorsque la lettre « **B** » est reçu. Le vigile doit appuyer sur le bouton le bouton « **B** » !

Rappel : il ne paut pas y avoir u nombre négatif de clients dans le supermarché !

Tout fonctionne à merveille ? Excellent, vous avez franchi une nouvelle étape.

Faites valider cette amélioration par votre professeur.

5. Le nombre de clients est limité.

Comme précédement, le nombre de clients présents dans le supermarché est limité à 5 personnes. La carte réceptrice doit donc envoyer une information vers toutes les cartes des vigiles afin de les informer de ne plus laisser entrer de clients à l'intérieur du supermarché lorsque le nombre limite de clients est atteint.

Compléter vos programmes afin de gérer cette nouvelle situation.

Tout fonctionne à merveille ? Excellent, vous avez franchi une nouvelle étape.

Faites valider cette amélioration par votre professeur.