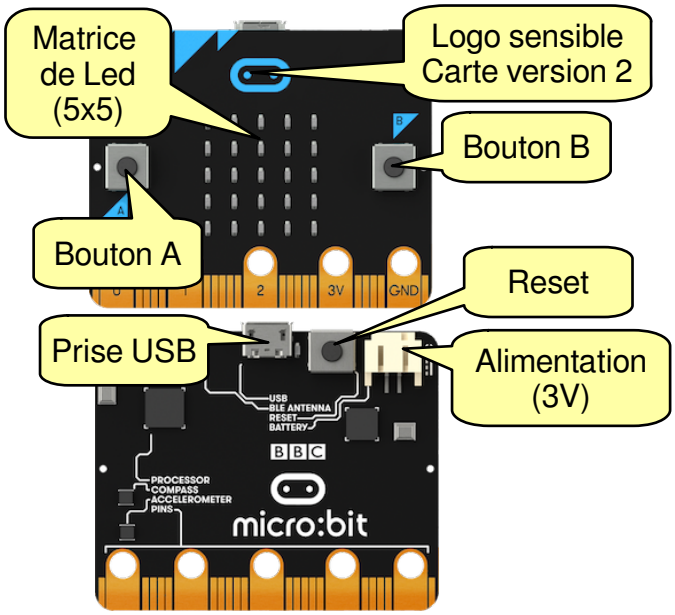
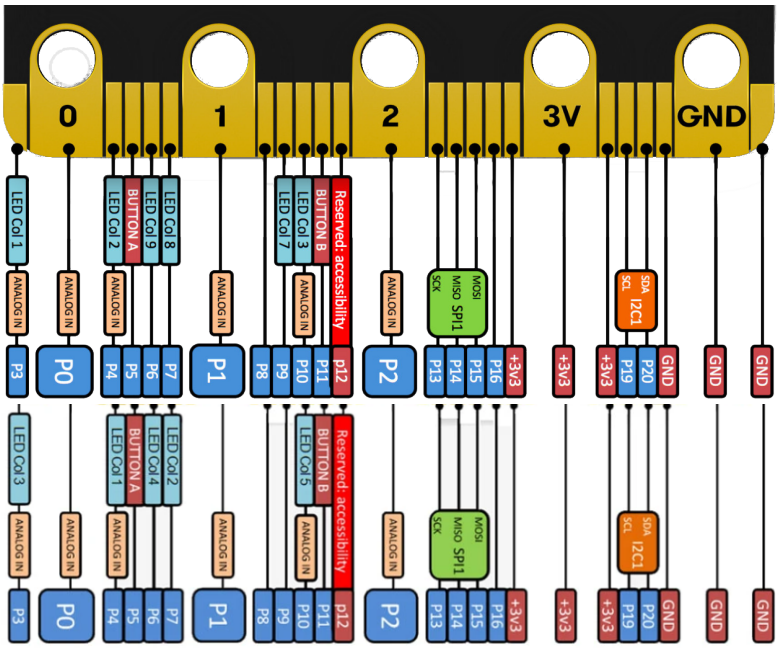




1. BIBLIOTHÈQUE

Tout programme commence par l'import de la bibliothèque des fonctions de la carte :

```
from microbit import *
```

Brochage des entrées/sorties

- Carte version 1
- Carte version 2

2. MATRICE DE LEDS

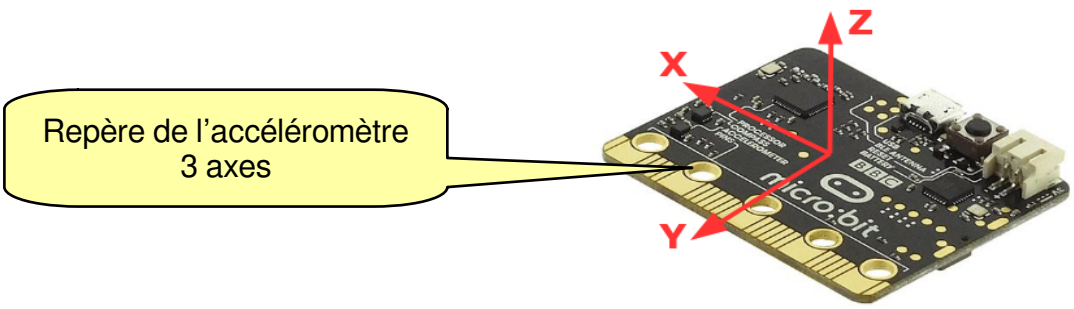
Commandes	Instruction Python	Remarques
Allumer, éteindre l'écran		
Effacer l'écran	`display.clear()`	
Allumer l'écran	`display.on()`	Utile lorsqu'on veut récupérer les broches E/S (3,4,5,7,9,10)
Éteindre l'écran	`display.off()`	
Messages		
Afficher un message caractère par caractère	`display.show(23)`	
Afficher un message avec défilement	`display.scroll("Hello!", wait=False,loop=True)`	`wait`: en mode bloquant `loop`: répétition
Afficher / Lire des pixels		
Retourne l'état de la led colonne `x` et ligne `y`	`display.get_pixel(x,y)`	Retourne un entier de 0 (éteint) à 9 (complètement allumé)
Allume la led colonne `x` et ligne `y`	`display.set_pixel(x,y, level)`	`level` : niveau de 0 (éteint) à 9 (complètement allumé)

Commandes	Instruction Python	Remarques
Afficher une image prédéfinie		
Afficher une image prédéfinie	`display.show(Image.HAPPY)`	
`'HEART','HEART_SMALL','HAPPY','SMILE','SAD','CONFUSED','ANGRY','ASLEEP','SURPRISED','SILLY','FABULOUS','MEH','YES','NO','CLOCK12','CLOCK1','CLOCK2','CLOCK3','CLOCK4','CLOCK5','CLOCK6','CLOCK7','CLOCK8','CLOCK9','CLOCK10','CLOCK11','ARROW_N','ARROW_NE','ARROW_E','ARROW_SE','ARROW_S','ARROW_SW','ARROW_W','ARROW_NW','TRIANGLE','TRIANGLE_LEFT','CHESSBOARD','DIAMOND','DIAMOND_SMALL','SQUARE','SQUARE_SMALL','RABBIT','COW','MUSIC_CROTCHET','MUSIC_QUAVER','MUSIC_QUAVERS','PITCHFORK','XMAS','PACMAN','TARGET','ALL_CLOCKS','ALL_ARROWS','TSHIRT','ROLLERSKATE','DUCK','HOUSE','TORTOISE','BUTTERFLY','STICKFIGURE','GHOST','SWORD','GIRAFFE','SKULL','UMBRELLA','SNAKE'`		
Créer sa propre image		
Définir l'image avec une variable texte	`monImage = Image("90009:06060:00300:06060:90009")`	Les valeurs sont les niveaux de chaque pixel (0 (éteint) à 9 (complètement allumé))
Afficher l'image	`display.show(monImage)`	
Afficher des animations		
Afficher une animation avec une liste d'image	`display.show([Image.SAD, Image.HAPPY], delay=1000, loop=True, wait=False)`	**`delay`**: attente entre deux images en milliseconde **`loop`**: répétition de l'animation **`wait`**: en mode bloquant
Capteur de lumière		
Mesure le niveau de lumière	`display.read_light_level()`	La matrice de leds est utilisée comme capteur de lumière.

3. BOUTONS

Commandes	Instruction Python	Remarques
Bouton pressé présentement		
Bouton A pressé	`if button_a.is_pressed():` `  display.show("A")`	
Bouton B pressé	`if button_b.is_pressed():` `  display.show("B")`	
Logo touché	`if pin_logo.is_touched():` `  display.show("L")`	Uniquement sur la version 2 de la carte
Bouton actionné pendant que le programme était occupé à une autre tâche		
Bouton A a été pressé	`if button_a.was_pressed():` `  display.show("A")`	Par exemple, la commande **`scroll`** en mode bloquant (**`wait=True`**) ne détecte pas l'appui du type **`is_pressed`** .
Bouton B a été pressé	`if button_b.was_pressed():` `  display.show("B")`	
Comptage du nombre de pressions		
Nombre de pressions sur le bouton A	`button_a.get_presses()`	Le compteur est remis à 0 à chaque invocation de la commande. Il convient donc de stocker cette information dans une variable.
Nombre de pressions sur le bouton B	`button_b.get_presses()`	

4. CAPTEURS INTÉGRÉS



Commandes	Instruction Python	Remarques
Capteur de température		
Mesure de la température	`temperature()`	En degrés Celsius
Capteur sonore		
Mesure du bruit	`microphone.sound_level()`	Valeur comprise entre 0 et 255 Uniquement sur la version 2 de la carte
Magnétomètre (boussole)		
Calibrer la boussole	`if not(compass.is_calibrated()):` `  compass.calibrate()`	
Valeur de l'orientation	`compass.heading()`	En degrés par rapport au nord
Lors du calibrage de la boussole, le message 'Tilt to fill screen' est affiché, il alors faut manipuler la carte jusqu'à allumer les 25 leds.		
Accéléromètre		
Détection d'une inclinaison en cours	`accelerometer.current_gesture()`	
up : incliné vers l'avant, **down** : vers l'avant, **left** : vers la gauche, **right** : vers la droite, **face up** : à l'endroit, **face down** : à l'envers		
Détection d'un mouvement en cours	`accelerometer.current_gesture()`	
shake : secoué, **freefall**: chute libre, **3g** : accélération faible, **6g** : accélération moyenne, **8g** : accélération forte		
Mesure d'une accélération en x	`accelerometer.get_x()`	`get_y()` et `get_z()` pour y et z
Test sur le geste en cours	`if accelerometer.is_gesture("up"):` `  display.scroll("up")`	
Test sur un geste qui a eu lieu	`if accelerometer.was_gesture("up"):` `  display.scroll("up")`	
Liste des gestes passés	`accelerometer.get_gestures()`	

5. HORLOGE

Commandes	Instruction Python	Remarques
Attente	`sleep(1000)`	Temps en miliseconde
Temps écoulés	`running_time()`	Temps en miliseconde écoulé depuis la mise en marche de la carte

6. COMMUNICATION ENTRE PLUSIEURS CARTES

En début de programme, il faut l'importer la bibliothèque radio : **`import radio`**

Commandes	Instruction Python	Remarques
Envoyer et recevoir		
Activer la radio	`radio.on()`	Par défaut désactivée pour des raisons d'économie d'énergie
Désactiver la radio	`radio.off()`	
Envoyer un message	`radio.send("SOS")`	Message toujours sous forme d'une chaîne de caractères
Recevoir un message	`radio.receive()`	

Mesurer la force du signal radio reçu	`msg= radio.receive_full()`
	`if msg:`
	`  signal = msg[1]`
		Valeur comprise entre -98 (signal faible) et -45 (signal fort)

Définir les destinataires		
Définition de l'adresse et du groupe	`radio.config(address=1, group=7)`	Seules les cartes avec la même adresse et le même groupe communiqueront.
Si l'adresse et le groupe ne sont pas définis (par défaut), la carte envoie les messages à l'ensemble des cartes (broadcast).		
Configuration		
Nombre de messages dans la file d'attente	`radio.config(queue=3)`	Au delà (ici 3 messages), ils seront supprimés.
Longueur maximum du message	`radio.config(length=32)`	Celle-ci peut aller jusqu'à 251
Puissance d'émission du signal radio	`radio.config(power=6)`	Valeur comprise entre 0 et 7
Vitesse de transmission	`radio.config(data_rate=radio.RATE_1MBIT)`	Vitesses admissibles : RATE_250KBIT, RATE_1MBIT ou RATE_2MBIT
Fréquence d'émission	`radio.config(channel=1)`	Valeur comprise entre 0 et 83
Réglages par défaut	`radio.reset()`	

7. ENTRÉES-SORTIES

Commandes	Instruction Python	Remarques
Signal binaire (0 ou 1) – Broches : p0 à p11, de p13 à p16, p19 et p20		
Éteindre l'écran	`display.off()`	Quand p3, p4, p6, p7, p9 ou p10 sont utilisées
Lire une broche binaire	`pin0.read_digital()`	Ici la broche est p0
Mettre à 0 une broche	`pin0.write_digital(0)`	
Mettre à 1 une broche	`pin0.write_digital(1)`	
Signal analogique (variable) – Broches : p0, p1, p2, p3, p4, p10		
Lire une broche analogique	`pin0.read_analog()`	Valeur comprise entre 0 (0V) et 1023 (3,3V).
Générer un signal PWM (modulation de largeur d'implusion)	`pin0.write_analog(512)`	Valeur moyenne obtenue par rapport cyclique : entre 0 (0 % - 0V) et 1023 (100 % - 3,3V)