Artistes électroniques

« Que les lumières soient ! »

14 & 15 Février 2023

Document de travail & références de base

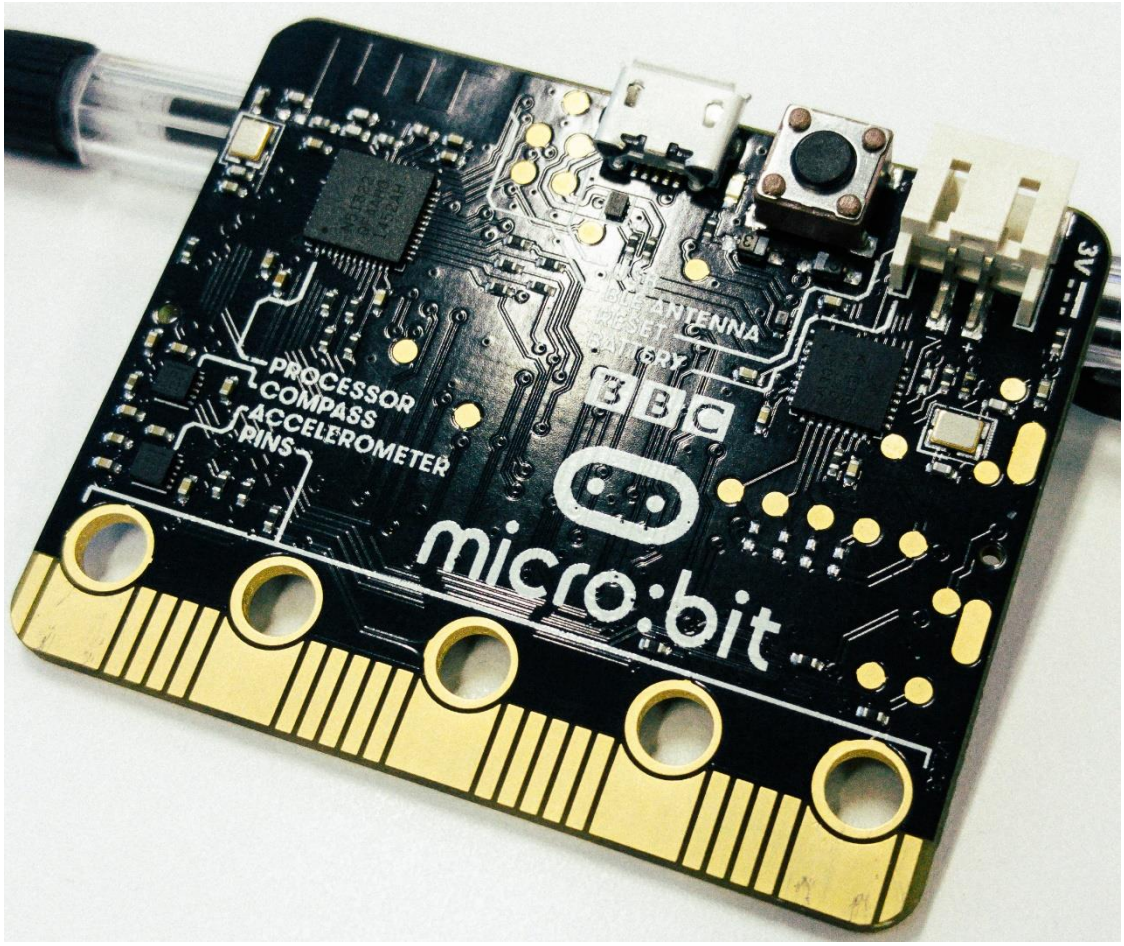
Les stages « artistes électroniques » ont pour but de permettre à des élèves de la cité des arts de s'initier *(superficiellement, il faut le dire)* à l'intégration de l'électronique dans leur pratique artistique au sens large.

Dans ce cadre, « que les lumières soient ! » est un stage qui a pour but d'initier à l'usage du microcontrôleur BBC Micro:bit, afin d'utiliser celui-ci pour contrôler une bande de leds multicolores programmables et de faire réagir celle-ci en fonction des capteurs intégrés au Micro:bit.

Table des matières

Introduction : le BBC micro:bit

Le Micro:bit *(aussi nommé BBC Micro:bit ou micro bit)* est un ordinateur à carte unique doté d'un processeur ARM.
Conçu au Royaume-Uni pour un usage éducatif dans un premier temps, le nano ordinateur est maintenant disponible au grand public dans de nombreux pays.



BBC Micro:bit

La platine de 4 × 5 cm embarque un processeur ARM Cortex-M0, un capteur de mouvement 3D *(ou accéléromètre)* et un magnétomètre 3D *(ou boussole numérique)*, des connectiques Bluetooth et USB, une matrice de 5 x 5 DEL *(25 diodes électroluminescentes)*, un bouton de réinitialisation et deux boutons programmables.

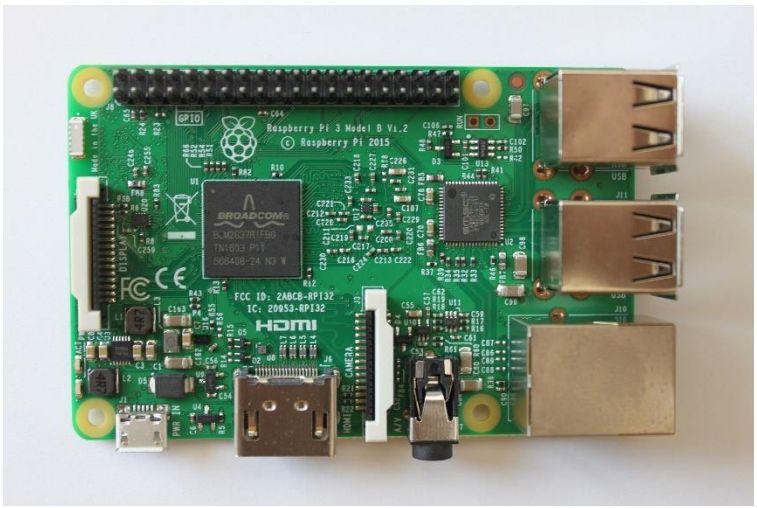
Lancé en juillet 2015 par la BBC, le projet prévoit de distribuer gratuitement un million d'exemplaires à des écoliers britanniques de onze et douze ans pour leur apprendre les fondements de la programmation.
Le but est à la fois de familiariser les enseignants avec ces technologies, mais aussi d'initier les enfants avec des cas simples et concrets.

Selon la BBC, *«Le BBC Micro:bit est un ordinateur de poche que vous pouvez programmer, personnaliser et contrôler afin de rendre concrets vos idées numériques, des jeux et des applications»*.

Le programme se place dans la droite lignée – et tire son nom – du microordinateur Micro développé par la BBC dans les années 1980 pour favoriser l'apprentissage de l'informatique.
Construit par Acorn Computers en partenariat avec la chaine publique anglaise, le BBC Micro connait un grand succès avec 1.5 million d'unités vendues et une percée autant dans les écoles britanniques que les universités.
30 ans plus tard, la BBC relance ce programme.

BBC Micro

Le BBC Micro:bit s'inspire aussi du Raspberry Pi, l'autre grand succès britannique dans le secteur des ordinateurs éducatifs.
Or, le Micro:bit est pensé pour toucher un public plus jeune et moins familier de l'informatique, afin de l'initier dès son plus jeune âge au numérique. Il n'est pas sans rappeler l'Arduino.



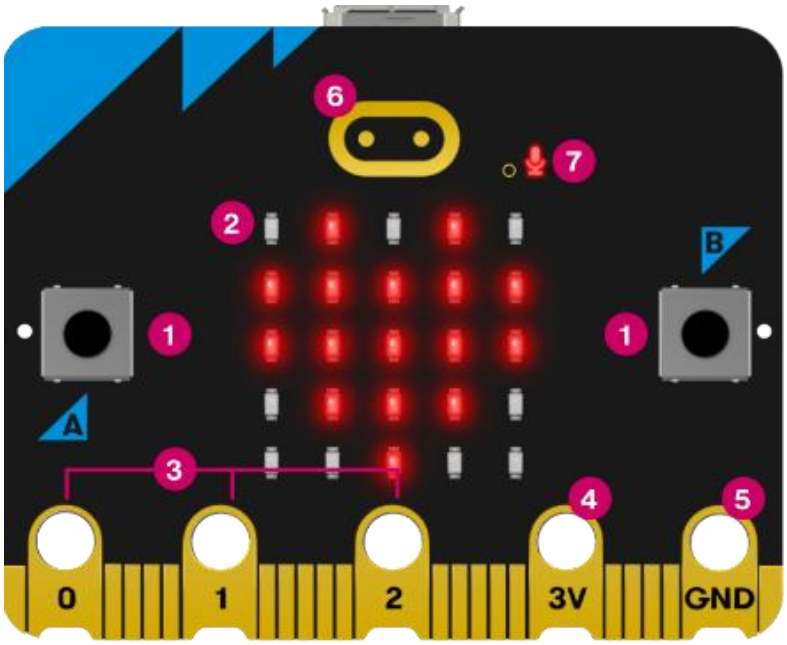
Raspberry Pi B & Arduino Uno

BBC Micro:bit : Tout petit, mais concentré !

A un format souvent nommé « demi carte de crédit », le microcontrôleur Micro:bit est rempli de fonctionnalités, comportant un nombre importants de capteurs directement sur sa carte là ou des microcontrôleurs comme les Arduino ont souvent besoin de « shields » ou de capteurs externes pour être capable des mêmes fonctions.

Le modèle utilisé est la version 2 du Micro:bit, avec quelques fonctionnalités supplémentaires par rapport à la V1.

Face avant



1) 2 boutons pression *(modèles très simples & courants)*. Matrice de leds 5*5 et capteur de lumière *(25 leds pour afficher des lettres, des chiffres, des images… Elles peuvent aussi servir de capteur de lumière)*

2) Broches GPIO *(General Purpose Input Output, « Entrées Sorties à Usage Général »)*.
Elles permettent de connecter des écouteurs, des capteurs de toucher, et d'autres appareils électroniques pour augmenter les capacités de votre micro:bit

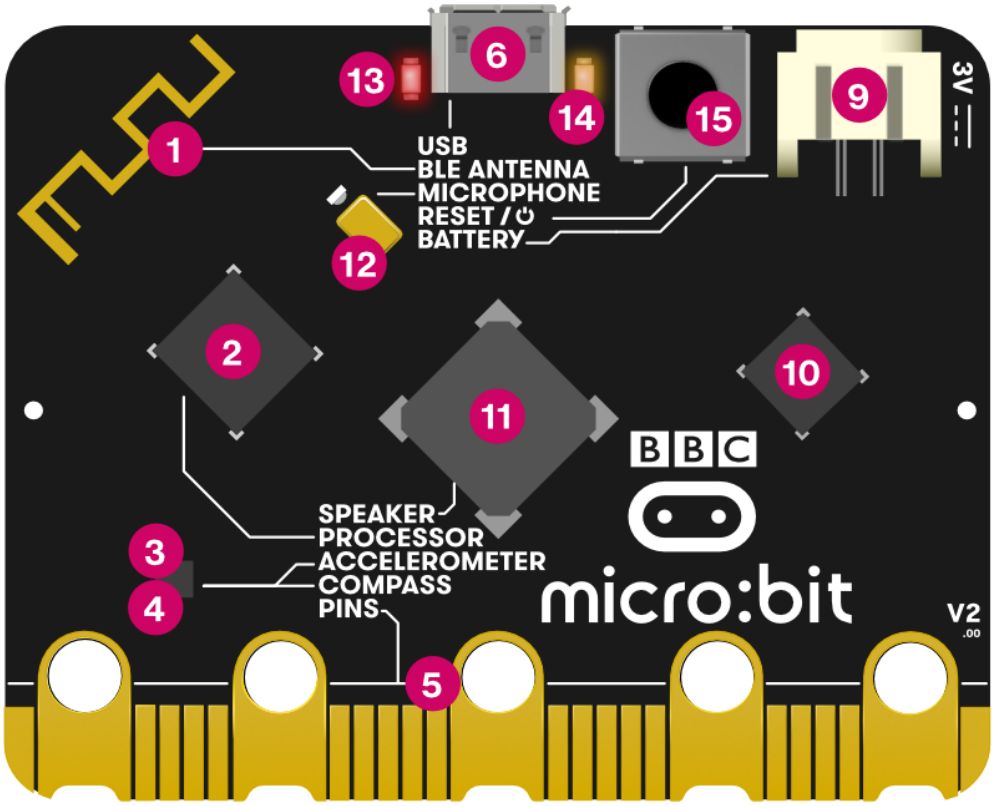
3) Broche 3 Volts, pour alimenter des leds, des capteurs…

4) Broche de masse, pour compléter les circuits électroniques crées avec des éléments exterieurs.

5) Logo Tactile. Une nouveauté de la V2, il s'agit d'un bouton tactile déguisé en Logo Micro:bit.

6) Led du microphone. Indique si le microphone est en fonction *(nouveauté de la V2)*.

Face arrière



1) Antenne Radio & Bluetooth. Le Micro:bit est capable de communiquer avec d'autres micro :bit par ondes radio OU avec d'autres appareils par Bluetooth.

2) Processeur & capteur de température. Le processeur est l'élément central du Micro:bit, celui qui exécute vos programmes. Il est aussi équipé d'un capteur de température.

3) Magnétomètre. Un capteur de champs magnétiques, il peut servir à créer une boussole ou à détecter la présence de camps magnétiques dans diverses applications.

4) Accéléromètre. Un détecteur d'accélération dans les 3 dimensions *(X, Y & Z)*, capable aussi de détecter la gravité et l'orientation physique de la carte.

5) Broches. Le dos des broches vues sur la face avant !

6) Prise Micro USB. Utilisée pour charger les programmes et alimenter le Micro:bit si il n'est pas alimenté par le connecteur d'alimentation.

9) Prise de batterie. Format JST, permet de connecter le Micro:bit sur 2 piles AAA

10) Puce interface USB. Effectue la liaison USB avec votre ordinateur.

11) Haut-parleur. Petit haut-parleur intégré *(nouveauté de la V2)*

12) Microphone *(nouveauté de la V2)*.

13) Led d'alimentation *(rouge)*. Indique la présence d'alimentation électrique.

14) Led *(jaune)*. Indique si un programme est en téléversement *(clignotement)*, ou si la carte est alimentée par l'USB *(lumière fixe)*.

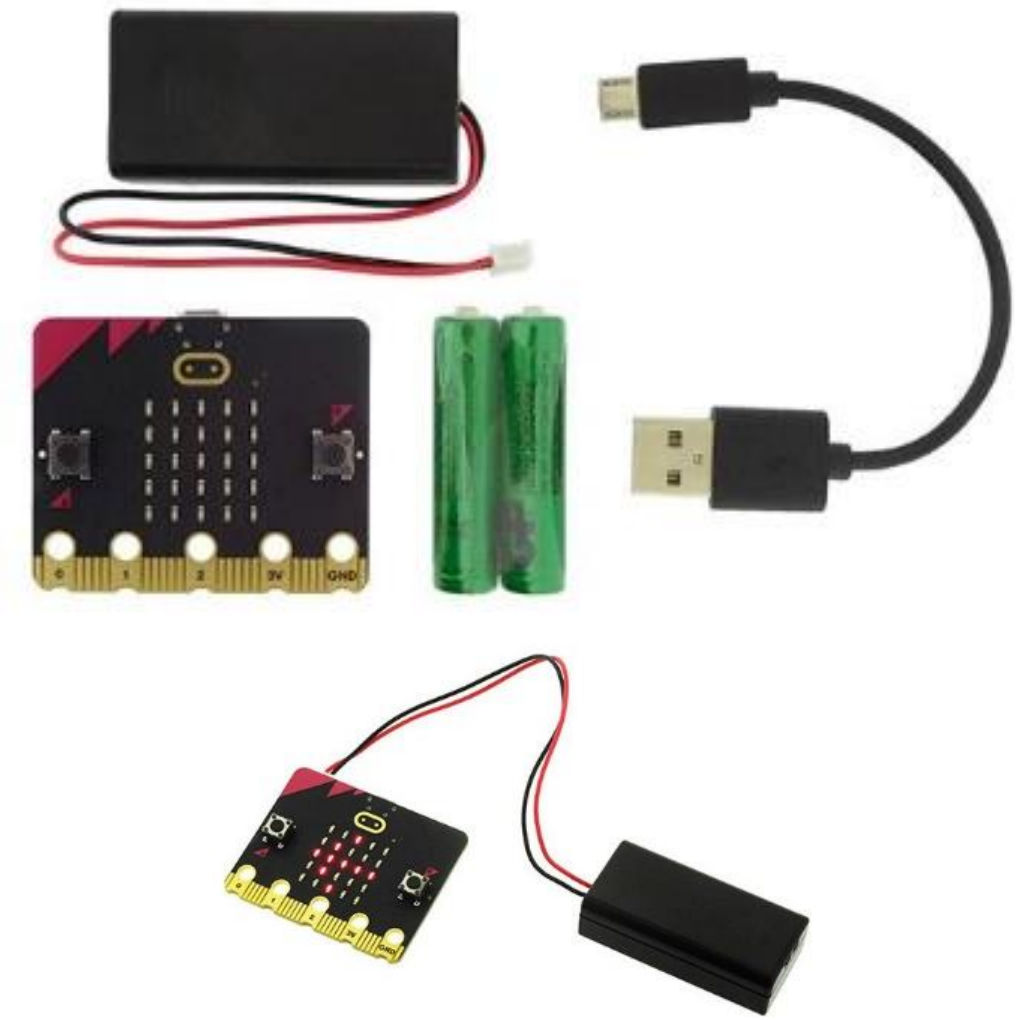
15) Bouton de réinitialisation. Sert à redémarrer le programme à bord de la carte. Sert aussi à arrêter la carte V2 si on garde le bouton appuyé 3 secondes.

Pour bien commencer : « Qu'est ce qui y'a dans la boite ?? »

Avant de regarder plus avant le reste du stage, arrêtons-nous un moment sur les éléments présents dans chaque kit de base nommé « BBC Micro:bit GO », qui comporte les éléments de base pour bien commencer.

Le contenu de chaque kit est le suivant :

- 1 carte micro:bit V2.2
- 1 câble micro-USB d'environ 15 cm *(pour relier à un ordinateur)*
- 1 coupleur 2 piles AAA avec cordon JST
- 2 piles AAA 1,5 Vcc
- Indications de démarrage *(en anglais)*



Le montage est assez simple…

Comment programmer le BBC Micro:bit

Le Micro:bit peut être programmé de beaucoup de manières différentes, les principaux moyens étant les suivants :

Le site Microsoft Makecode (https://makecode.microbit.org)

Très facile d'accès et d'utilisation, ce site utilise la méthode des « blocs » visuels pour permettre de coder sans avoir à connaître les bases d'un logiciel ou d'un langage de programmation.
Initialement prévu pour les enfants, il est accessible à tous et à l'avantage de permettre de conserver les codes qui sont créés par un utilisateur sur sa machine *(tant qu'on ne vide pas les données de navigation de son navigateur Web favori)*.
Il est possible à tout moment de basculer entre l'interface de codage par blocs et une interface de codage plus classique utilisant le langage JavaScript.
(Il sera notre interface de travail pour ce stage)

L'éditeur Python microbit (https://python.microbit.org/v/3)

Il permet de coder à l'aide du langage Python, un langage orienté objet très puissant et commun.
Il existe aussi divers logiciels de codage Python hors ligne qui peuvent être utilisés pour coder sur Micro :bit.

Applications mobiles Makecode (https://microbit.org/get-started/user-guide/mobile)

Makecode existe aussi sur IOS & Android, vous pouvez donc utiliser le même système de codage par bloc sur votre tablette ou votre téléphone.
L'avantage premier de ces logiciels est qu'il permettent de téléverser le code directement par Bluetooth dans votre Micro :bit, sans avoir besoin de le relier par USB *(mais une alimentation électrique est tout de même nécessaire… :P)*.

Scratch (https://scratch.mit.edu/microbit)

Scratch est l'un des premiers site & logiciel de codage par blocs conçu au départ par le MIT pour permettre l'apprentissage du codage par les enfants, il est actuellement très utilisé un peu partout dans le monde pour permettre une première approche du code, en particulier en 6ème et 5ème au collège.
Il est possible d'utiliser Scratch pour programmer le Micro:bit si on le désire *(voir le lien)*.

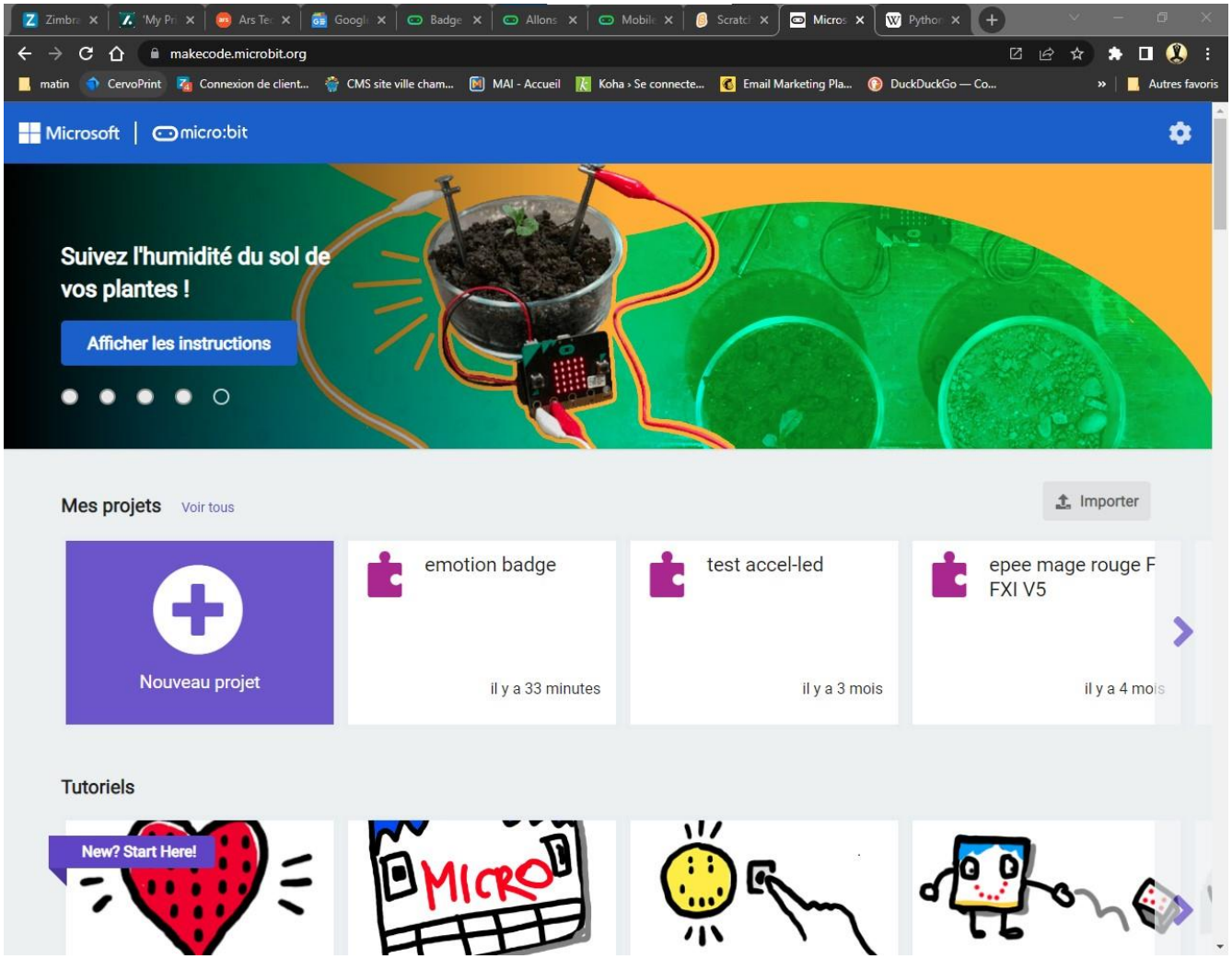
Autres ! (https://microbit.org/fr/code/#other-editors)

Il existe bien d'autres moyens de coder sur Micro:bit : EduBlock, Mu Python, l'IDE Arduino…
Si on possède déjà une certaine maitrise d'un programme, d'un langage ou d'une IDE particulière il est tout à fait possible que ceux-ci soient une possibilité supplémentaire pour coder *(la liste est disponible sur le site web indiqué)*.

Comme indiqué, nous utiliserons Microsoft Makecode pour ce stage, étant donné qu'il est facile d'accès *(un navigateur internet à jour et une connexion internet sont les seuls éléments nécessaires)*, et facile d'usage *(interface par bloc, simulateur de carte…)*

Le site Microsoft MakeCode

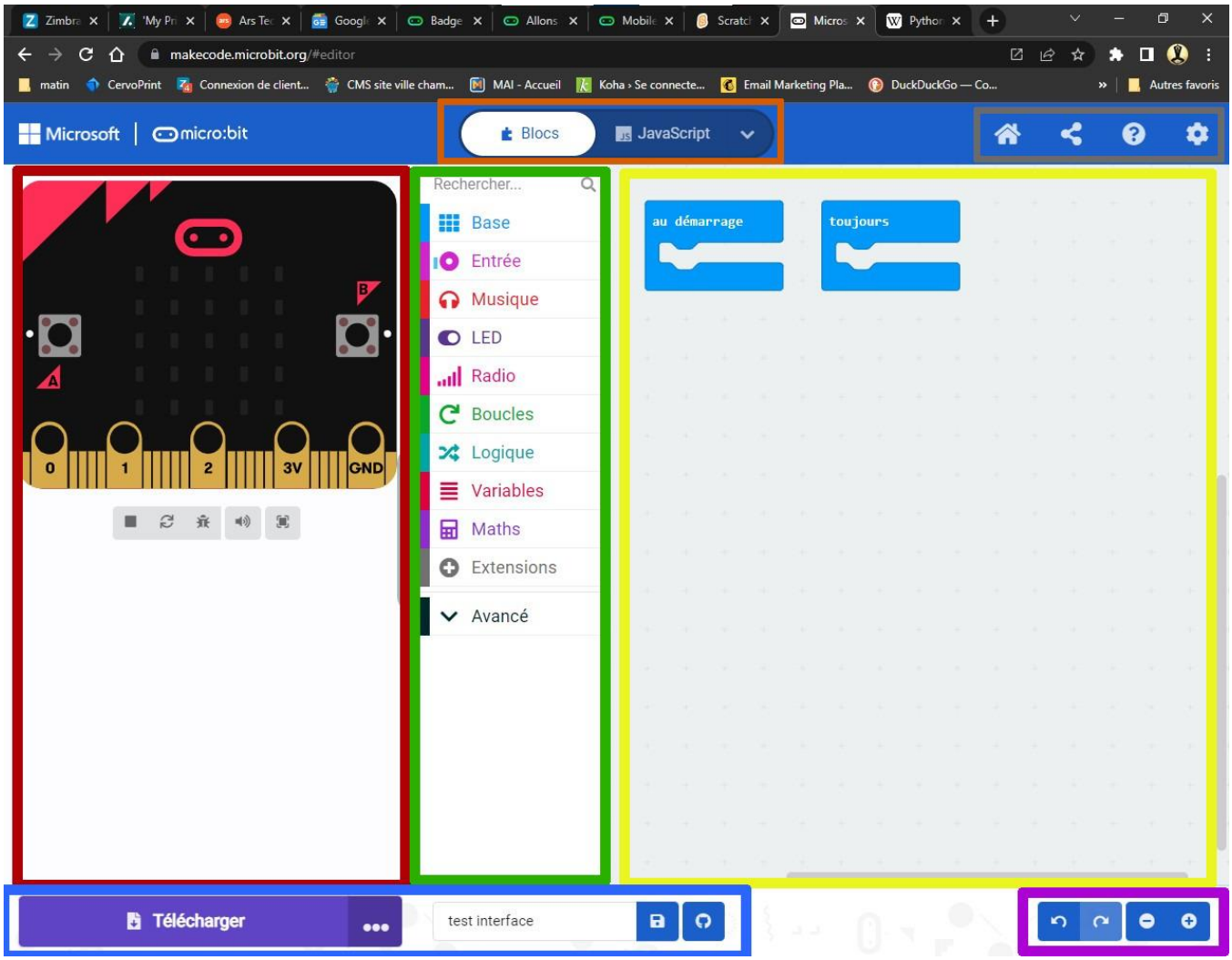
Pour commencer, rendons nous sur le site web **https://makecode.microbit.org** **!**



La page d'accueil est très simple, elle liste l'ensemble des projets de l'utilisateur sous « mes projets », et toute une liste de tutoriels dans la rubrique du même nom.

Il est possible d'importer des fichiers « .hex » qui sont le format des programmes conçus pour le Micro:bit à l'aide du bouton « Importer ».
il est possible une fois un projet crée de l'exporter au même format, ce qui permet de sauvegarder & archiver les programmes de manière plus pérenne *(les projets disparaissant si les données de navigation sont effacées)*.

La page d'un Projet *(qu'il soit créé ex-nihilo ou importé)* se présente sous la forme suivante :

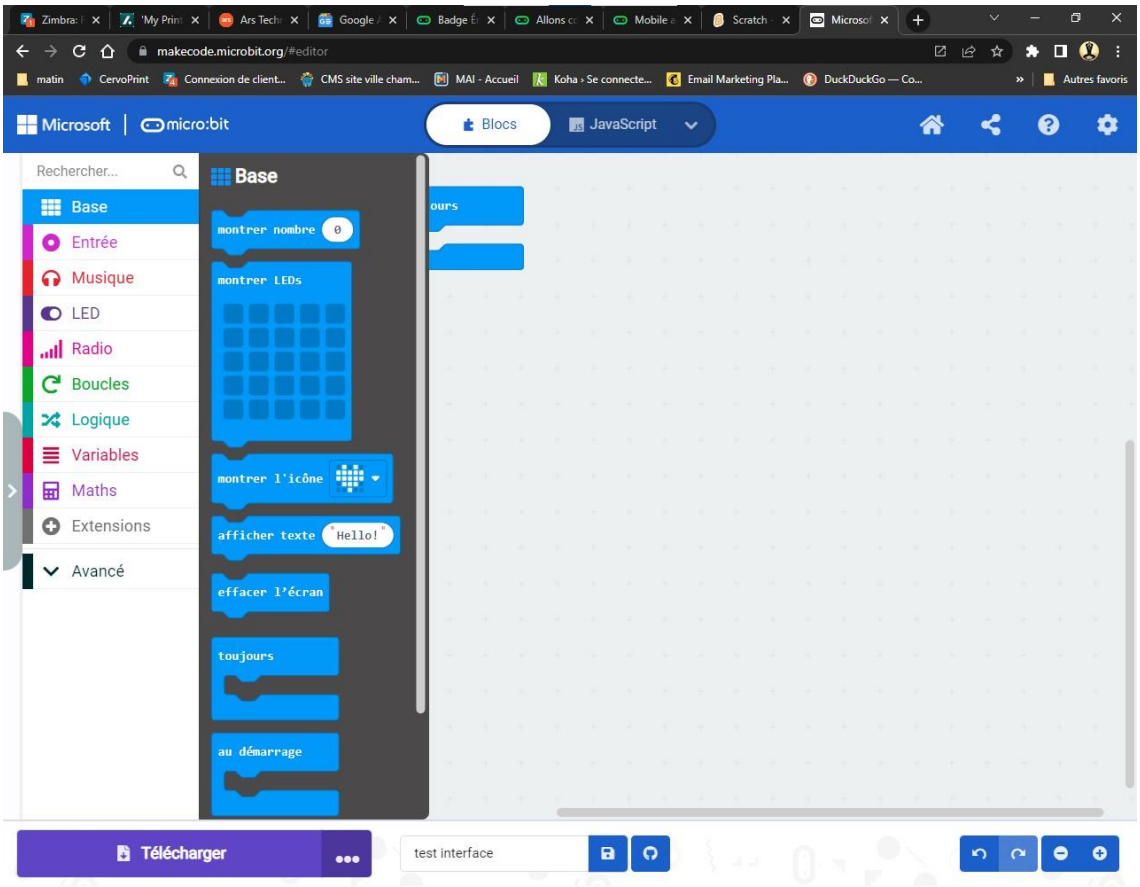


- En Orange : le sélecteur permettant de passer du mode de programmation par blocs à la programmation JavaScript et inversement *(il peut parfois s'avérer que repasser de JavaScript au mode blocs soit limité selon les codes utilisés)*.
- En Gris : les boutons de gestion du projet *(Accueil, partager le projet de manière publique, aide, divers paramètres…)*
- En Rouge : Un simulateur de Micro:bit, ou il est possible de tester directement le code sans avoir à le téléverser encore et encore dans le vrai microcontrôleur *(il est possible de cacher cette zone pour gagner de l'espace de travail)*
- En Vert : les Blocs de programmation disponibles, rangés par catégories
- En Jaune : l'espace de travail, ou on peut disposer et assembler les blocs à volonté
- En Bleu : pour télécharger le projet en cours sur votre ordinateur *(au format .hex)* ou directement dans le Micro:bit si votre matériel le permet.
- En Violet : boutons Annuler/Refaire et boutons de Zoom

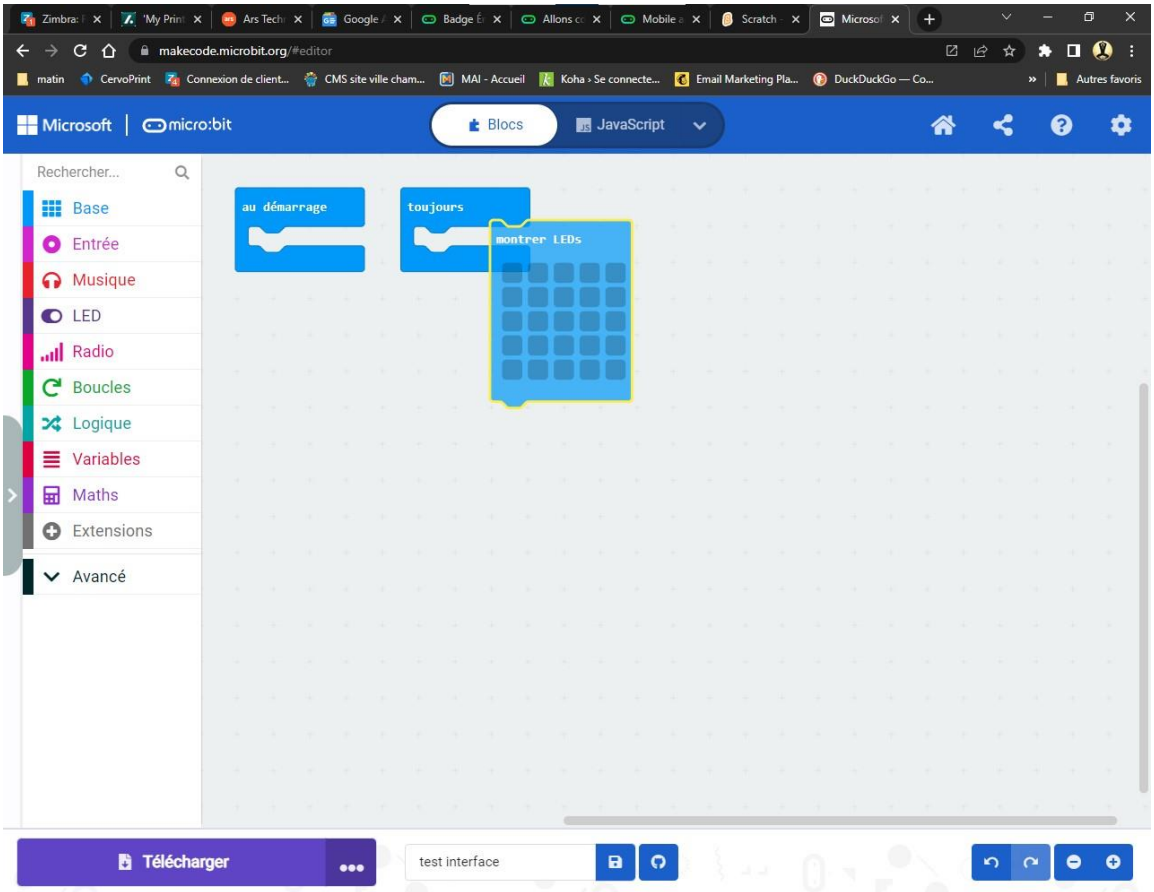
Pour gagner un maximum de place pour travailler il est possible de cacher le simulateur de Micro:bit, mais aussi de passer le navigateur en mode plein écran au besoin.

Au démarrage d'un nouveau projet, les blocs « Au Démarrage » et « Toujours » sont présents par défaut, ils représentent les 2 sections de code de base les plus importantes pour faire fonctionner le microcontrôleur : le Setup *(Démarrage)* et la Loop *(Boucle)*.

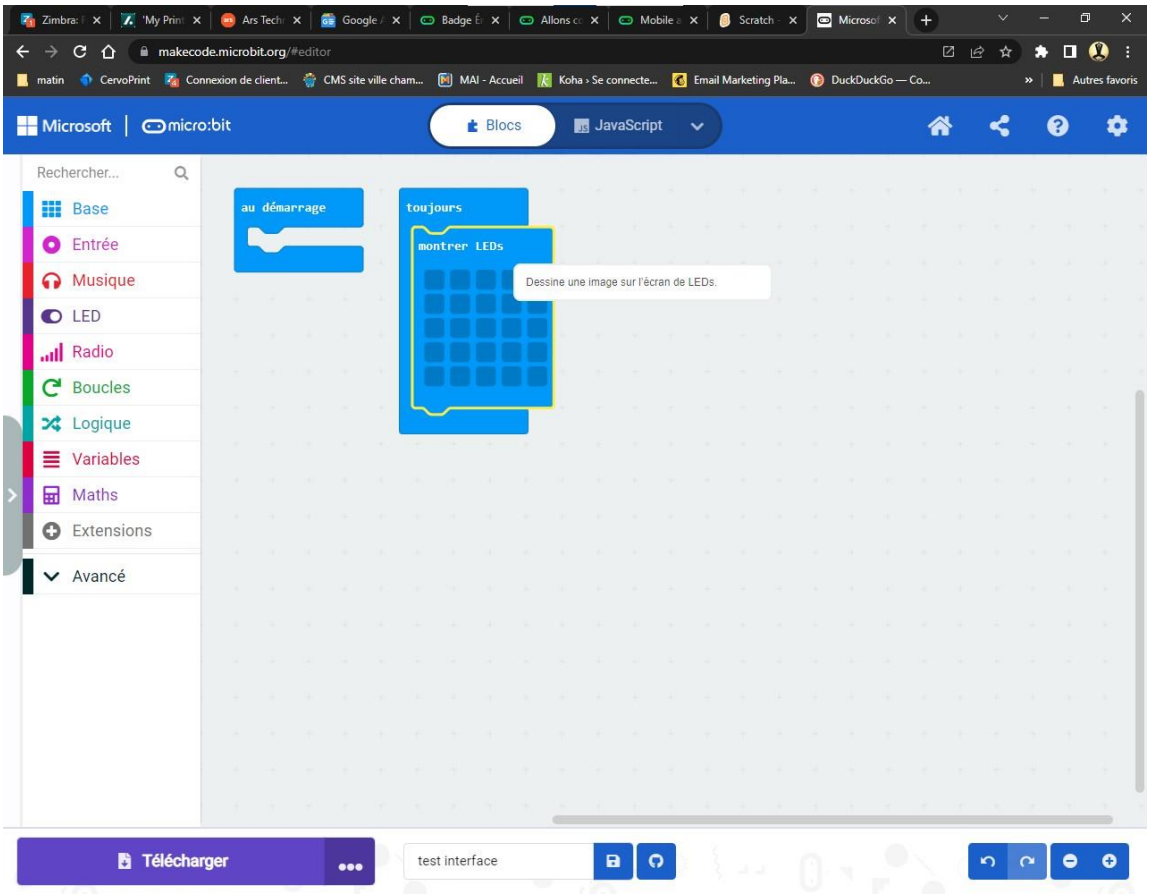
Pour commencer à coder, il suffit d'ouvrir une des rubriques de blocs à gauche



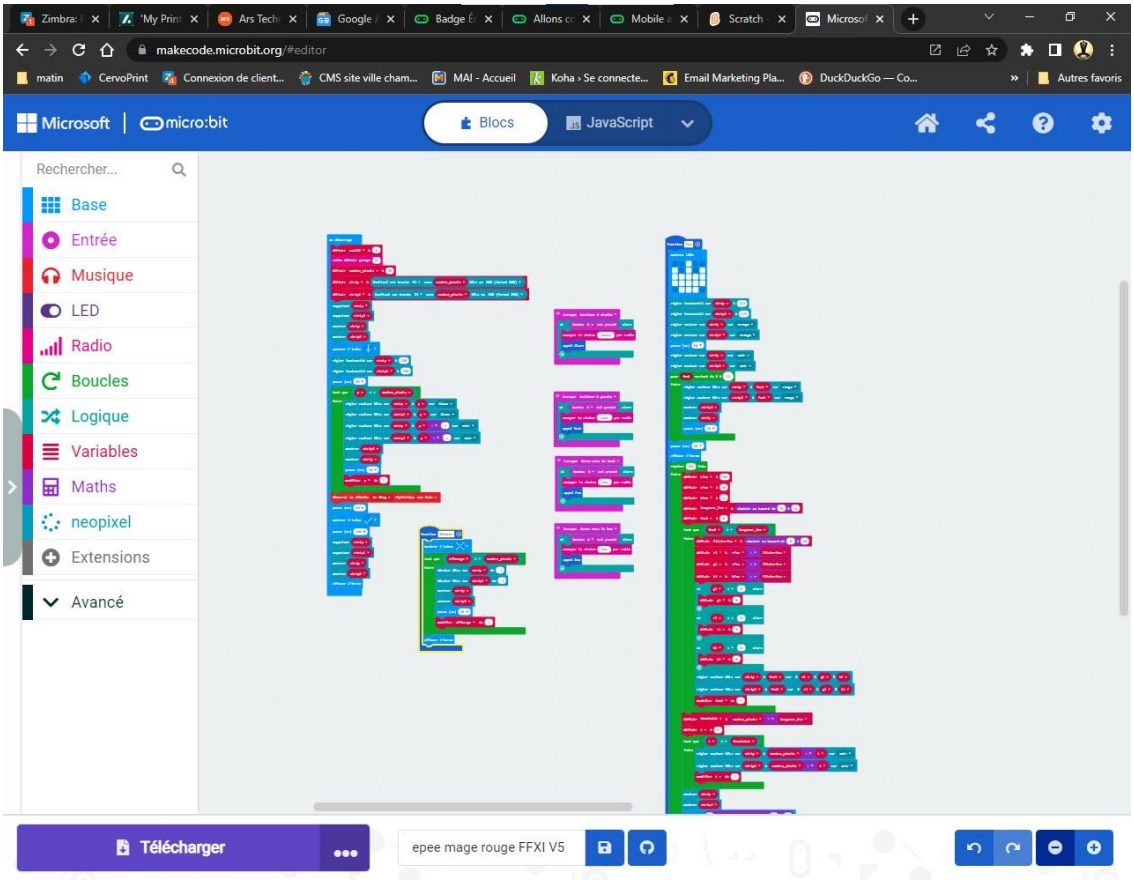
D'attraper le Bloc qui vous intéresse avec un clic de souris



Et de l'insérer dans le bloc de départ qui vous intéresse, comme un puzzle !



Les blocs ont tendance à se magnétiser les uns aux autres, ce qui peut poser problème quand on essaye de créer des codes très complexes, mais une fois maitrisé cette méthode permet de fabriquer des codes assez avancés si on le désire



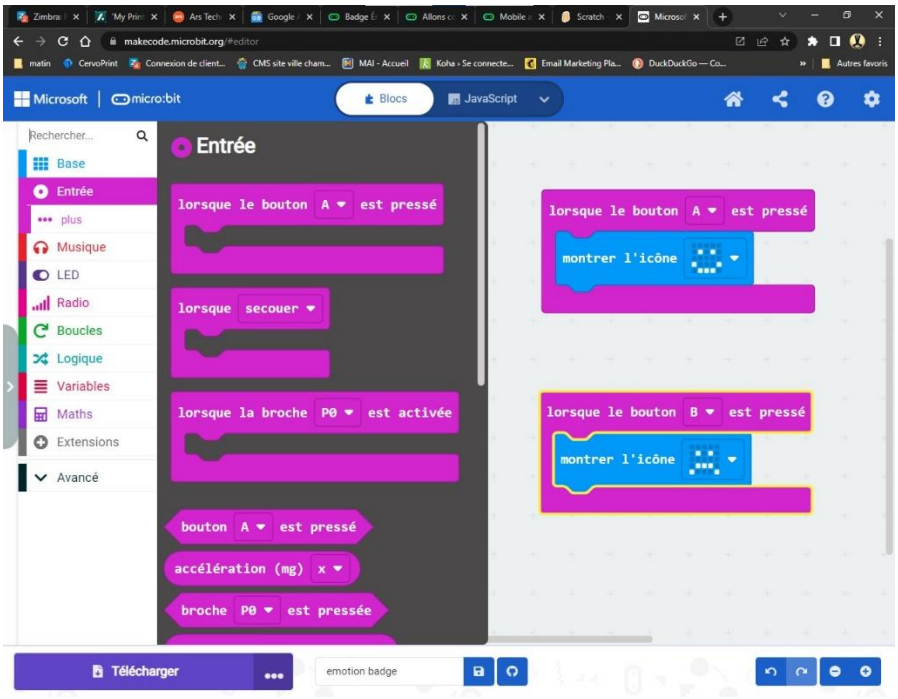
Premier Projet rapide : Le « Badge Emotions »

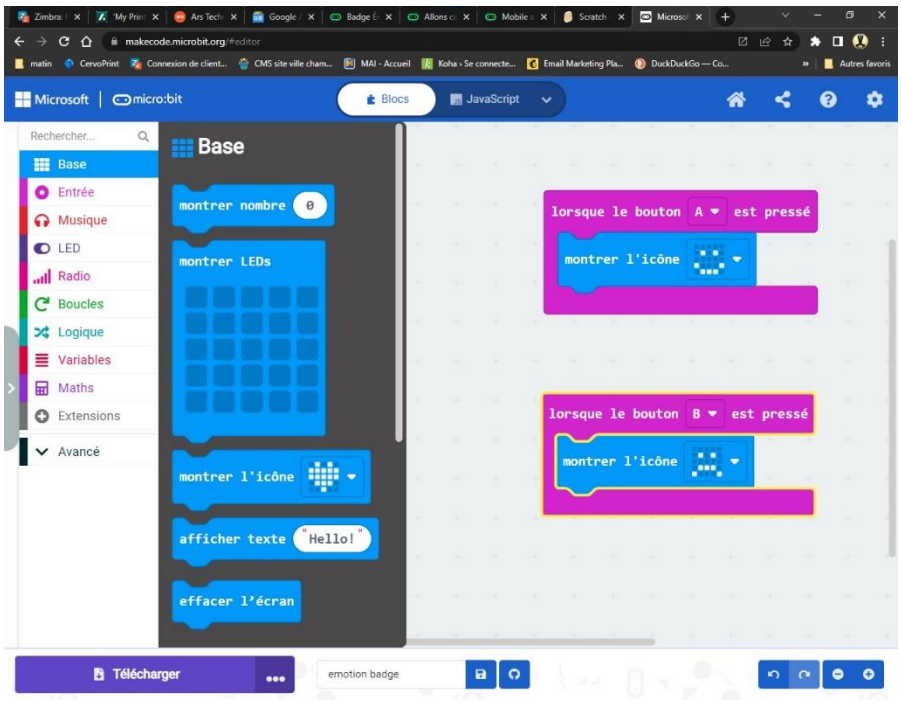
Nous allons commencer la manipulation du Micro:bit par un des projets de base proposé par le site :

Le Badge Emotion (https://microbit.org/fr/projects/make-it-code-it/emotion-badge)

Il s'agit de simplement utiliser les boutons A et B sur la première face pour faire apparaitre soit un sourire, soit un visage renfrogné.

Le code est extrêmement simple, il suffit de par 2 fois utiliser l'entrée « Lorsque le bouton [ ] est pressé », de choisir « A » pour l'un et « B » pour l'autre, puis de rajouter un bloc de base « Montrer l'icône » différent pour chacun, Et voilà !





Il suffit ensuite de récupérer le fichier .hex final en cliquant sur « Télécharger », et l'envoyer sur le Micro:bit branché en USB sur l'ordinateur.

Ensuite, on peut essayer notre programme une fois le transfert terminé, voir le bricoler un peu !
(Essayez de modifier le bloc « entrée » pour voir !)

Que les lumières soient : un petit explicatif du projet

Le projet qui va nous occuper va demander un poil plus de travail (cela dit, nous avons un fichier .hex final sur lequel nous reposer au besoin).

Il s'agit ici de relier notre micro:bit à une bande de leds programmables, et de faire varier celle-ci en couleurs (ou autre) selon les entrées (mouvements, rotation, accélération, orientation, bruit…) disponibles du microcontrôleur.

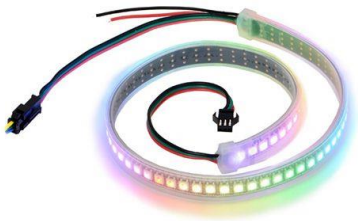
Pour y arriver, nous pourrions parfaitement relier la bande de leds au microbit par 3 pinces crocodiles, sur les sorties P0, 3V et Ground, et lâcher l'affaire, mais cela serait un peu trop basique et assez peu résistant aux divers mouvements que notre nano ordinateur pourrait subir.

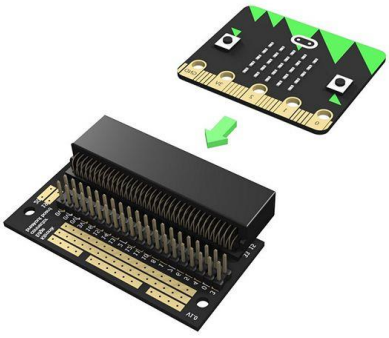
C'est pourquoi nous allons utiliser une interface de branchement par broches, et souder quelques fils Dupont sur notre bande de leds, histoire de rendre le tout à la fois plus solidement relié et plus facile à bricoler dans le futur.

Voici donc nos éléments nécessaires pour la fabrication :

← Fils Dupont (3)

Bande de leds rgb addressables (1m) →





Kit Micro:bit Go →

← Connecteur Edge

Le montage est assez simple :

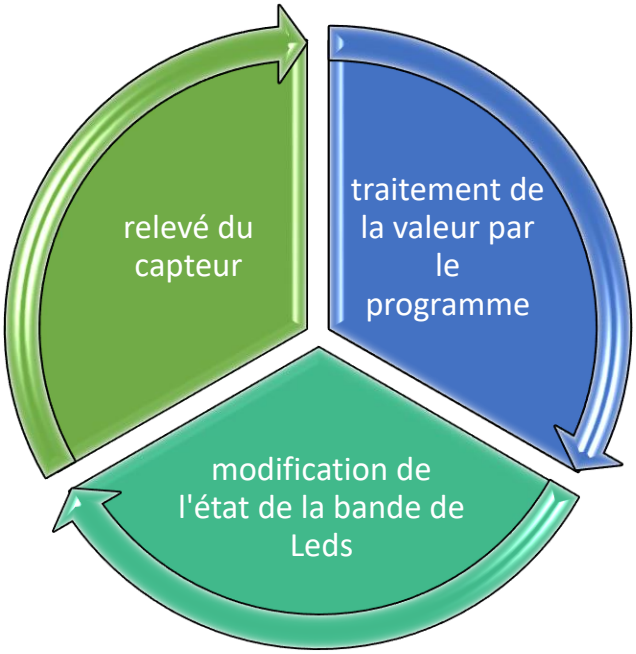
- Souder 3 fils Dupont femelle sur les 3 pastilles Vin (rouge) – Data(au choix) – Grd(noir) de la bande de leds
- Brancher les 3 fils sur les pis P0 3V et GRD du connecteur (Vin ←→3V, Grd←→GRD & Data←→P0)
- Brancher le Micro:bit dans le connecteur Edge

(Donc pas de l'ingénierie spatiale, mais 3 soudures un peu délicates il est vrai)

Que les Lumières soient : le concept de la programmation

L'idée de base est la suivante : on se sert d'un des capteurs du micro:bit *(bouton, orientation, vitesse, boussole…)* pour modifier les couleurs / intensité / allumage des leds de la bande programmable.

Le capteur est donc l'entrée de notre programme, l'état des leds est la sortie du programme.



Le programme va :

1. Relever la valeur du capteur choisi
2. Traiter celui-ci pour le traduire en tant qu'éléments à modifier sur la bande de leds
3. Envoyer ces modifications à la bande de Leds
4. Recommencer au début de la boucle ! 

Le résultat, c'est que le capteur du micro:bit va en somme fonctionner comme un potentiomètre de réglage, qui va faire défiler les lumières d'avant en arrière selon comment on agit sur le microcontrôleur.

Pour ce faire, il nous faut utiliser les blocs « Au Démarrage » *(Setup)* et « Toujours » *(Loop)*

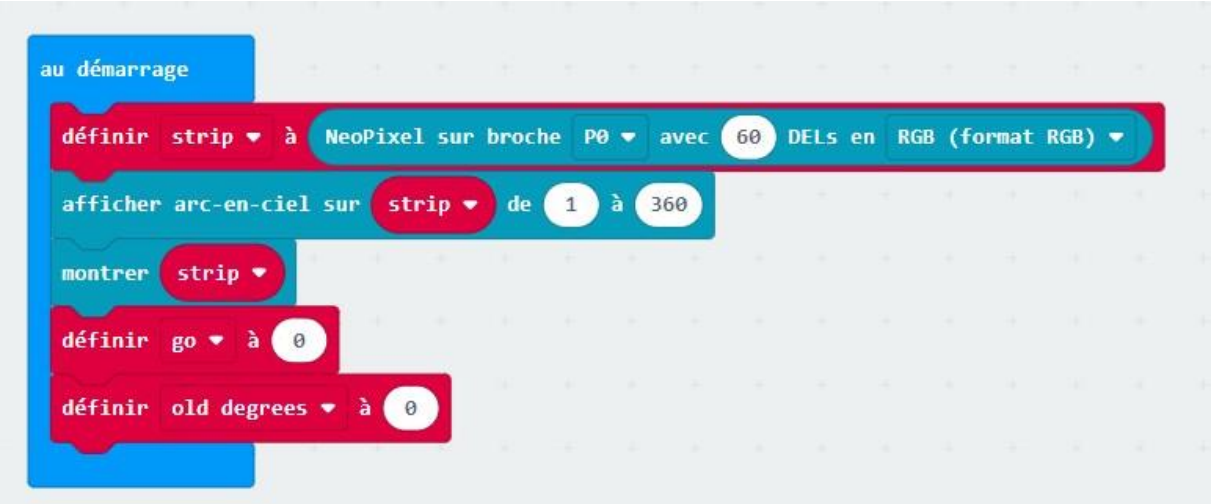
(Importons le fichier « microbit-Lumières-soient.hex » dans Makecode)

Décorticage du programme

Bloc « Au Démarrage » :

Il sert à déclarer l'ensemble des éléments que nous allons utiliser, il s'agit de la première partie du programme qui est exécuté 1 seule fois et avant tout le reste :

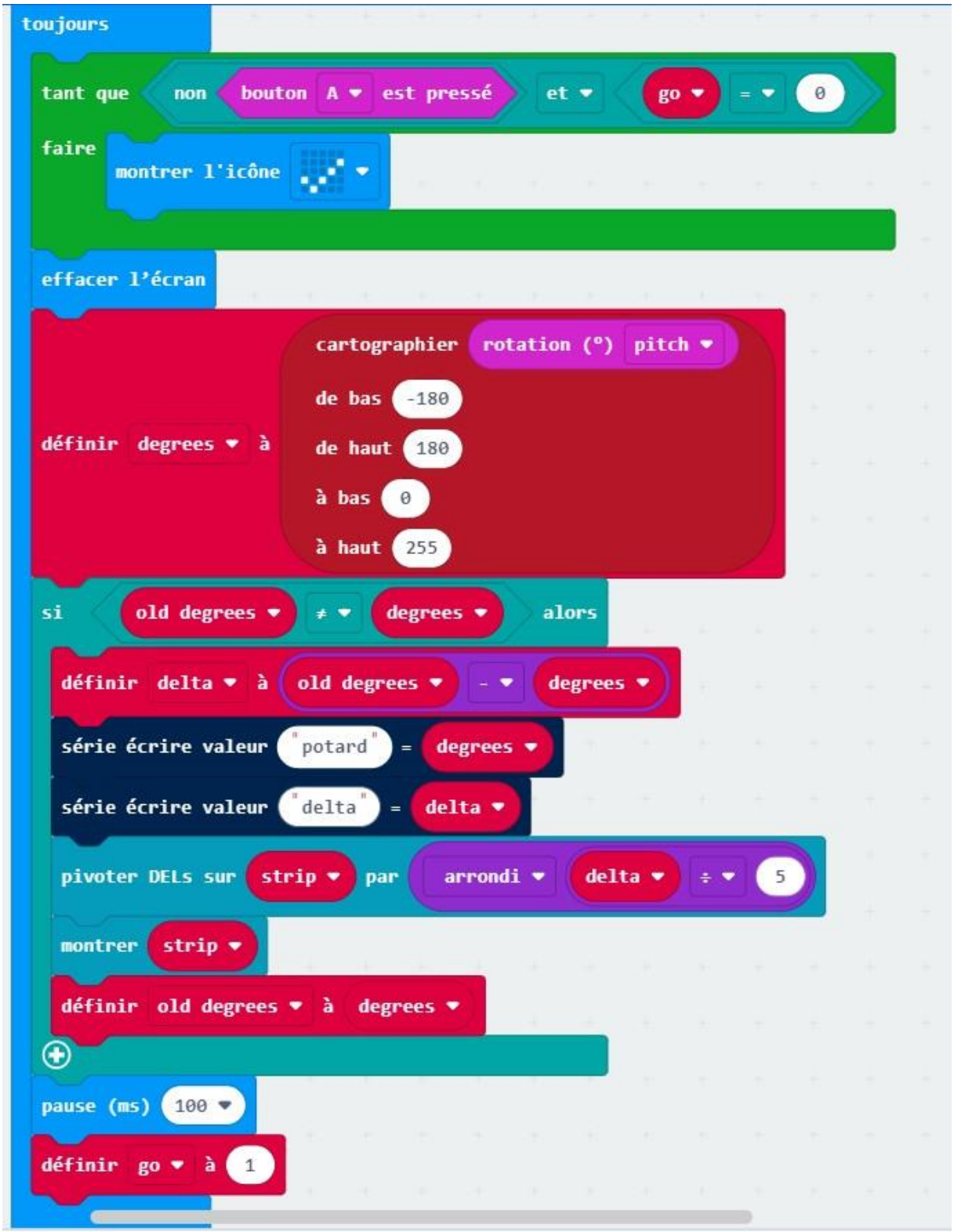
- Nous définissons notre bande de leds *(sa nature, sa longueur en nombre de leds, quel pin est utilisé pour la transmission de données) (nommée « strip »)*
- Pour le fun, on affiche un arc en ciel sur « strip » ! il sera notre élément qui changera visuellement.
- On « montre » strip, c'est-à-dire qu'on envoie l'ordre à la bande de leds de s'allumer comme on le veut.
- On crée une variable relative à l'état de fonctionnement du programme *(une détente en somme, pour éviter de le démarrer de suite comme nous allons le voir) (nommée « go »)*
- Et une variable relative à notre capteur *(« old degrees »)*, qui nous sera utile ensuite.



Il est à noter que les bandes de leds Neopixels ne sont pas présentes dans la liste des éléments de base quand on crée un nouveau code sur Makecode, il faut rajouter cette bibliothèque d'éléments en appuyant sur « Extensions », et trouver la bonne bibliothèque nommée « Neopixel »

Les diverses bibliothèques proposées permettent d'augmenter les capacités de notre microcontrôleur, pour lui permettre de manipuler des bandes de leds, des servomoteurs, des capteurs à ultrasons… De quoi très fortement améliorer notre micro:bit déjà bien paqué !

Bloc « Toujours » :



Décortiquons un peu ce code, il est assez direct.

- **Partie en vert « Tant que » :**
  Il s'agit en vérité d'une petite boucle, qui peut se lire de la manière suivante :
  « Tant que le Bouton A n'est pas pressé ET que la variable Go est égale à zéro, affichez sur l'écran l'icône OK »
  Si une des conditions devient fausse *(si A est pressé ou si Go n'est plus égale à zéro)*, les conditions de la boucle ne sont plus bonnes et on sort de la boucle pour continuer le programme.

- **« Effacer l'écran »**
  est assez explicite : on efface l'écran du micro:bit *(une bonne indication qu'on est sorti de la boucle)*

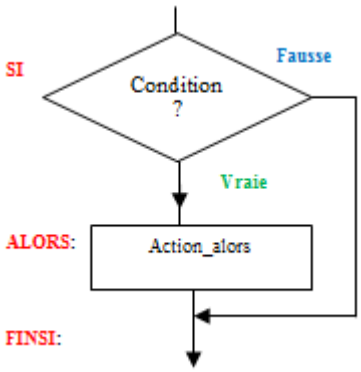
- **Le gros bloc en rouge « Définir degrees à »**
  est un poil plus complexe, il s'agit là de la partie du code ou on va récupérer la valeur actuelle du capteur qui nous intéresse, et effectuer un ensemble d'opérations dessus afin d'obtenir une variable *(une valeur en somme)* nommée « degrees ».

  la donnée brute choisie ici est le pitch en degrées, c'est à dire l'angle de tangage ou assiette du Micro:bit *(son inclinaison avant-arrière par rapport à l'horizontale, de -180° à 180°)*.

  On ne peut pas hélas passer directement cette donnée au reste du programme, car celui-ci fonctionne comme un potentiomètre qui va de 0 à 255 *(valeur maximale à ce format)*, alors que le capteur va de -180 à 180.
  Il faut donc la convertir, ou ici la « cartographier », c'est-à-dire faire correspondre les valeurs du capteur à celles de la variable :
  → -180° du capteur correspond à 0 pour la variable « degrees »
  → 180° du capteur correspond à 255 pour la variable « degrees »
  Et la fonction du bloc est de se charger de faire correspondre les valeurs entre elles entre ces extrêmes qu'on a définis !

- **La partie en bleu vert « Si – Alors »**
  est une part du code qui s'appelle **une Expression Conditionnelle**, elle permet d'effectuer tel ou tel traitement en fonction de la valeur d'une condition.

  Très simplement, il faut que ce qui est juste derrière **« Si »** *(la Condition)* soit vraie pour que le code **« Alors »** en dessous puisse être exécuté.
  Ici, il faut que les variables « old degrees » et « degrees » soient différentes, sinon on saute tout le bloc !



  Si elles sont bien différentes, Alors nous exécutons le code :
  - On crée une variable nommée « delta », qui est égale à « old degrees » - « degrees »
    (notez qu'elle peut donc être négative ou positive, selon si on penche vers l'avant ou l'arrière)
    « delta » est utile pour déterminer la différence de position entre nos deux données *(ce qu'on appelle justement le Delta en physique)*, et savoir si on doit faire varier les lumières de la bande « strip » vers l'avant ou l'arrière *(+ ou -)* et de combien de leds *(sa valeur absolue)*.

  - Les 2 « série écrire valeur » sont juste des lignes permettant de créer une trace (un log) des variables « degrees » et « delta », histoire de vérifier…

  - « pivoter dels sur « strip » » est la fonction qui va modifier les lumières sur la bande de leds (on y est enfin ! ^^)
    on se sert donc de la valeur de « delta » pour avancer ou reculer les lumières d'autant sur l'ensemble de la bande led.
    La raison pour laquelle la valeur de « delta » est auparavant divisée par 5 et arrondie est pour permettre d'éviter que le changement soit trop anarchique, en somme on

lisse un peu le résultat (mais ces éléments sont totalement dépendent de ce qu'on désire faire !)

- « montrer « strip » » : il ne faut pas oublier d'envoyer l'ordre à la bande de leds d'afficher les modifications, ou sinon ça n'apparait pas !

- « Définir « old degree » à « degrees » : on prend la valeur de « degrees » qui était relevée auparavant, et on l'utilise pour remplacer celle de « old degrees ».

  L'idée ici est de s'assurer que à la prochaine exécution de la boucle *(qui est infinie si vous vous rappelez bien)*, on ait la valeur précédente de « degrees » pour la comparer à sa nouvelle valeur.
  de cette manière, on peut modifier les positions des lumières sur la bande led en incrémentation (petit à petit en gros), et ça aide à rendre la transition plus souple.

- Et enfin, **nous sortons de la boucle SI- ALORS !**

- **Le bloc bleu « pause ms '100' »** :
  on fait une petite pause ! cela permet de temporiser un peu, la bande de leds mettant un certain temps pour terminer de modifier la position des lumières.

- **Le bloc rouge « définir « go » à 1 »** :
  vous rappelez vous notre petite boucle « Tant que » du début ?

  Notre code ne peut tourner à l'infini que si le bouton A est appuyé, ou « go » est différent de 0, sinon il affiche juste le symbole pour OK sur l'écran, n'est-ce pas ?

  Eh bien, pour éviter d'avoir à appuyer sur le bouton A constamment, on passe la variable « go » à 1, et c'est tout comme !
  Il existe des solutions qui peuvent permettre de faire du bouton A un bouton Marche / Arrêt sinon, en faisant de lui le capteur qui modifie la valeur de « go » directement, mais…
  C'est à vous d'essayer maintenant ! ;)

Comment adapter ce code facilement ?

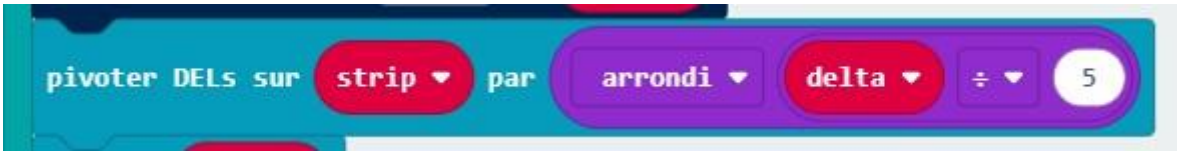
Il existe bien des manières d'adapter ce code selon ses besoins, et bien qu'il ne soit pas très optimisé *(soyons honnêtes)*, il existe 2 éléments en particuliers intéressants à modifier pour expérimenter un petit peu…

La définition de la variable « degrees » (bloc « définir « degrees » à »)

Ce bloc contrôle ce **QU'EST** « degrees » en fait, donc au lieu de relier cette variable à la rotation en tangage du micro:bit, pourquoi ne pas essayer en roulis ?
Ou une accélération, un niveau sonore, l'orientation magnétique ?

Les éléments à bords arrondis de couleur violette présents dans le menu « Entrées » sont tous utilisables dans cette partie du code, il faut alors retirer « rotation (°)pitch » et le remplacer par un autre, tout en vérifiant les maximales et minimales à mettre à la place de « -180 » et « 180 » au besoin.

Le « lissage » de la variable « delta » (bloc « pivoter dels sur »)



On peut parfaitement décider de ne pas arrondir « delta », ou le diviser par autre chose, ou le multiplier ??

A vous d'expérimenter la aussi !

En conclusion

Ce code est extrêmement basique, il est possible de l'améliorer et de le rendre plus efficace, mais ne vous formalisez pas non plus là-dessus, pour la plupart des usages il peut être tout à fait utilisable.

Je tiens à préciser tout de même 2 petites choses :

- La première c'est que sur ce stage la longueur de la bande de leds est limitée à 1 mètre environ, et ce n'est pas une raison d'économie…

En effet, les bandes Neopixels sont censées être alimentées en 5 volts, et le micro:bit ne sort un voltage d'alimentation que de 3 volts.
Cela peut poser problème sur pas mal d'autres éléments d'électronique, mais les Neopixels ne sont pas trop strictes dans leurs spécifications, le 3 volts fait largement l'affaire tant qu'on en reste à une longueur en en gros 1 mètre.
De plus, même à 3 volts elles peuvent assez vite consommer beaucoup d'énergie, donc 2 piles AAA peuvent vite être vidées (avoir des remplacement sous la main serait utile en cas d'usage en live !)

- La seconde, c'est que l'usage du connecteur Edge sur lequel on branche la bande de leds n'est pas anodin non plus, car il permet d'accéder aisément à l'ensemble des autres GPIO du micro:bit, ce qui peut permettre de très vite augmenter les capacité de sa machine !

(pour les plus avancés, la liste des pins sur le bord du micro:bit est là : https://tech.microbit.org/hardware/edgeconnector)

Notes personnelles

Un espace de plusieurs pages pour noter des trucs !