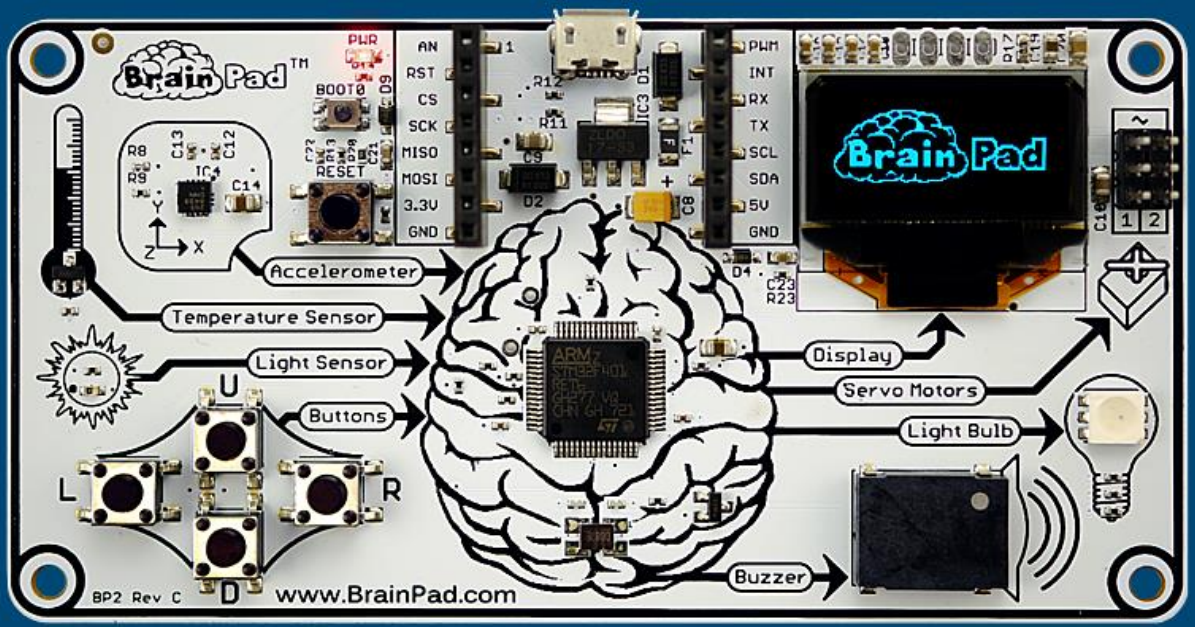
GHI electronics

BrainPad™

BEGINNERS' GUIDE



CONTENU

INTRODUCTION

La carte BrainPad est un puissant outil pédagogique permettant d'aborder les sciences, la technologie, l'ingénierie et les mathématiques destiné à tous, aux enfants comme aux étudiants et même aux professionnels.

Ce livre est gratuit. Il permet de débuter la programmation de cette carte. D'autres ressources peuvent être trouvées sur : http://www.brainpad.com.

L'AUTEUR

Gus Issa est le fondateur et le PDG de GHI Electronics. Il a commencé à bricoler des programmes et des circuits électroniques alors qu'il n'était encore qu'un adolescent. À ses débuts, disposant de peu de moyens et sans accès à Internet il s'est formé à force de persévérance. Les difficultés rencontrées et l'impact de l'éducation sur sa vie l'ont incité à partager ses connaissances.

Ne comptant pas son temps et exploitant au mieux les ressources de son entreprise, ses efforts ont abouti à la conception de cet outil pédagogique qu'est la carte BrainPad.

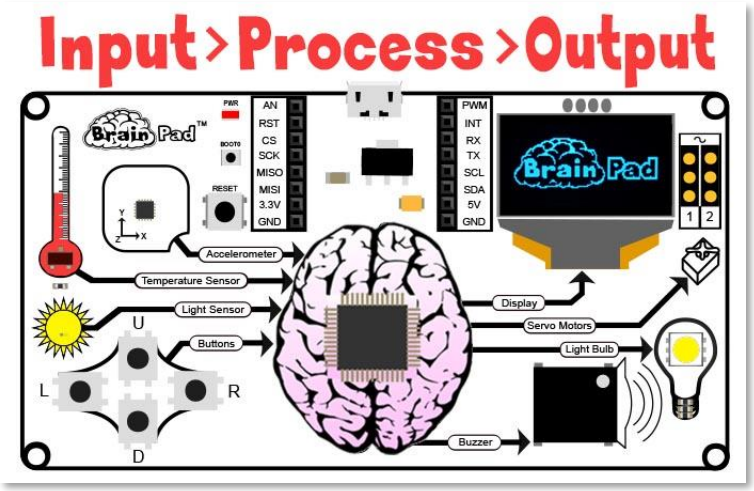
REMARQUE

Nous travaillons dur pour rendre le BrainPad accessible à tous. Si vous pensez pouvoir traduire ce livre, faites le nous savoir ici : http://www.brainpad.com/contact-us.

LICENSE

LE CONCEPT

Le BrainPad est intuitif, il présente de façon logique le fonctionnement d'un ordinateur. Il y a quatre entrées connectées au cerveau. Le cerveau (le processeur) réfléchit à ce qui doit être fait pour contrôler les quatre sorties.



STEM

« STEM » signifie Science, Technologie, Ingénierie et Mathématiques. L'acronyme STEM a été introduit en 2001 par la U.S. National Science Foundation. STEM est maintenant l'un des sujets dont on parle le plus dans l'éducation. L'éducation STEM, cependant, couvre bien plus que ces quatre champs disciplinaires. L'approche éducative par STEM a pour but de relier les apprentissages dans la classe avec le monde réel en soulignant les notions de communication, de collaboration, de sens critique et de créativité, tout en apprenant le processus de conception de l'ingénierie. Le terme « STEAM » est comme celui de STEM avec un accent plus particulier sur les Arts.

LA PHILOSOPHIE

L'éducation STEM nécessite une plateforme évolutive, pas un jouet. Lorsque vous progressez, BrainPad évolue. De nombreux jouets sont commercialisés comme des outils STEM, mais la plupart d'entre eux manquent de la polyvalence nécessaire pour que les élèves restent motivés sur une longue période. La carte BrainPad permet de progresser en programmation (du débutant au professionnel), en électronique, en ingénierie, en robotique, et bien plus encore en mettant en oeuvre une plateforme peu coûteuse et exploitable de l'école primaire à l'université. Ceci grâce à nos 15 ans d'expérience professionnelle en ingénierie. La carte BrainPad peut utiliser les mêmes outils de programmation que ceux que nos clients professionnels utilisent avec leurs microcontrôleurs. Aucune autre plate-forme STEM ne peut faire passer les étudiants de la programmation par blocs à la programmation et à l'ingénierie professionnelles.

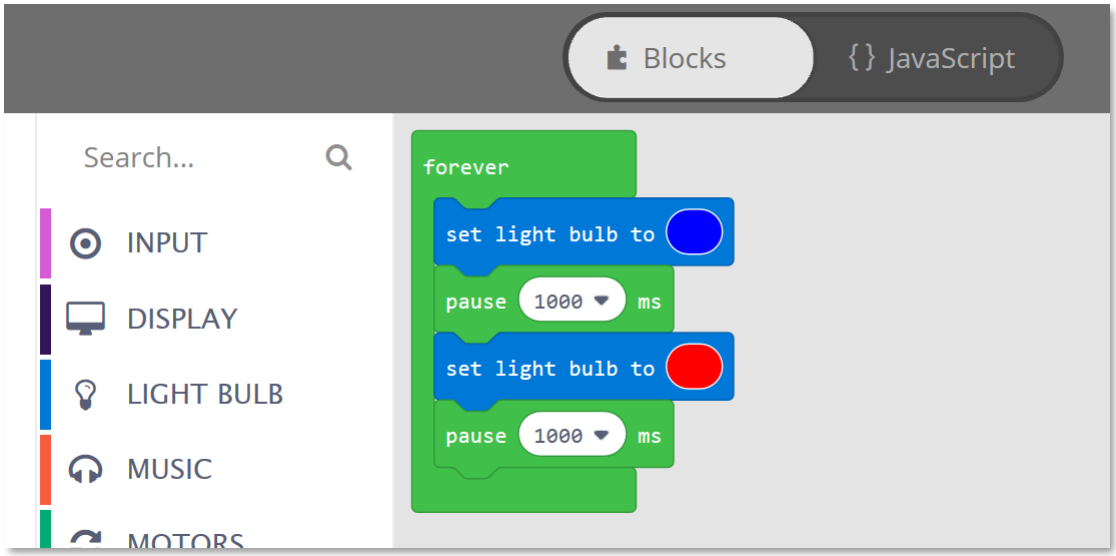
DEUX APPROCHES

Le BrainPad propose deux approches pour apprendre à programmer : **Débuter (Start Making)** et **Pour aller plus loin (Go Beyond)**. Comme vous pouvez l'imaginer, la première facilite les projets « Pour débuter », la seconde vous emmène plus loin et vous permet d'utiliser les mêmes outils de programmation que ceux mis en oeuvre par des professionnels.

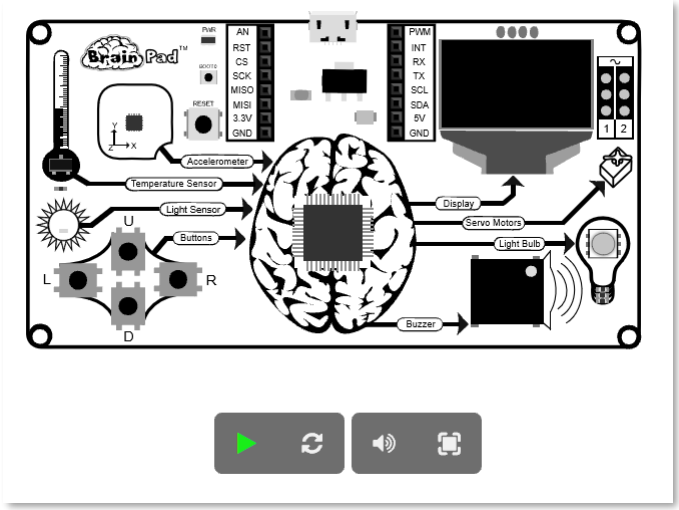
DEBUTER – START MAKING

Avec la méthode « Débuter », il n'y a rien à installer sur votre ordinateur – tout fonctionne dans un navigateur grâce à Microsoft MakeCode. « Pour aller plus loin », nécessite l'installation et la configuration de logiciels. Le logiciel couramment utilisé pour aller plus loin est Microsoft Visual Studio.

Avec Microsoft MakeCode (« Débuter »), vous programmez simplement en arrangeant des blocs et éventuellement en tapant du code JavaScript dans votre navigateur Internet. Dans le programme ci-dessous, la DEL Light Bulb s'éclaire en bleu puis en rouge toutes les secondes et ceci indéfiniment (« forever »).



Le simulateur intégré est un BrainPad virtuel qui exécute votre programme sur l'écran de l'ordinateur. Vous le trouvez sur le côté gauche de la fenêtre de Microsoft MakeCode. Vous pouvez ainsi tester votre programme sans le charger dans le BrainPad. C'est une excellente solution si vous ne l'avez pas encore reçu.



POUR ALLER PLUS LOIN - GO BEYOND

Les choses se compliquent avec l'approche "Pour aller plus loin". Vous aurez à configurer un ordinateur avec Microsoft Visual Studio pour coder et déboguer vos programmes dans le BrainPad.

Grâce au TinyCLR OS™ de GHI Electronics, le BrainPad peut être programmé sous .NET en C# ou en Visual Basic. Toutes les fonctionnalités de débogage comme l'exécution en mode pas-à-pas, les points d'arrêt et l'inspection des variables lors de l'exécution sont disponibles. Vous allez programmer le BrainPad de la même façon que le ferait un professionnel développant un programme pour PC ou une application pour téléphone portable. Ce livre électronique vous fera découvrir les deux approches.

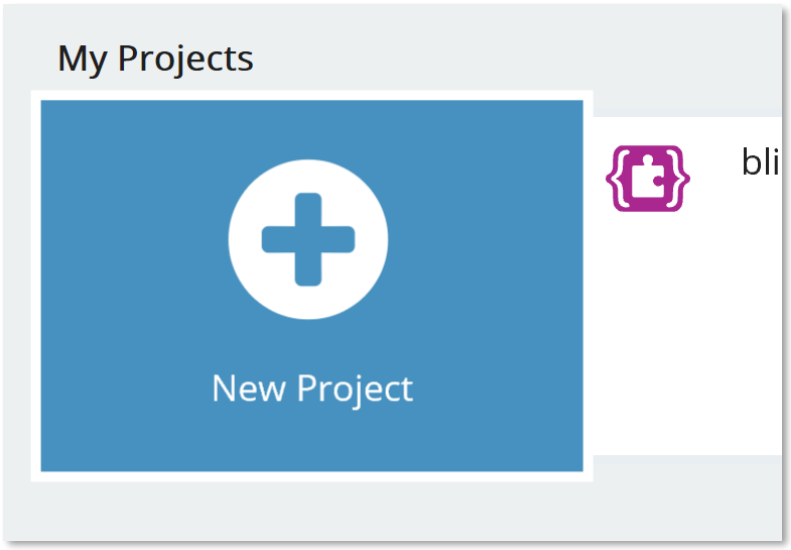
```
using System;
using System.Collections;
using System.Text;
using System.Threading;
using GHIElectronics.TinyCLR.BrainPad;

namespace BlinkLED {
    class Program {
        static void Main() {

            for(int i = 0; i < 100; i++) {
                BrainPad.Display.DrawNumberAndShowOnScreen(10, 10, i);
            }
        }
    }

    public static class BrainPad {
        public static Accelerometer Accelerometer { get; } = new Accelerometer();
        public static Buttons Buttons { get; } = new Buttons();
        public static Buzzer Buzzer { get; } = new Buzzer();
        public static Display Display { get; } = new Display();
        public static LightBulb LightBulb { get; } = new LightBulb();
        public static LightSensor LightSensor { get; } = new LightSensor();
        public static ServoMotors ServoMotors { get; } = new ServoMotors();
        public static TemperatureSensor TemperatureSensor { get; } = new TemperatureSe
        public static Wait Wait { get; } = new Wait();
```

POUR ALLER ENCORE PLUS LOIN !

Le BrainPad peut aussi être programmé avec d'autres méthodes qui ne sont pas abordées dans ce livre. Par exemple avec les outils Arduino https://www.arduino.cc, MicroPython https://www.micropython.org et Arm Mbed https://www.mbed.com.

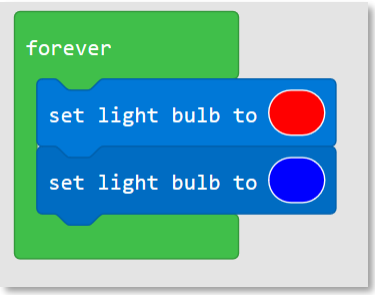
DEBUTER

Assez parler – Commençons à nous amuser et à réaliser des projets ! Nous allons utiliser Microsoft MakeCode donc rendez-vous sur https://makecode.brainpad.com/. Le site inclut un grand nombre de ressources et de projets, mais nous allons juste commencer par réaliser le notre. Allez-y et cliquez sur le bouton « Nouveau Projet » (New Project).



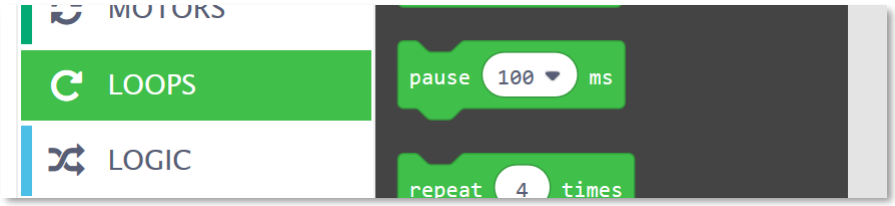
Maintenant, cliquez sur le menu LIGHT BULB, glissez le bloc « set light bulb to » et déposez-le dans la boucle « forever ». Cette boucle est utilisée pour le code devant fonctionner indéfiniment. Les instructions à l'intérieur du bloc forever vont s'exécuter dès que le BrainPad sera alimenté. Dès que l'ensemble des instructions contenues dans la boucle auront été exécutées, le programme s'exécutera à nouveau en commençant par la première. Cela s'appelle une boucle infinie et c'est très souvent utilisé en programmation.

Recommencez les mêmes étapes, mais dans le second bloc, cliquez dans l'ovale coloré pour changer la couleur. Voici ce que vous devriez obtenir à présent :

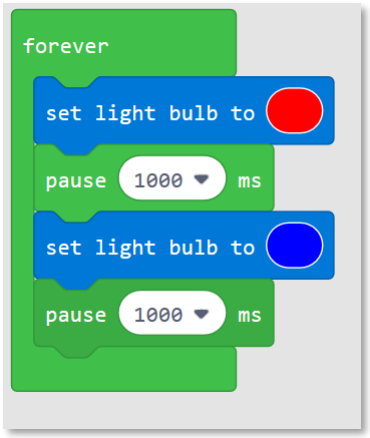


Le programme que nous avons créé va indéfiniment faire passer la couleur de la DEL Light Bulb au rouge puis au bleu ! Cependant, si vous observez le comportement du simulateur, le Light Bulb ne se comporte pas comme prévu. Si vous chargez le programme dans le BrainPad, cela ne va pas fonctionner correctement non plus. Pourquoi ? Vous n'avez tout simplement pas indiqué combien de temps attendre avant de changer la couleur du Light Bulb. Cela se passe tellement rapidement que vous ne pouvez pas différencier les couleurs.

À partir du menu LOOPS, glisser le bloc « pause ».



Nous aurons besoin d'une pause après que le Light Bulb soit passé à rouge et d'une autre après que le Light Bulb soit devenu bleu. Changez également le temps de pause à une seconde.



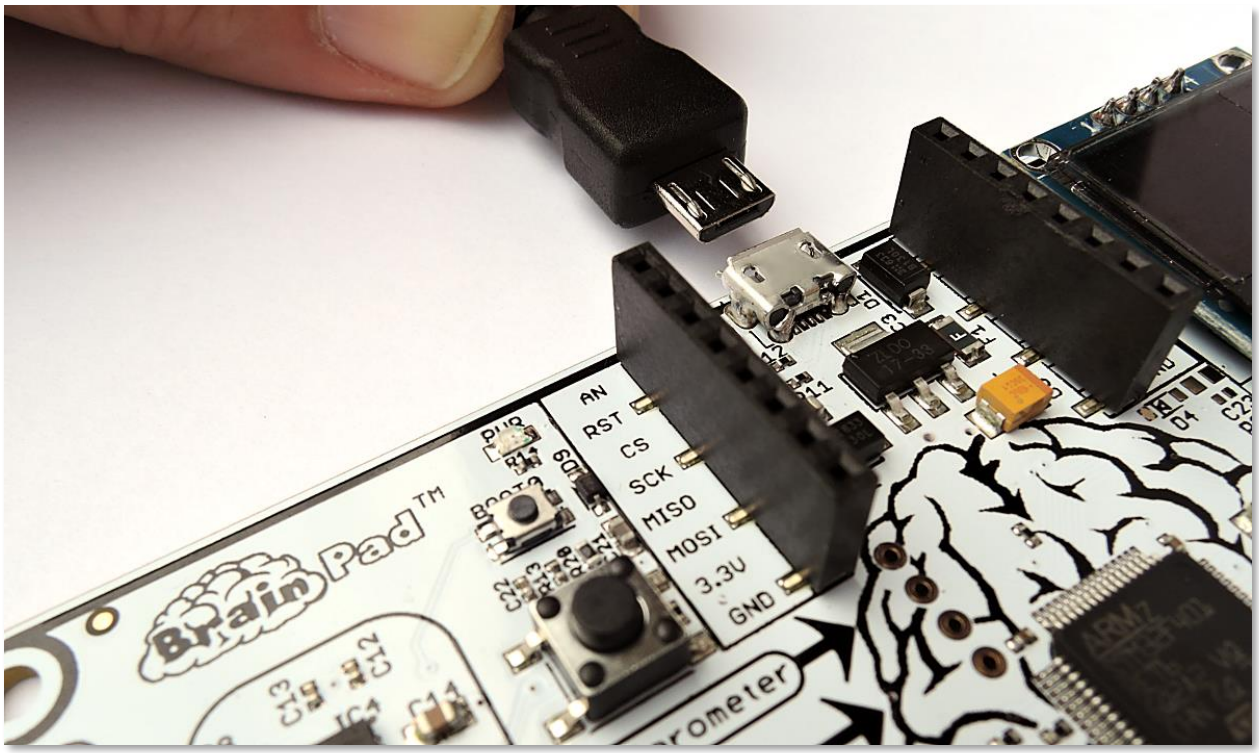
Observez le simulateur pour vérifier que le Light Bulb fonctionne comme prévu. Pas mal, non ? Que diriez-vous de charger le programme dans le BrainPad maintenant ?

TELECHARGER LE PROGRAMME DANS LE BRAINPAD

Le processus de copie du programme dans le BrainPad est simple, mais peut sembler un peu complexe la première fois. Premièrement, vous devez connecter le BrainPad à l'ordinateur en utilisant un câble Micro-USB. Vous pouvez utiliser Windows, un ChromeBook, un Mac ou quasiment n'importe quel matériel disposant d'un navigateur Internet récent et d'un port USB.

Un câble Micro-USB devrait être inclus avec votre BrainPad. Vous en avez certainement d'autres à la maison. Si vous n'en avez pas, ils sont facilement disponibles. Ils sont même vendus dans les stations-service !

L'extrémité la plus petite du câble USB se branche sur le BrainPad.

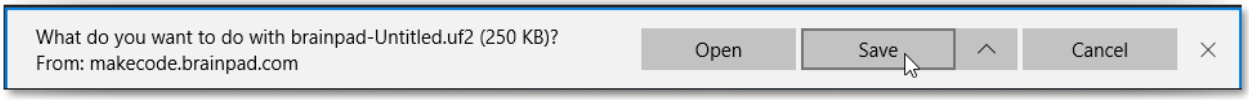


L'extrémité la plus large se connecte à votre ordinateur.

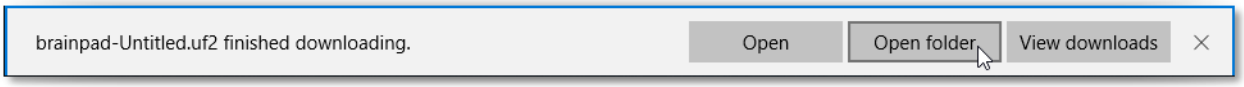
Lorsque le BrainPad est connecté à votre ordinateur, le voyant d'alimentation rouge (PWR) doit être allumé.

Revenez dans le navigateur contenant le programme que nous avons fait précédemment. Cliquez sur le bouton de téléchargement (Download) et enregistrez le fichier sur votre ordinateur.

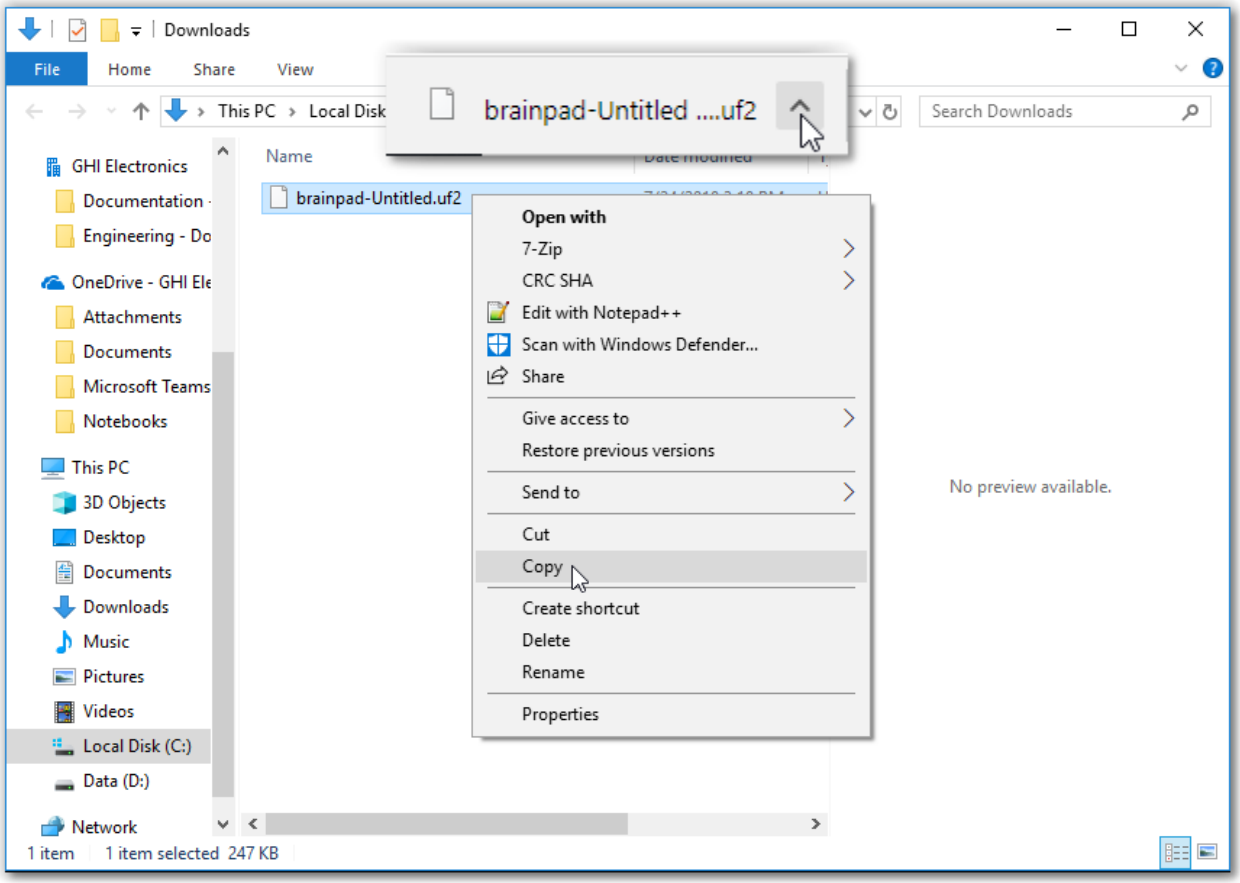
Si vous utilisez Microsoft Edge, vous devriez voir une boîte de dialogue comme celle représentée ci-dessous :



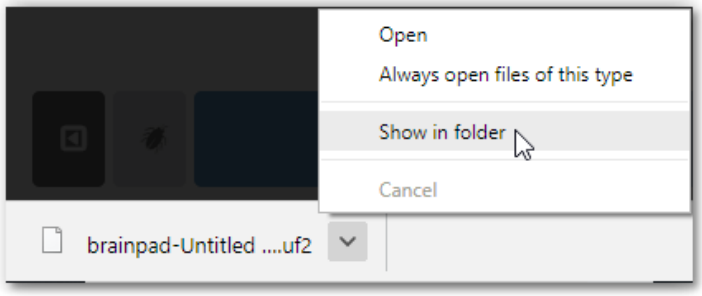
Cliquez sur le bouton Enregistrer (Save), puis sur le bouton "Ouvrir le dossier" (Open folder) dans la boîte de dialogue suivante pour afficher le fichier téléchargé.



Le fichier sera mis en surbrillance. Vous pouvez faire un clic droit dessus pour le copier.

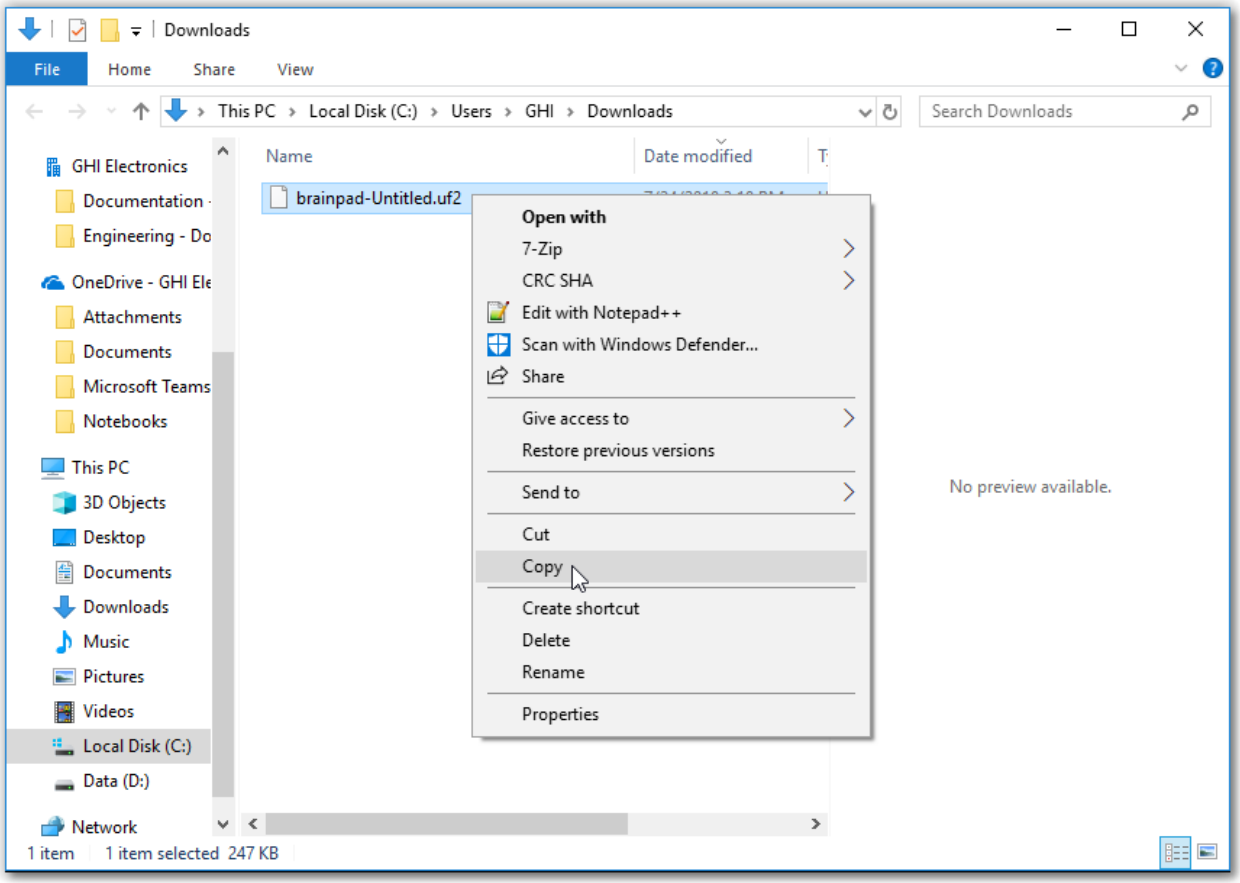


Si vous utilisez le navigateur Chrome, le fichier sera affiché dans le coin inférieur gauche de l'écran :



Cliquez sur la petite flèche vers le haut, puis sélectionnez "Afficher dans le dossier" (Show in folder).

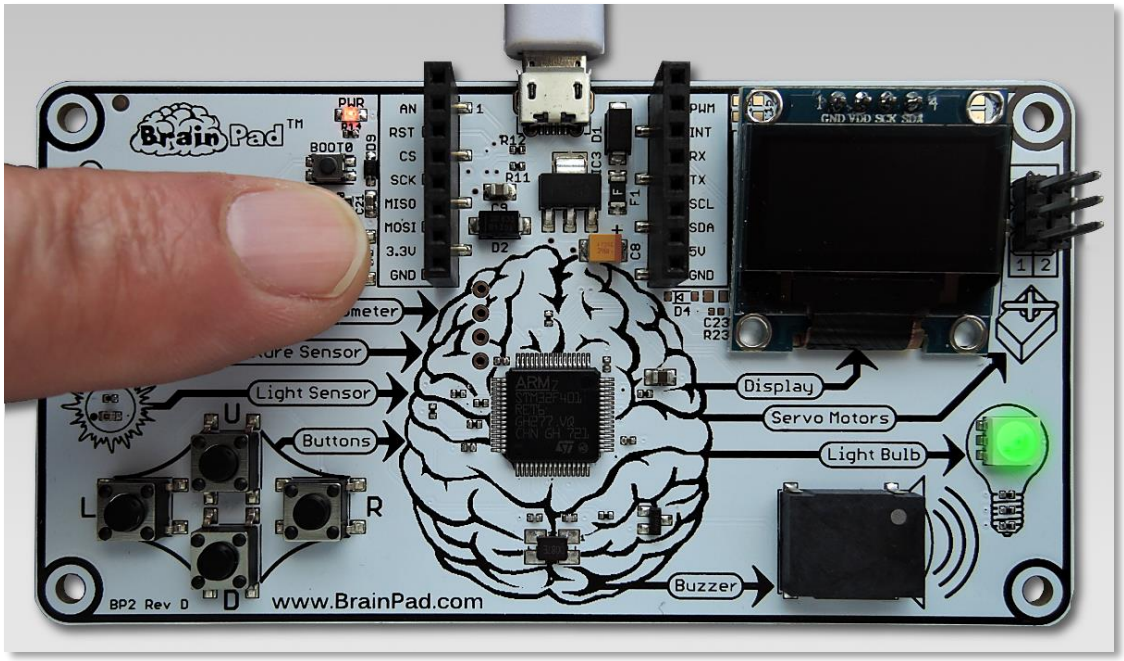
Le fichier téléchargé sera mis en surbrillance. Vous pouvez faire un clic droit sur le fichier pour le copier.



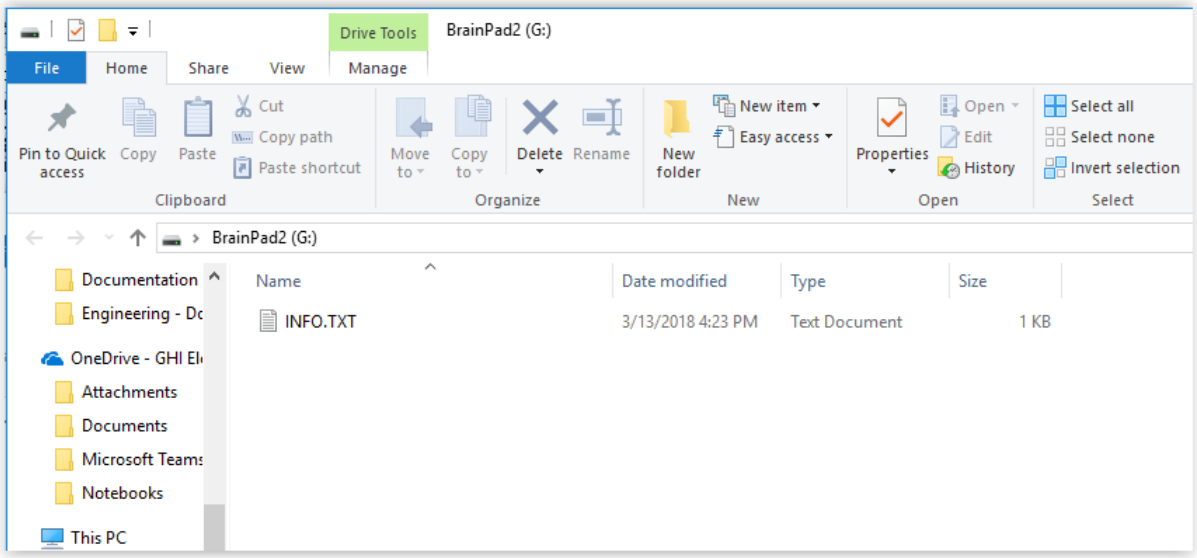
Avec ces deux navigateurs, Microsoft MakeCode ne supprime pas vos anciens fichiers. Si vous téléchargez le même programme plusieurs fois, un numéro sera ajouté au nom de fichier (par exemple "brainpad-Untitled (1).uf2"). Assurez-vous de copier le dernier fichier. Ce dernier fichier sera le fichier en surbrillance. Il aura aussi le plus grand nombre entre parenthèses et la dernière date / heure.

Les autres navigateurs devraient avoir le même comportement.

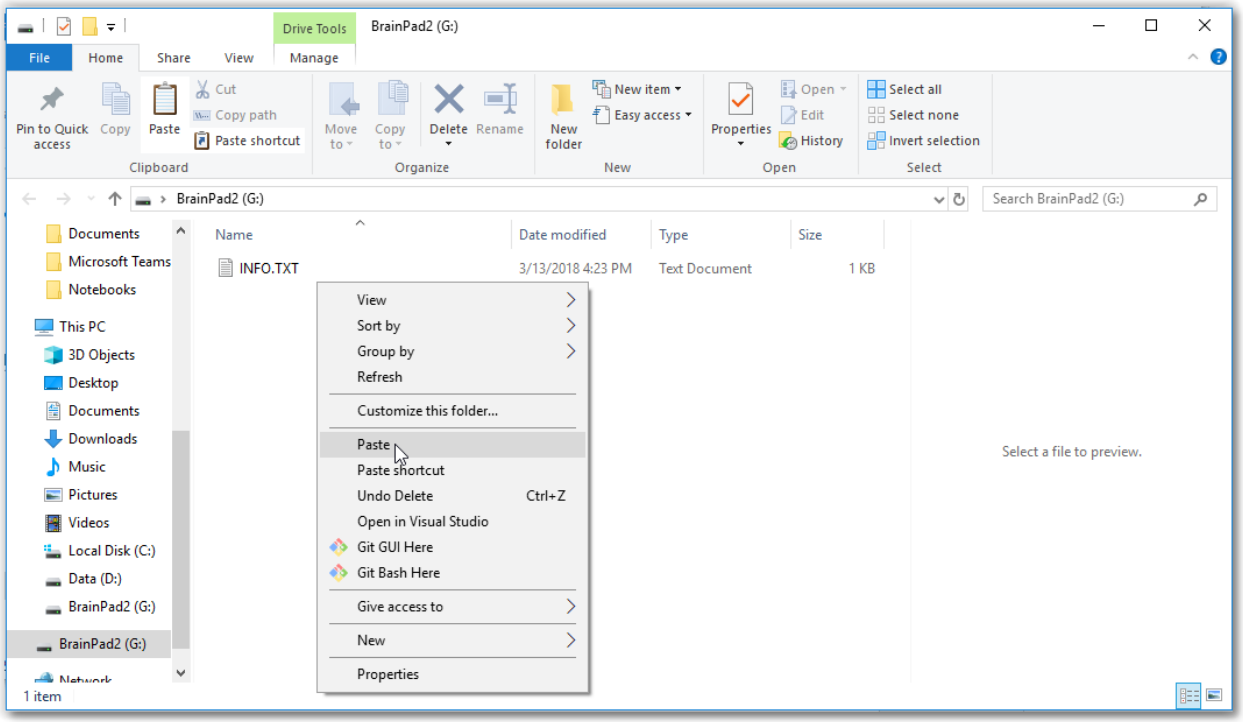
Le BrainPad doit être connecté à votre ordinateur et le voyant d'alimentation rouge doit être allumé. Appuyez et maintenez le bouton de réinitialisation pendant environ trois secondes. Le Light Bulb devient vert.



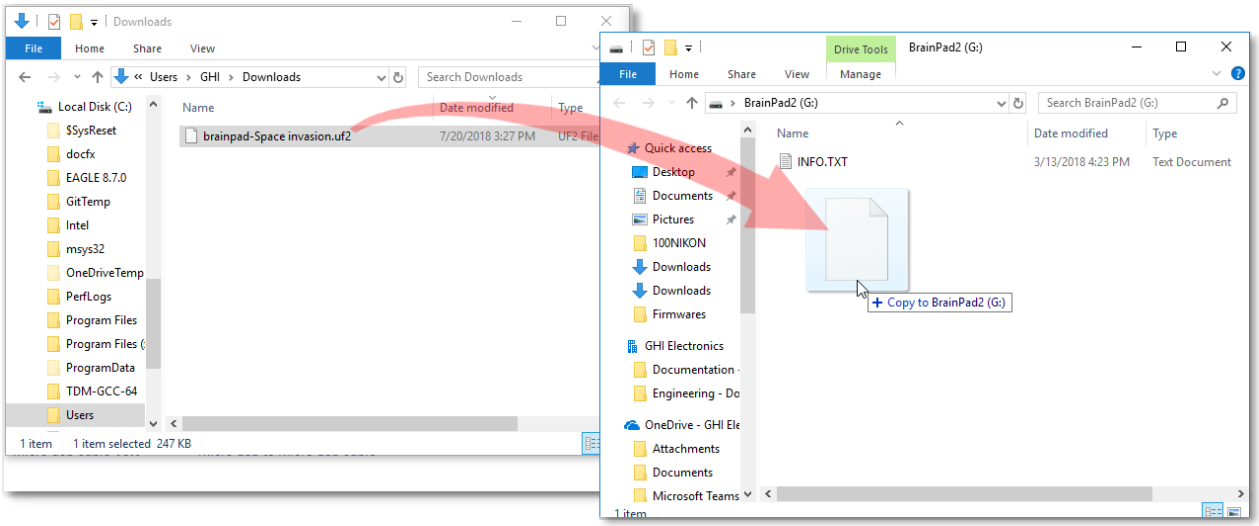
Votre ordinateur va maintenant détecter un périphérique de stockage USB. Cela peut prendre un peu de temps, mais vous verrez apparaître une fenêtre nommée "BrainPad".

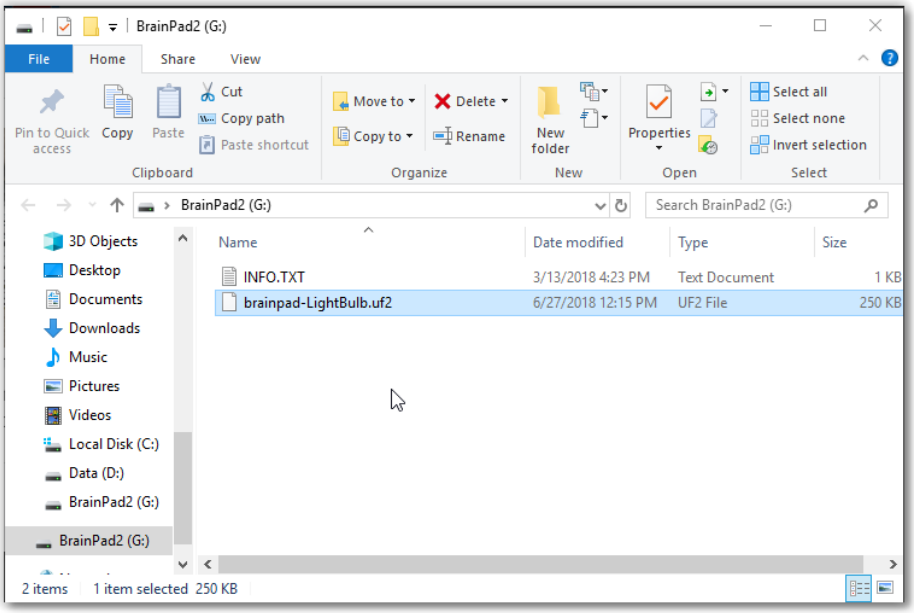


Cette fenêtre montre le contenu de la mémoire du BrainPad. Faites un clic droit et sélectionnez coller (Paste) pour déposer le fichier dans le BrainPad .



Vous pouvez aussi copier le fichier en effectuant un glisser-déposer.

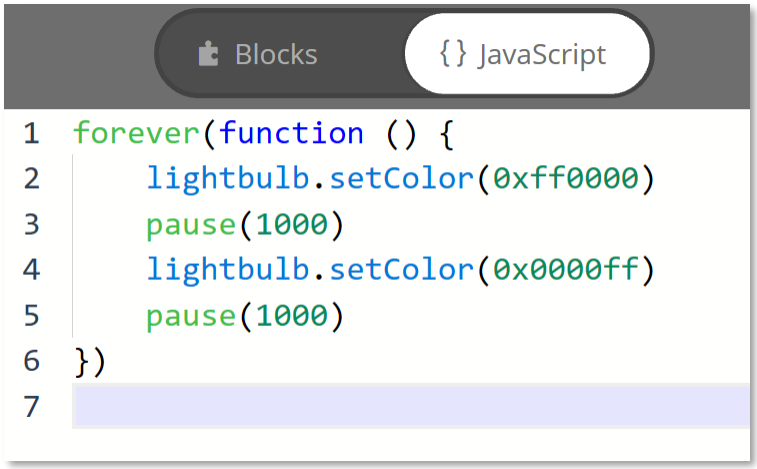




Après toutes ces manipulations, le Light Bulb clignotera un moment et le BrainPad exécutera le programme chargé. Maintenant, les choses deviennent plus intéressantes !

JAVASCRIPT

Les blocs sont amusants pour commencer, mais lorsque vos programmes deviennent plus volumineux, vous devrez vous enhardir et commencer à écrire du code. Ne vous inquiétez pas, vous pouvez le faire une fois que vous êtes prêt. Une des caractéristiques étonnantes de Microsoft MakeCode est la façon dont il convertit les blocs en code JavaScript et le code JavaScript en blocs. Revenez au programme que nous avons créé auparavant et cliquez sur l'onglet JavaScript.



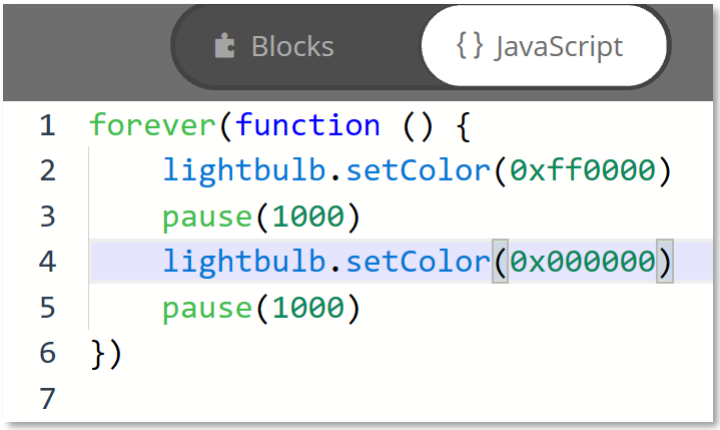
Observez le code. Remarquez-vous la corrélation entre les blocs et le code ? Vous ne comprendrez certainement pas comment est représentée la couleur car le code utilise une valeur hexadécimale.
Comprenez-vous le reste du code ?

L'hexadécimal est un système numérique basé sur seize valeurs constituant un chiffre et non pas 10 comme habituellement. N'importe quel chiffre dans un nombre hexadécimal peut prendre les valeurs allant de 0 à F au lieu de 0 à 9 en décimal (les nombres que nous utilisons tous les jours). Compter de 0 à 16 en hexadécimal ressemblera à cela : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10.

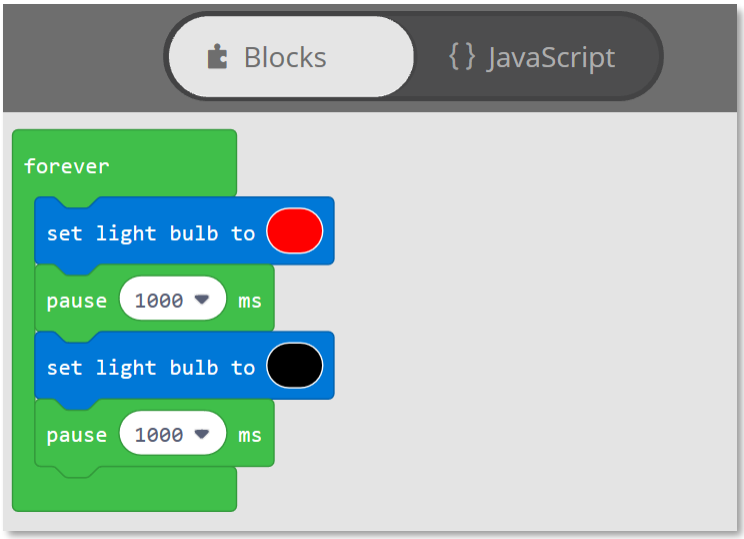
Le « 0x » au début du nombre précise que ce nombre est bien hexadécimal et évite ainsi la confusion avec une autre base. Les nombres hexadécimaux sont aussi connus sous l'abréviation « hex » en anglais et également appelés « base 16 ». Le binaire est aussi appelé « base 2 » et le décimal « base 10 ». Si cela vous parait compliqué, ne vous inquiétez pas. Il vous faudra un peu de temps avant d'être à l'aise avec les différents systèmes numériques.

Lorsqu'un nombre hexadécimal est utilisé pour définir une couleur, les deux premiers chiffres représentent la quantité de rouge dans la couleur, les deux suivants, la quantité de vert et enfin les deux derniers la quantité de bleu. « 00 » signifie que l'intensité de la couleur est à zéro, ou est éteinte. « FF » signifie que la couleur est à sa pleine intensité. Le rouge intense sera donc 0xFF0000. Le blanc sera représenté par 0xFFFFFF.

Quel est l'effet de 0x000000 dans l'une des méthodes « setColor » ? Essayons.



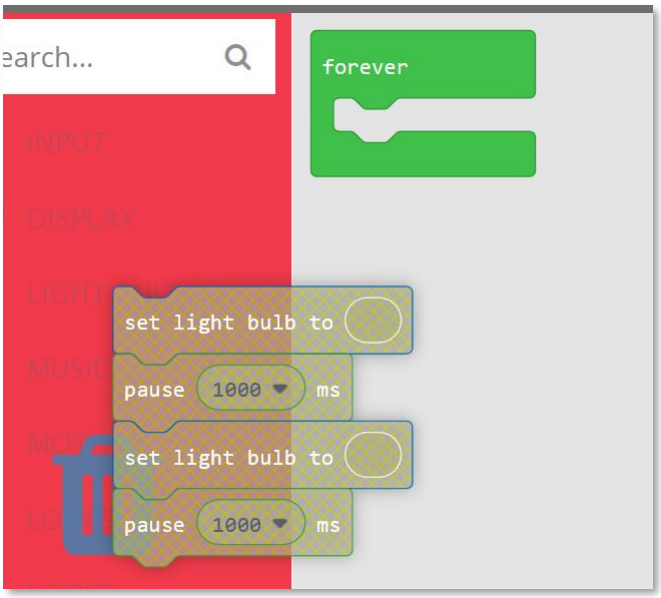
Maintenant, repassons aux Blocs pour voir le résultat.



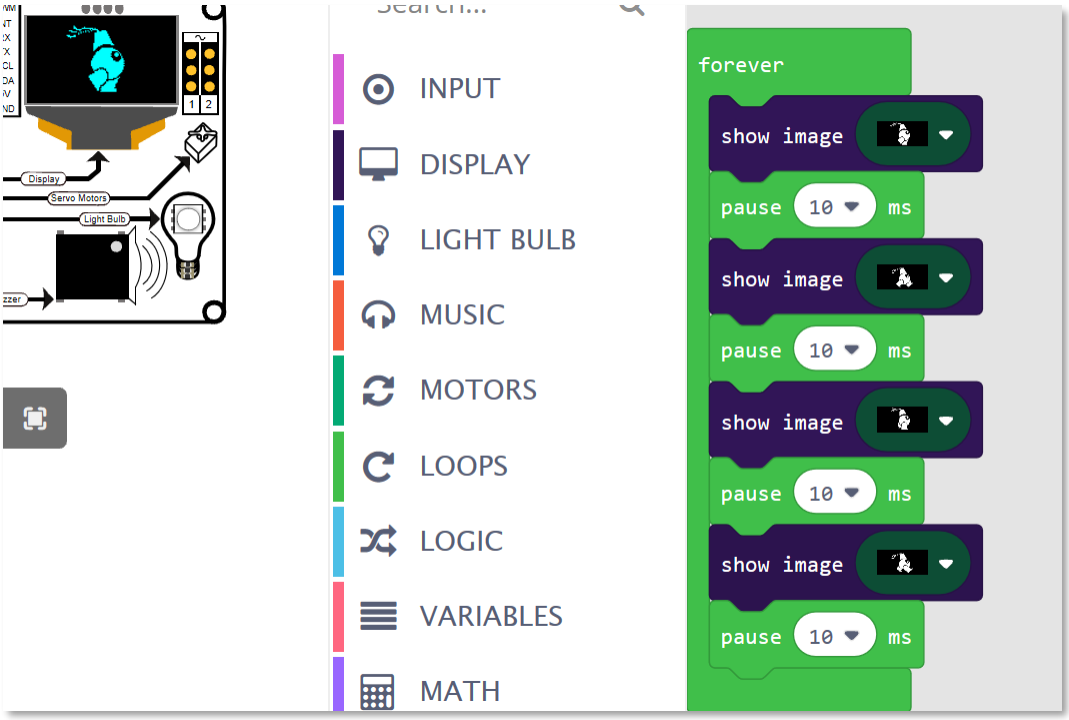
C'est comme si la couleur était passée du bleu au noir. Vous avez pu deviner que donner la couleur noire au Light Bulb revient à l'éteindre. Vous pouvez voir cela dans le simulateur. Le Light Bulb devrait être rouge pendant une seconde puis s'éteindre pendant une seconde.

AFFICHAGE AMUSANT

Si vous pensez que le Light Bulb est excitant, attendez d'essayer l'affichage ! Si vous avez toujours les blocs de l'exemple précédent, glissez-les dans le menu pour les effacer.



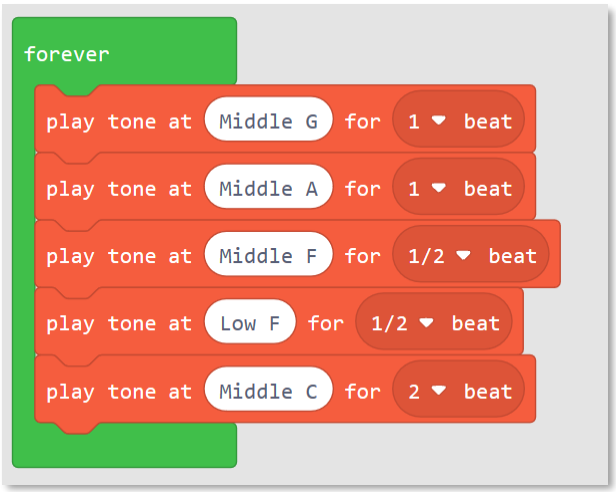
Si vous avez effacé par mégarde le bloc « forever », vous pouvez simplement le récupérer dans le menu « LOOPS ». Depuis le menu « DISPLAY », faites glisser le bloc « show image » et sélectionnez la première des quatre images de l'homme en train de marcher « walking man ». Ajoutez chacune des trois autres images et ajoutez une pause de 10 ms entre. Il n'y a pas l'option 10ms dans le bloc, mais vous pouvez la saisir avec le clavier.



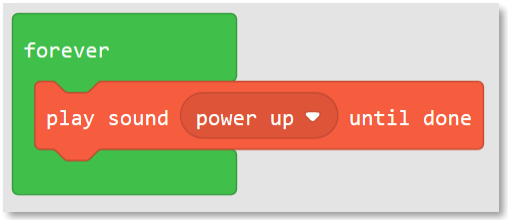
Notre personnage devrait marcher sur l'écran du simulateur maintenant. Poursuivons en téléchargeant ce programme dans le BrainPad.

JOUER DES NOTES DE MUSIQUE

Depuis le menu « MUSIC », vous pourrez trouver des notes…



… ou des sons.

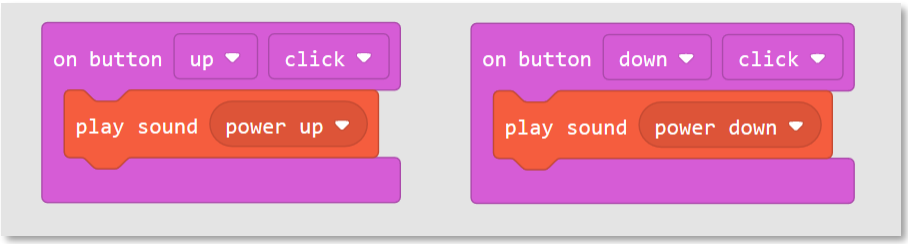


Remarquez comment on utilise le bloc « play sound until done » : ce bloc suspend l'exécution du reste du programme jusqu'à ce que le son soit complètement joué. Le bloc « play sound » (sans le « until done ») commence à exécuter le prochain bloc avant la fin du précédent. Si vous essayez les deux types de blocs, vous verrez que les résultats sont très différents.

UTILISER LES BOUTONS

Il y a deux méthodes pour lire l'état des boutons. L'une consiste simplement à vérifier si le bouton est dans l'état appuyé et à effectuer l'action appropriée dans ce cas. Le problème de cette méthode est que vous devez vérifier sans arrêt l'état du bouton pour être sûr de ne pas rater un appui. Une autre méthode consiste à laisser le système le faire pour vous, et dans le cas où l'évènement (event) surviendrait, réaliser automatiquement une action.

Nous allons voir en premier lieu comment lire l'état d'un bouton avec les évènements. Nous voulons faire jouer le son « power up » lorsque le bouton haut (up) est appuyé, et le son « power down » lorsque le bouton bas (down) est appuyé. Cet exemple est repris plus loin dans le document mais si vous voulez le tester, Le bloc « on button » est dans le menu « INPUT ».



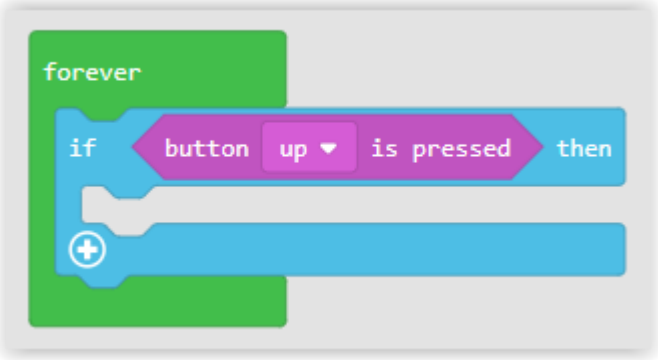
Avez-vous remarqué qu'il n'y a pas de bloc « forever » dans cet exemple ? Vous pouvez toujours utiliser un bloc « forever » dans votre code, mais il n'est pas nécessaire ici. C'est parce que le système va s'occuper automatiquement de l'évènement (event) lorsque le bouton sera appuyé.

Voici le code JavaScript équivalent au bloc ci-dessus :

```
input.buttonU.onEvent(ButtonEvent.Click, function () {
    music.playSound(music.sounds(Sounds.PowerUp))
})

input.buttonD.onEvent(ButtonEvent.Click, function () {
    music.playSound(music.sounds(Sounds.PowerDown))
})
```

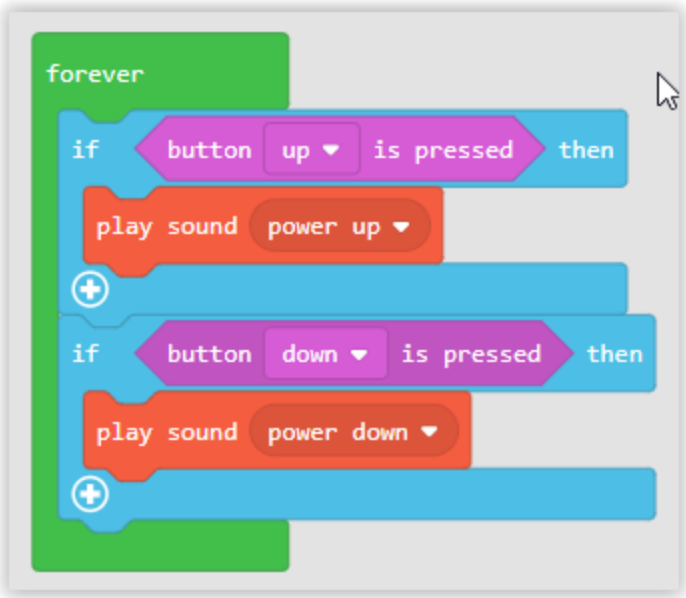
Cela fonctionne bien, mais revenons à notre bonne vieille boucle « forever ». Nous allons créer un programme qui va faire exactement la même chose, mais sans évènement. Pour vérifier si (if) un bouton est appuyé, vous allez utiliser une instruction « if ». Vous la trouverez dans le menu « LOGIC ». Glissez-la dans la boucle « forever ».

Le « if » vérifie l'état de quelque chose qui peut être soit vrai (true) soit faux (false). Par défaut, cela indique « true », mais nous voulons le remplacer par l'état du bouton. Pour cela, glisser le bloc « button … is pressed » depuis le menu INPUT dans le bloc « if … then » comme cela a été vu précédemment. Notez que ce n'est pas l'évènement « on button » utilisé précédemment.



Le bloc ci-dessus va vérifier l'état du bouton haut (up) indéfiniment. Si le bouton est appuyé au moment où il est vérifié, le programme va exécuter tous les blocs situés à l'intérieur de l'instruction « if ».

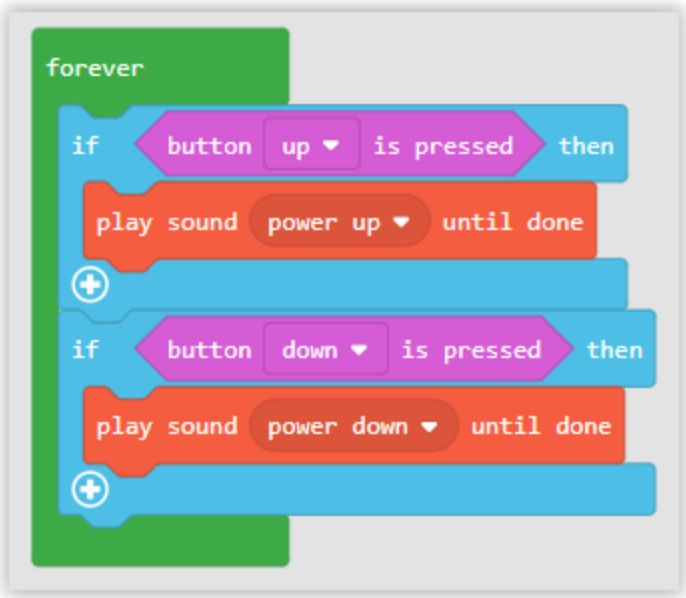
Ajoutez maintenant quelques blocs supplémentaires pour obtenir le programme suivant :



Lorsque vous exécuterez ce programme, vous remarquerez que les sons ne sont pas joués correctement tant que le bouton n'est pas relâché. Cela vient du fait que le Brainpad joue un nouveau son chaque fois qu'un des boutons haut ou bas est appuyé - même si le son précédent n'est pas terminé. Le Brainpad jouera donc le son plusieurs fois par seconde lorsque le bouton sera appuyé.

Il y a deux façons de régler ce défaut. Une solution consiste à remplacer le bloc « button … is pressed » par le bloc « button … was pressed ». Le bloc « button was pressed » sera vrai si le bouton a été appuyé au moins une fois depuis le dernier test du bouton. Même si le bouton a été appuyé plusieurs fois, un seul appui sera pris en compte. Appuyez sur le bouton haut ou bas lorsqu'un son est joué va interrompre le son, mais le son ne se répètera pas pendant que l'on maintient le bouton appuyé.

Une autre solution consiste à remplacer le bloc « play sound … » par le bloc « play sound … until done ». Le bloc « play sound until done » empêche le programme de faire autre chose jusqu'à ce que le son soit joué. En faisant cela, on empêche le programme de vérifier si le bouton est appuyé et de jouer un autre son tant que le son en cours n'est pas terminé. En utilisant le bloc « play sound until done », on empêche l'interruption du son. Les blocs sont représentés ci-dessous.

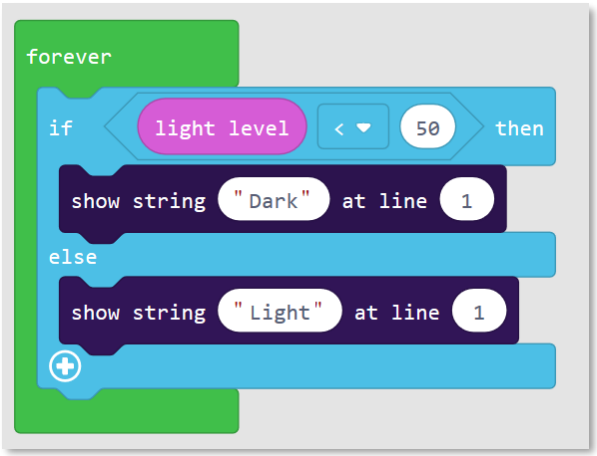


FAIT-IL NOIR ICI ?

Tout comme avec les boutons, vous pouvez lire le niveau de lumière ambiant et faire quelque chose, ou vous pouvez déclencher un événement, quand il fait clair ou sombre. Cependant, la lecture d'un bouton est légèrement différente de la lecture d'une grandeur physique telle que la luminosité.

Un bouton est enfoncé ou relâché. Alors, quand je dis : est-ce que le bouton est enfoncé ? Votre réponse sera vraie s'il est appuyé, et fausse sinon. Un capteur de lumière (Light Sensor) donne le niveau de lumière sous la forme d'une valeur. Pour les programmeurs, c'est une valeur analogique. Nous allons lire l'intensité de la lumière et l'afficher sur l'écran. Ceci est réalisé dans une boucle infinie (forever), le programme lit constamment le niveau de lumière et l'affiche sur l'écran.

Puisque vous êtes un expert dans l'utilisation de l'instruction if, pouvez-vous écrire « clair » ou « sombre » sur l'écran selon ce que renvoie le capteur de lumière ? Les choses se compliquent, mais nous allons vous expliquer. L'instruction « if » attend quelque chose qui soit vrai ou faux, comme "est-ce que le bouton est pressé ?". Le niveau de lumière n'est pas vrai ou faux - il peut prendre un grand nombre de valeurs. Si vous y réfléchissez, cela n'a aucun sens de dire « si la luminosité fait quelque chose ». Vous devez dire : « si la luminosité est plus claire (ou plus faible) qu'un certain niveau, alors il faut faire quelque chose ». Votre instruction « if » doit également avoir un « else » (sinon). Voir ci-dessous :



Remarque : Vous trouverez « light level » dans le menu « INPUT » et « show string » dans le menu « DISPLAY ».

Comment pouvons-nous interpréter ces blocs ?

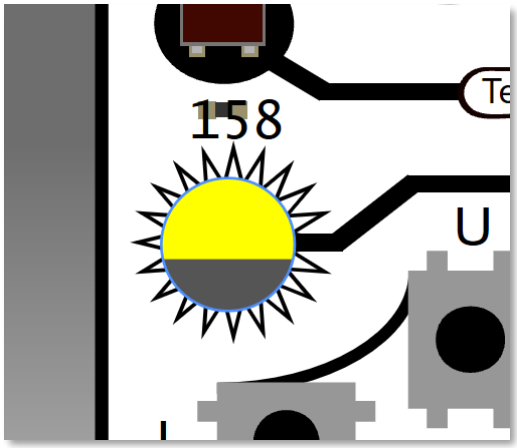
Si l'éclairage est inférieur à 50, écrivez "Dark" (sombre) sur la première ligne ; sinon, écrivez "Light" (clair) sur la première ligne.

N'ayez pas peur de regarder régulièrement dans l'onglet JavaScript pour voir à quoi ressemble le code.

```
forever(function () {
    if (input.lightLevel() < 50) {
        display.showString("Dark", 1)
    } else {
        display.showString("Light", 1)
    }
})
```

Le simulateur fonctionne également avec le capteur de lumière. Lorsque celui-ci est ajouté au projet, le simulateur va vous afficher de quoi régler (simuler) le niveau de lumière. Commencez par augmenter la taille du simulateur en cliquant sur le bouton « full screen » (plein écran).

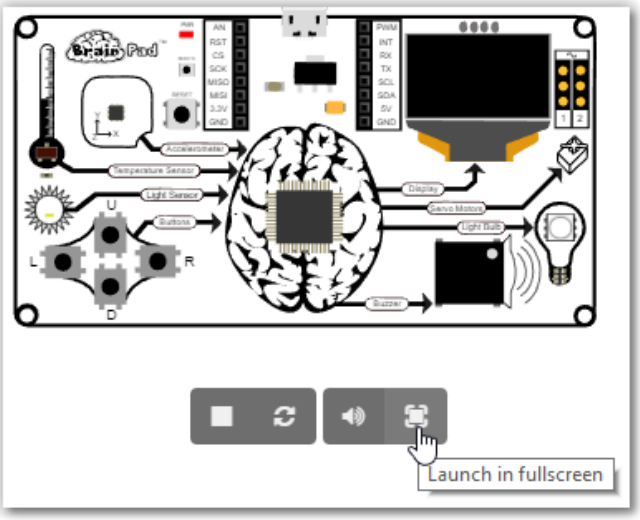
Le capteur de lumière montre alors le niveau de lumière pouvant être changé en déplaçant la ligne vers le haut ou vers le bas. Cela revient à placer votre main devant le capteur du BrainPad. Selon que vous déplacez le niveau de lumière vers le haut ou vers le bas, l'écran devrait alternativement afficher « Light » ou « Dark ».



TEST DE L'AUDITION

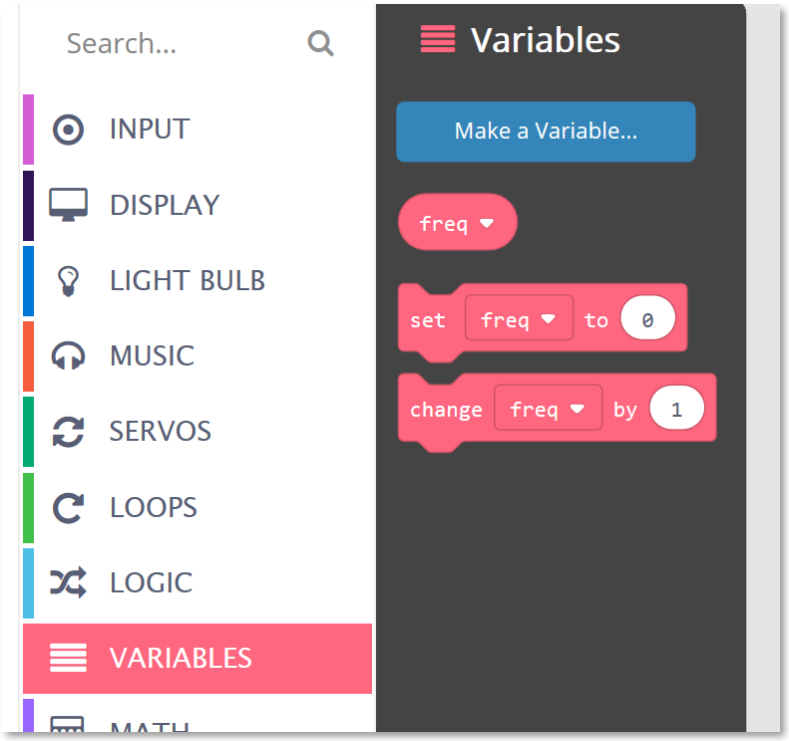
L'oreille humaine est capable de détecter des sons situés dans les basses fréquences, soit environ 20 Hertz (ou cycles par seconde) jusqu'aux hautes fréquences soit environ 20 000 Hertz. Les chiens peuvent entendre des sons de fréquences plus hautes que les humains, leurs oreilles captant jusqu'à 44 000 Hertz. C'est pour cette raison que nous ne pouvons pas entendre les sifflets pour chiens, mais eux les entendent parfaitement !

Lorsque l'on prend de l'âge, on perd la capacité à entendre les hautes fréquences. Le bruit engendre également une perte de l'audition plus importante dans les hautes fréquences que dans les basses. Voulez-vous connaître la fréquence maximum que peuvent percevoir vos oreilles ? Allons-y !

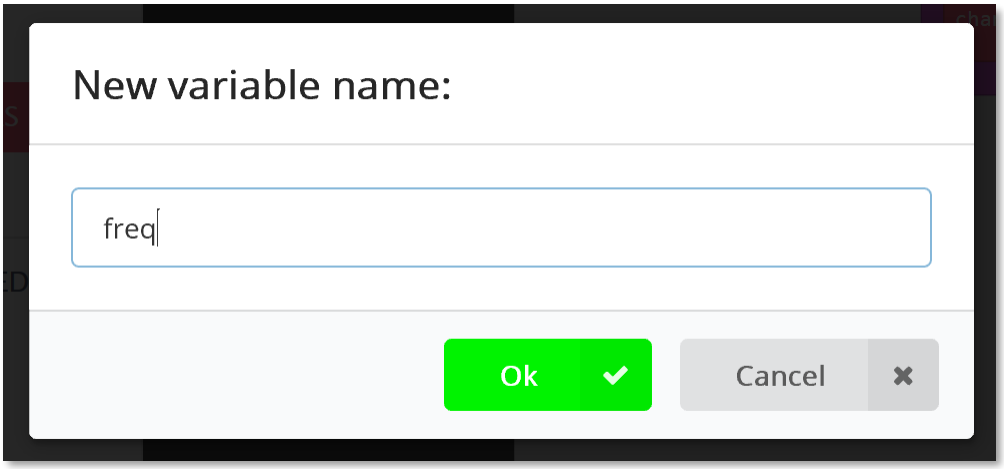


Nous allons devoir écrire un programme utilisant une « variable ». Une variable est un emplacement de stockage qui peut contenir une information et auquel on donne un nom. Dans notre cas, notre variable va contenir un nombre représentant la fréquence que nous ferons jouer au Buzzer.

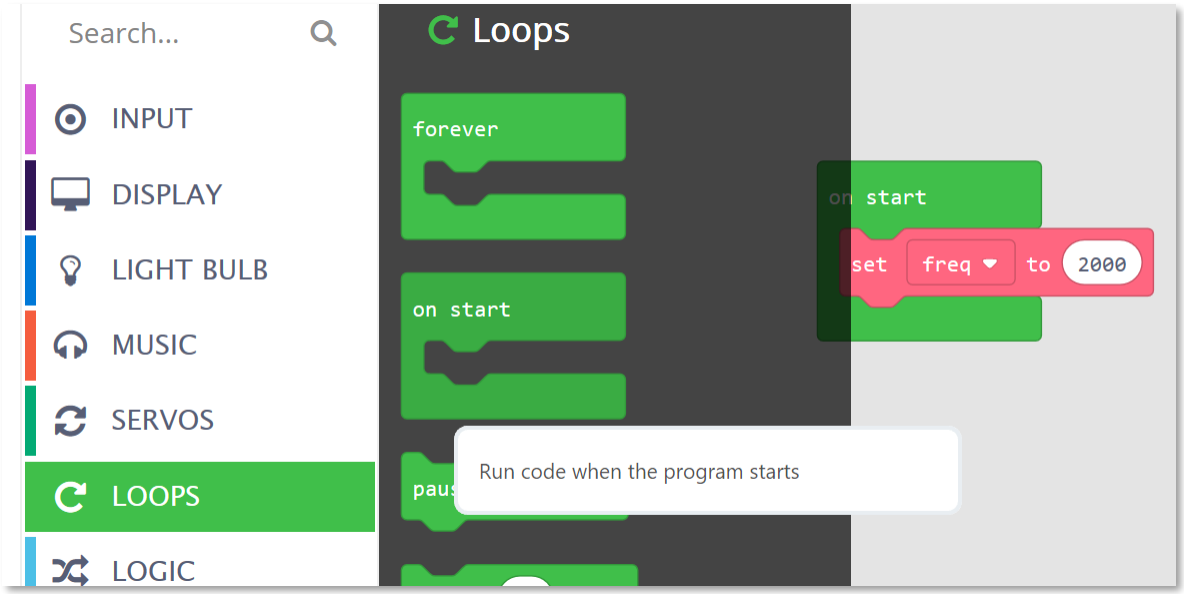
Allez dans VARIABLES et cliquez sur le bouton « Make a variable… » (Créer une variable).



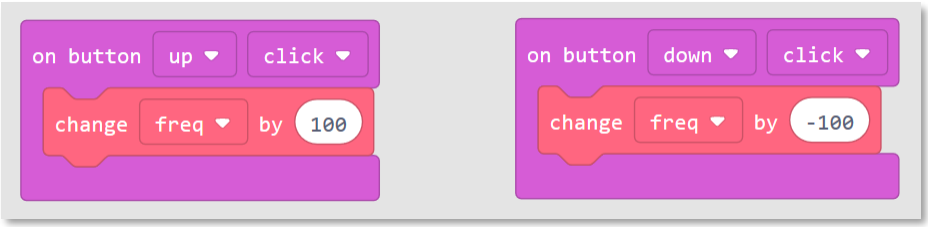
Donnez le nom « freq » à votre variable. Celle-ci stockera la fréquence que nous voulons faire jouer au Buzzer.



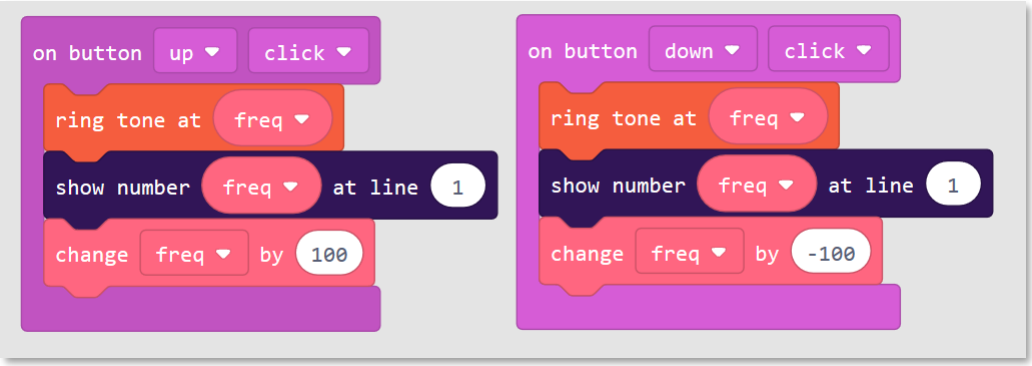
Allez dans le menu LOOPS et ajoutez un bloc « on start ». Quoi que vous ajoutiez dans ce bloc, cela s'exécutera lorsque le programme commencera. Ici, nous voulons régler la valeur à 2000. Nous pouvons tous entendre un son de fréquence égale à 2000 Hertz, c'est donc un bon point de départ. Le bloc « set » se trouve dans VARIABLES.



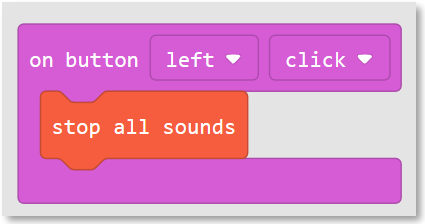
Nous devons créer un son correspondant à cette fréquence, mais nous voulons également augmenter la fréquence quand le bouton « up » est pressé et la diminuer lorsque le bouton « down » est pressé. Nous allons avoir besoin du bloc « change » du menu « VARIABLES ». Réglez la valeur à 100 lorsque « up » est appuyé et à -100 lorsque « down » est appuyé. Le bloc « on button » est dans le menu « INPUT ».



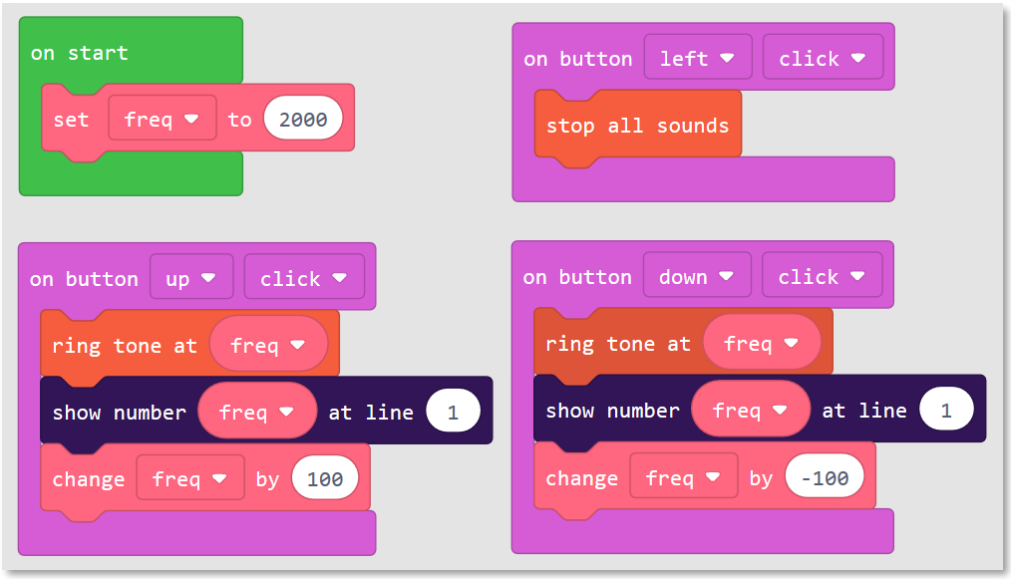
Maintenant que notre variable change comme on le souhaite, nous pouvons générer le son (« ring tone at » - ton de sonnerie à) du menu « MUSIC » et aussi afficher la fréquence sur l'écran (« show number » - afficher le nombre) du menu « DISPLAY ».



Une dernière chose ! Le son peut être particulièrement dérangeant, alors coupons-le lorsque le bouton left (gauche) est pressé.



Voici le programme complet :



SERVOMOTEURS

Un servomoteur (ou servo) est un moteur auquel on a ajouté un petit circuit interne permettant de contrôler le servo en utilisant des impulsions électriques.

Un servo possède trois connections électriques qui peuvent être directement reliées au BrainPad. Malheureusement, différents servos utilisent souvent des fils de couleurs différentes. Pour identifier correctement la connexion à effectuer, ignorez toujours le fil du milieu et regardez les fils aux extrémités. Le fil ayant la couleur la plus claire est celui qui doit être relié à côté du symbole ~ imprimé sur le BrainPad. C'est d'ailleurs le fil du signal de commande.

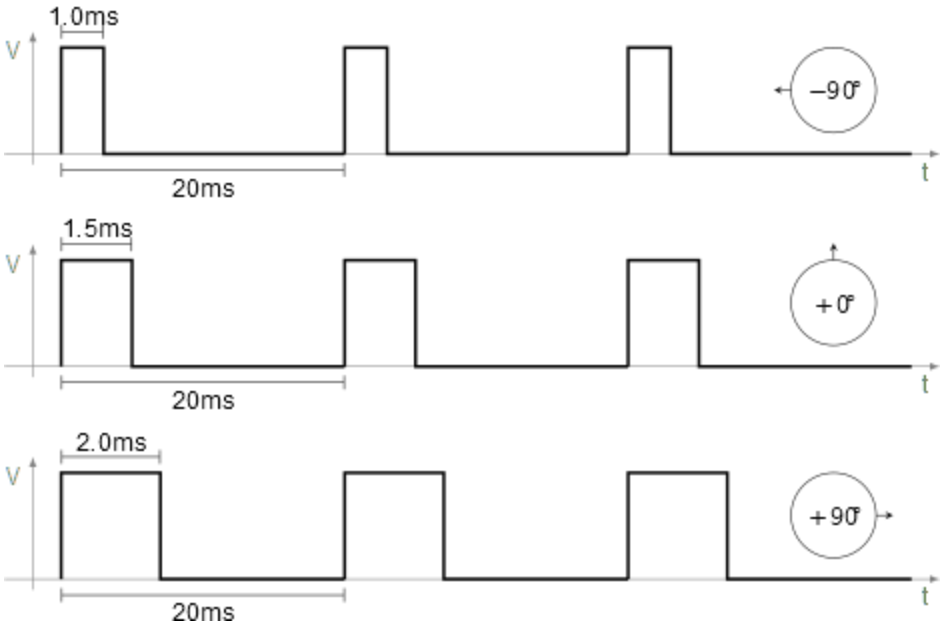
Les servomoteurs sont souvent utilisés dans les modèles télécommandés et peuvent être facilement trouvés sur Internet ou dans des magasins de modèles réduits. Ces servos obtiennent l'énergie qui leur est nécessaire du BrainPad, qui lui tire son énergie de votre PC ou de la batterie. C'est pourquoi, nous vous recommandons d'utiliser de petits servos, appelés encore micros servos. Certains des plus gros servos requièrent tellement d'énergie qu'ils ne peuvent pas fonctionner sans des alimentations supplémentaires.

Les servomoteurs existent sous la forme de servos à rotation continue ou de servos asservis en position. Bien qu'ils se ressemblent, un servo asservi en position tourne jusqu'à une position donnée puis reste à cette position jusqu'à ce que vous lui donniez l'ordre de se positionner ailleurs. Un servo à rotation continue va tourner continuellement dans une direction jusqu'à ce qu'il reçoive l'ordre de s'arrêter ou de changer de direction.

Si vous tournez à la main l'axe d'un servo asservi en position non alimenté, vous ne pourrez effectuer environ qu'un demi-tour quel que soit le sens avant d'être bloqué. Lorsque vous faites tourner l'axe d'un servo à rotation continue, il ne s'arrête pas. À cause de la présence d'un système d'engrenage interne, il vous faut appliquer un certain effort pour réussir à faire tourner l'axe, quel que soit le type de servo.

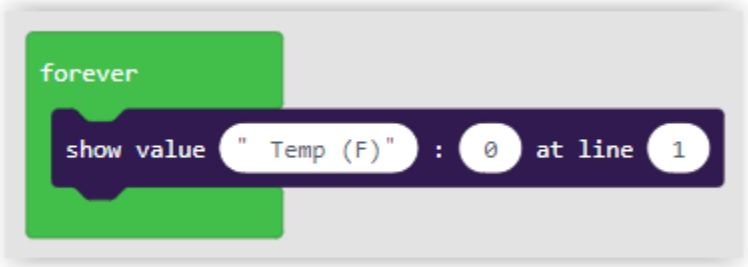
SERVOMOTEURS ASSERVIS EN POSITION

Comme indiqué précédemment, un servo asservi en position tourne jusqu'à une position (angle) que vous lui donnez et y reste jusqu'à ce que vous lui en donniez une autre. Si vous tournez l'axe d'un servo à la main et qu'il s'arrête, c'est un servo asservi en position. Une impulsion envoyée par le BrainPad indique au servo la position (l'angle) où il doit se positionner.

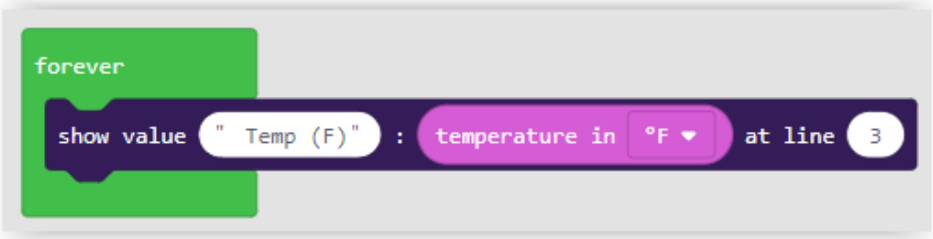


Il y a plusieurs façons d'utiliser un servo asservi en position. Dans cet exemple, nous allons créer un grand thermomètre analogique. Le code utilisé est simple et n'utilise même pas de variable !

Dans une boucle « forever », ajoutez un bloc « show value » (vous le trouverez dans le menu « DISPLAY »). Dans le premier ovale du bloc « show value », écrivez « Temp (F) » avec des parenthèses comme ci-dessous :



Depuis le menu « INPUT », sélectionnez le bloc "temperature in °C" et glissez-le dans le second oval. Notre thermomètre va afficher la température en Fahrenheit, changez-le "°C" en "°F", mais vous pouvez utiliser Celsius si vous préférez. Maintenant, modifiez le troisième oval en « 3 ». La température sera affichée sur la troisième ligne de l'écran du BrainPad. Vos blocs devraient ressembler à ce qui suit :

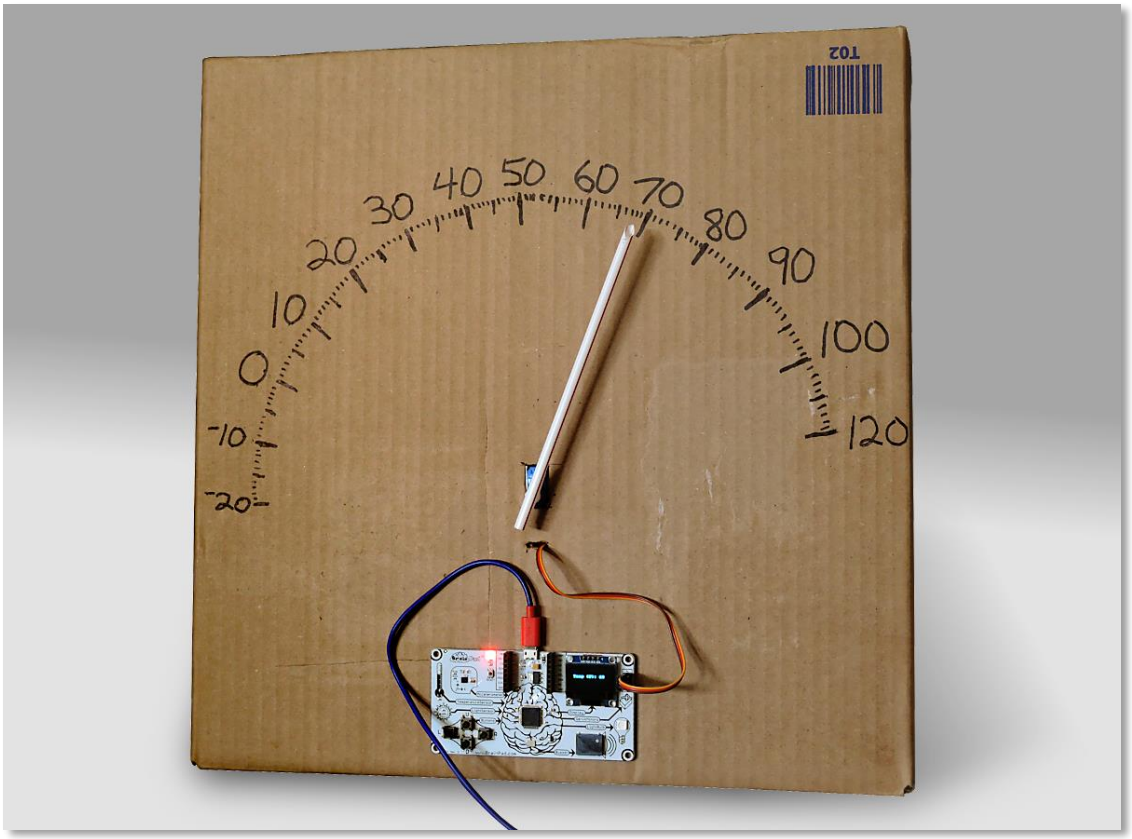


Jusque là, nous avons réalisé un thermomètre qui affiche la température sur l'écran du BrainPad. Maintenant, nous devons convertir cette température en un angle et la transmettre au servo pour déplacer l'aiguille du thermomètre. Faisons en sorte que notre thermomètre mesure la température entre -20° Fahrenheit à 120° Farhenheit.

Nous devons convertir la température (comprise entre -20 et 120) en un angle (compris entre 0 et 180). Si vous n'êtes pas fort en mathématiques, n'ayez pas peur ! MakeCode a un bloc de conversion qui va faire le calcul pour vous. Il se trouve dans le menu « MATH ». Nous pouvons terminer notre programme par l'ajout d'une ligne supplémentaire :

Vous remarquerez que l'angle diminue de la valeur 180 à la valeur 0 dans le bloc de conversion. Ne faudrait-il pas faire le contraire ? L'explication est la suivante : lorsque le servo est réglé à 0 degré, le moteur tourne vers la droite et, lorsque le servo est réglé sur 180 degrés, il tourne vers la gauche. Convertir la température de -20° Fahrenheit en un angle égal à 0 degré pour le servo aurait nécessité que le thermomètre soit lu à l'envers.

Voici le thermomètre terminé réalisé avec du carton et une paille collée sur le servo moteur.

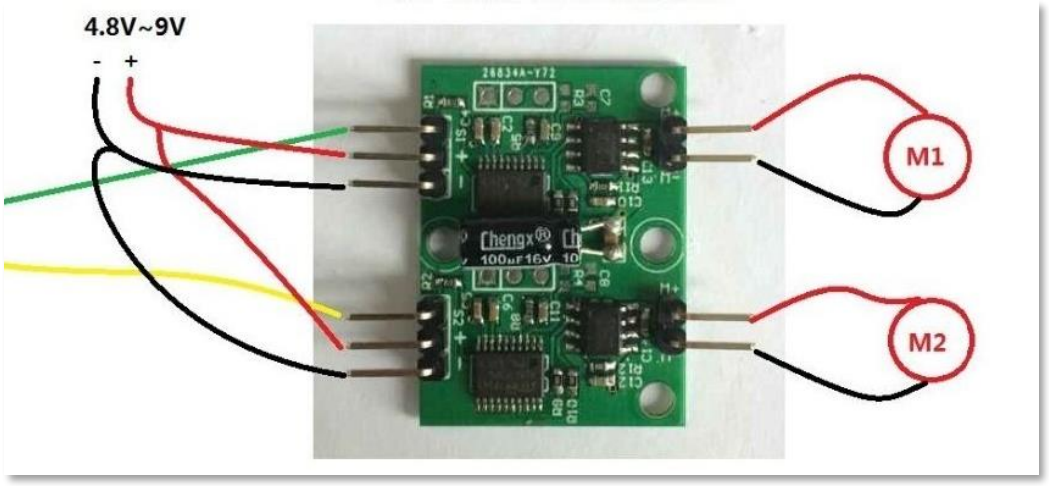


SERVOMOTEUR A ROTATION CONTINUE

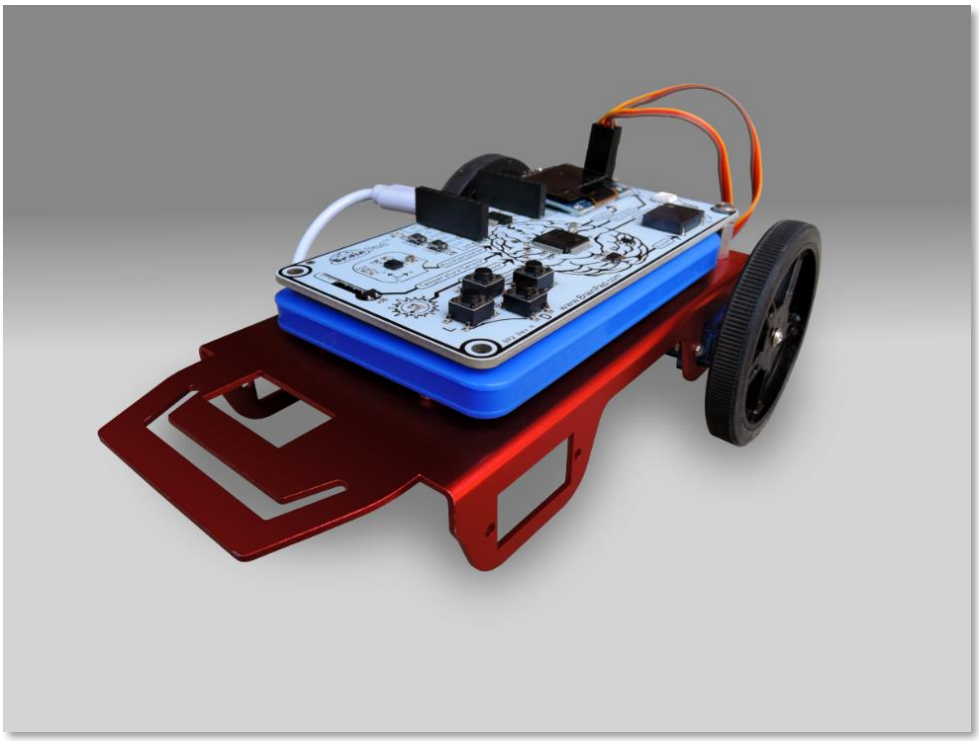
L'autre type de servomoteur est celui à rotation continue, qui comme son nom l'indique, peut tourner continuellement. Dans ce cas, l'impulsion va déterminer la vitesse et le sens de rotation du moteur. Le BrainPad peut commander deux moteurs. En contrôlant la direction est la vitesse de deux servos à rotation continue, vous allez pouvoir déplacer un robot très facilement. En cherchant sur Amazon.com, et en tapant « continuous micro servo with wheels », on peut trouver facilement ce dont nous avons besoin.

Nous aurions pu monter cela sur un boitier, mais nous pouvons utiliser ce châssis de FEETECH.

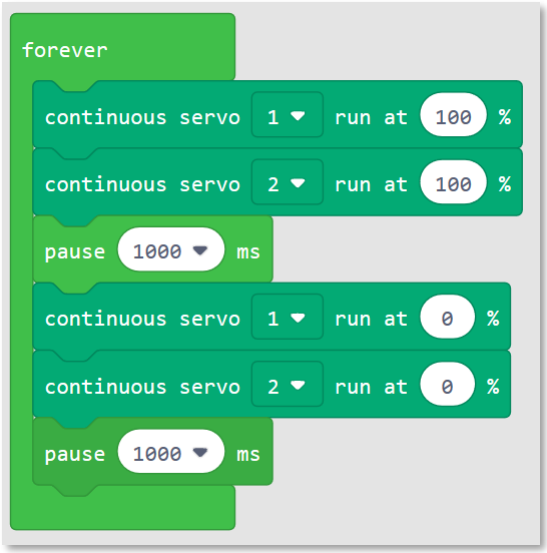
C’est là que les choses se compliquent un peu ! Le robot est fourni avec des roues et des moteurs. Les moteurs ressemblent à des servos, mais ce n’en sont pas, il s’agit de moteurs classiques. Nous pouvons nous en apercevoir, car ils n’ont que deux fils au lieu de trois. Le kit contient également un petit circuit, qui finalement va convertir ces moteurs classiques en servomoteurs !



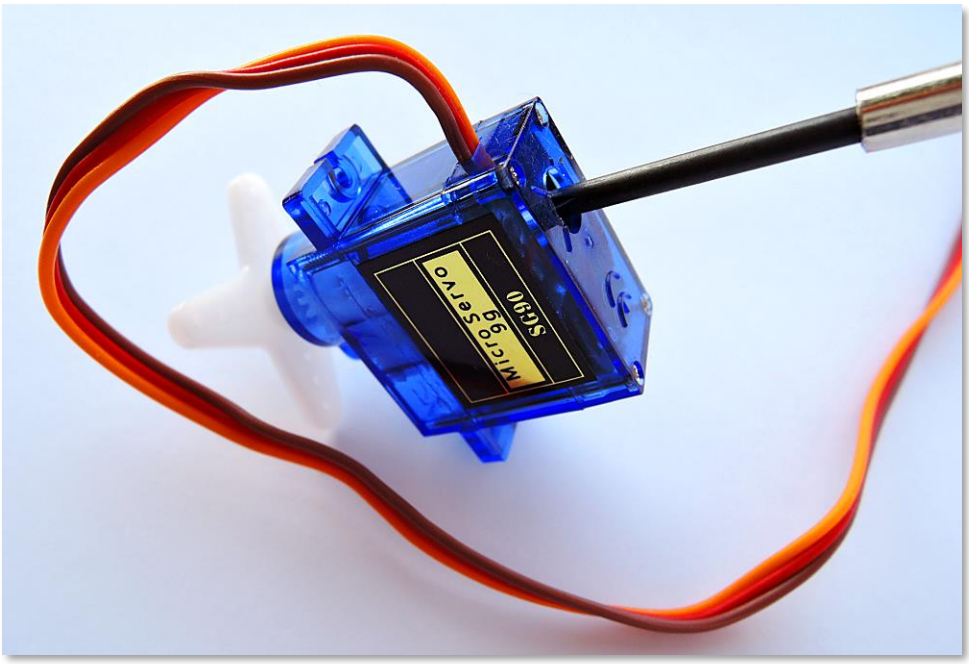
Ce serait faisable, mais bien trop compliqué pour la plupart des classes de collège, nous mettrons donc de côté ce circuit et ces moteurs et utiliserons de véritables servomoteurs. Le BrainPad tient parfaitement sur ce châssis comme s'il avait été fait pour lui ! Reste à trouver une batterie plate de téléphone qui tienne sous le BrainPad.



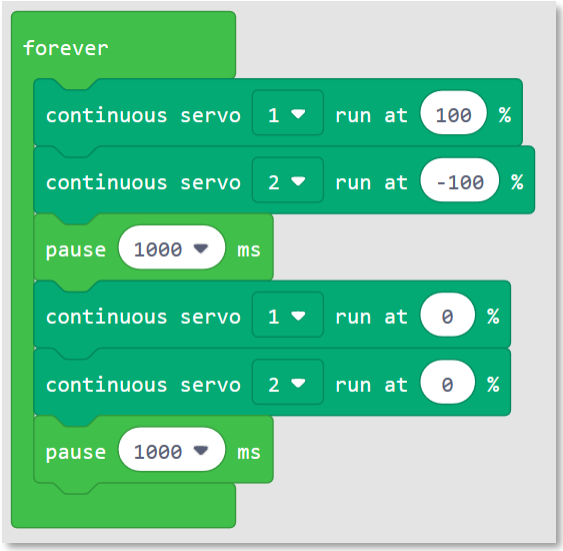
Ce robot est enfin prêt à danser ! Pour aller en avant, les deux moteurs doivent tourner vers l'avant. Cependant, comme un moteur est placé à gauche et l'autre à droite, ils doivent tourner dans le sens inverse l'un de l'autre pour que les roues aillent dans la même direction. Cela peut paraître confus, mais en essayant, vous comprendrez mieux. Ce code fait tourner les deux servos vers l'avant à pleine vitesse pendant une seconde, puis arrête les deux moteurs pendant une seconde, puis recommence.



Si les moteurs continuent de tourner lorsque la vitesse est à zéro, c'est qu'ils ont besoin d'être calibrés. La plupart des servos ont un système d'ajustement. Vous aurez besoin d'un petit tournevis pour effectuer cette opération.



Faisons avancer notre robot. Voyez-vous à présent qu'un des servos va dans la mauvaise direction ? Changez le réglage d'un des servos pour qu'il aille à pleine vitesse dans le sens opposé.



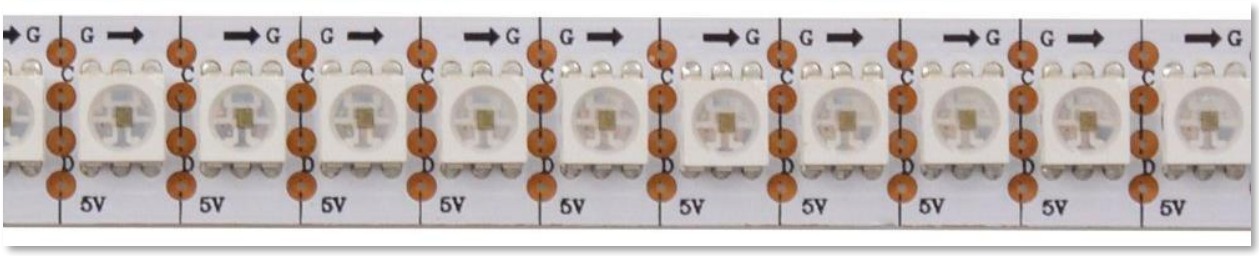
Ceci est le point de départ de notre projet de robot « Sun Seeker ». Vous pouvez le trouver ici : https://docs.brainpad.com/projects/sun-seeker.html. Prenez le temps de regarder la vidéo. Le robot utilise le capteur de lumière pour détecter s'il est au soleil ou à l'ombre. S'il est à l'ombre, le robot va tourner jusqu'à ce qu'il trouve un coin au soleil où il se déplacera.

GUIRLANDES DE NOËL

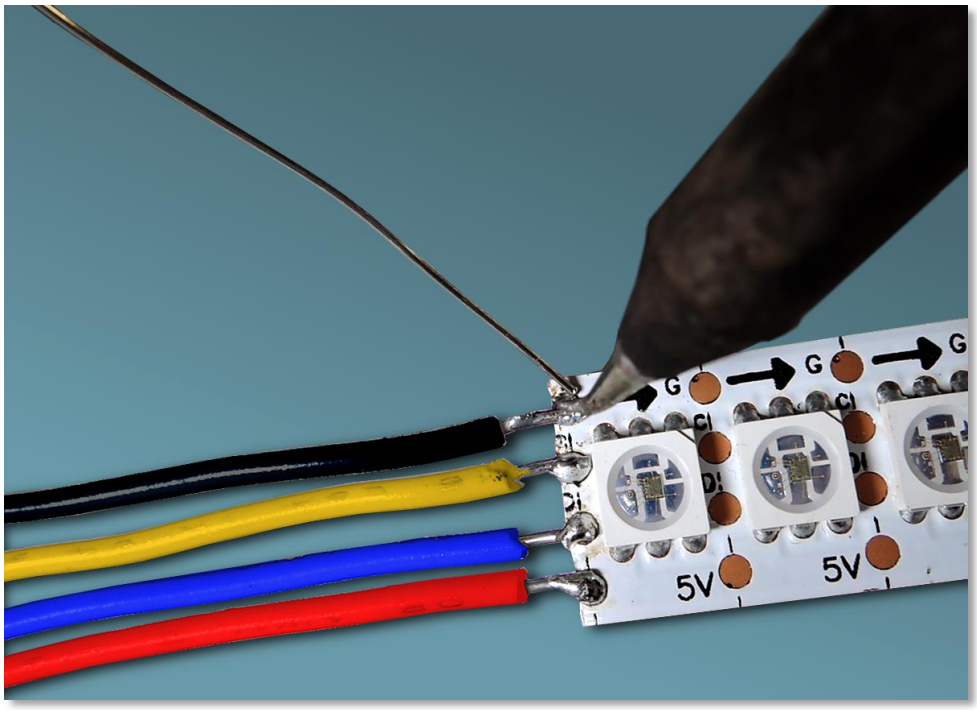
Il y a plein de choses à faire avec les circuits embarqués sur le BrainPad, mais dans certains cas, vous voudrez peut-être connecter le BrainPad à des composants extérieurs. Bien que ce soit au-delà des objectifs de ce livre, voici une rapide introduction qui vous donnera une idée de ce qu'il est possible de faire.

Pour ce projet nous utiliserons des bandes de DEL (Diode ElectroLuminescente) adressables. Ces bandes de DEL réalisent une ligne de lumières pouvant être contrôlées individuellement. Elles peuvent être réglées avec n'importe quelle combinaison de couleurs et de luminosité avec seulement deux fils. Ces fils envoient des impulsions aux DELs pour les commander à votre convenance.

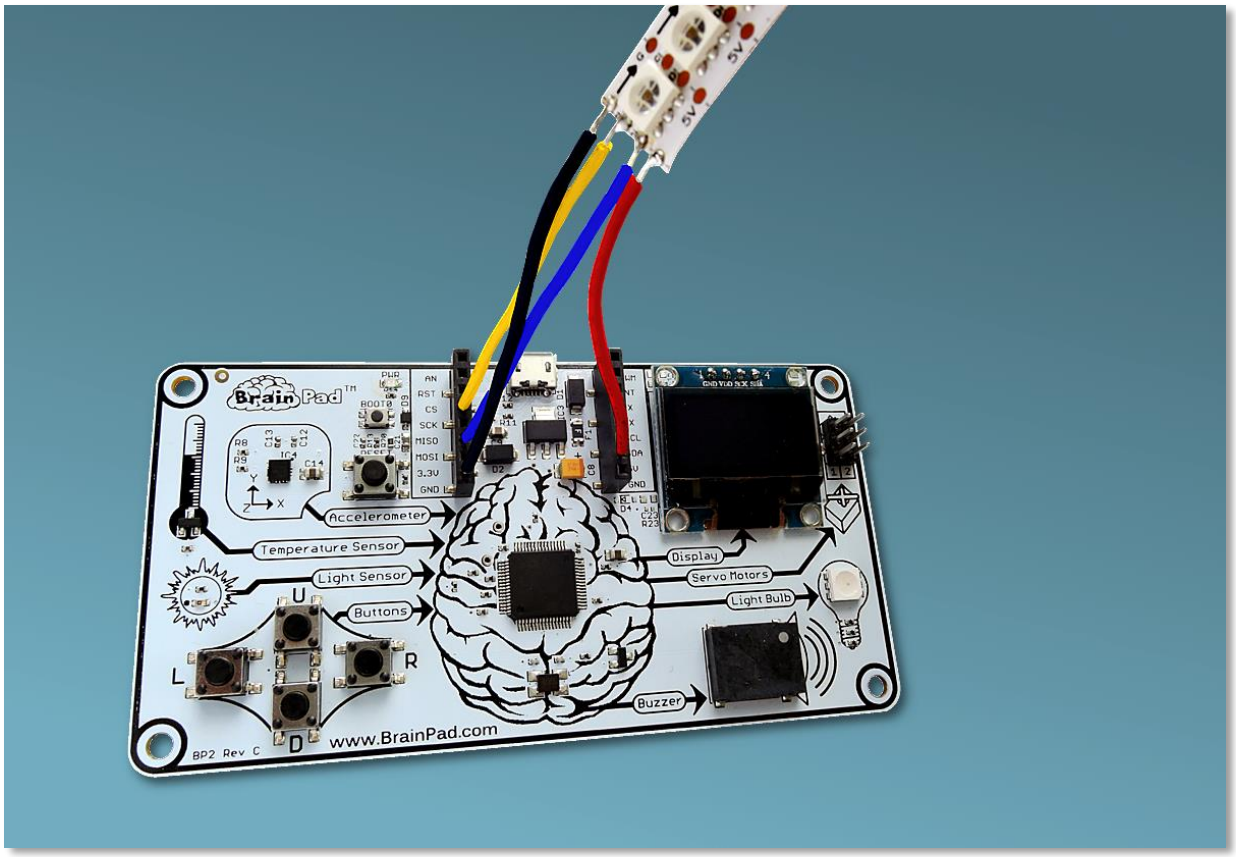
Il vous faudra des bandes avec un contrôleur APA102. Ce contrôleur utilise le bus SPI disponible sur le BrainPad. Ces bandes sont alimentées par différentes tensions. Nous aurons besoin de celles sous 5V, car nous avons du 5V à notre disposition.



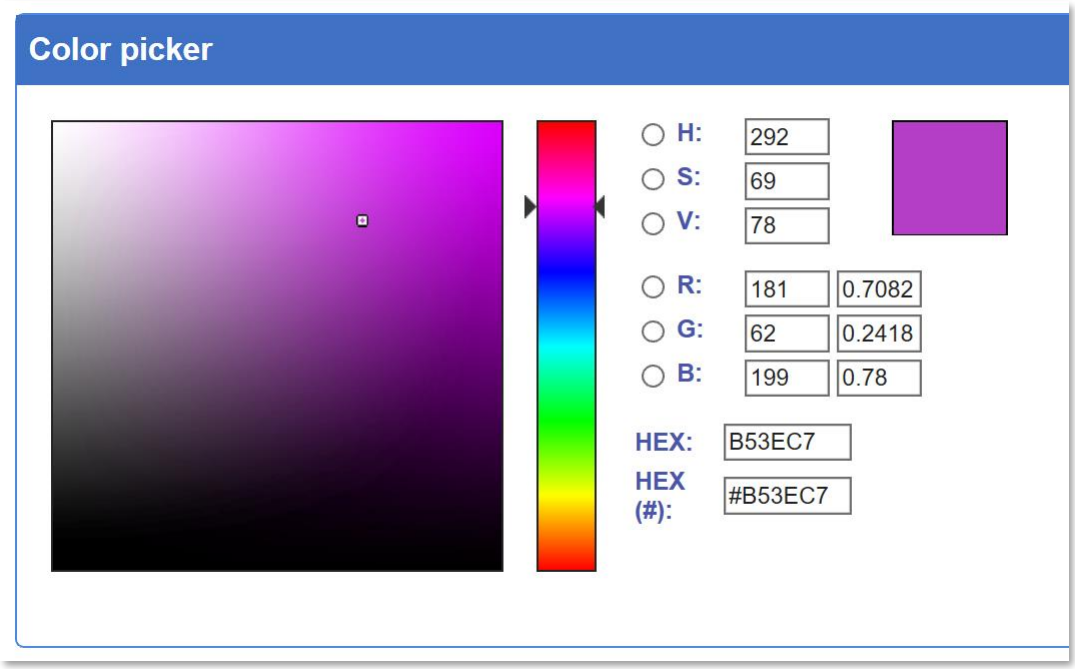
Ces bandes peuvent être découpées à n'importe quelle longueur. Pour éviter que notre configuration nécessite trop d'énergie, nous allons la couper pour conserver 25 DELs. Maintenant, soudons quelques fils. Les flèches sur les bandes doivent pointer à l'opposé du côté où nous souderons les fils. Les quatre fils sont G-Ground (Masse), C-Clock (Horloge), D-Data (Données) et le 5V.



La masse (Ground (GND)) et le 5V sont clairement indiqués sur le connecteur d'extension du BrainPad. Le C-Clock doit être relié à SCK sur le BrainPad et D-Data à MOSI.



Chaque DEL a en réalité trois DELs internes. Elles sont rouges, bleues et vertes. En contrôlant la luminosité de chacune de ces couleurs primaires, vous pouvez obtenir n'importe quelle couleur. Il existe plusieurs outils en ligne pour vous aider dans le réglage des couleurs. Un bon exemple est http://www.rgbtool.com. Ce qui vous intéresse ce sont les valeurs R (Red – Rouge), G (Green – Verte) et B (Blue – Bleue).



Les commandes sont envoyées aux DELs en utilisant le bus SPI du BrainPad. Pour écrire une séquence de couleurs à envoyer à la bande de DELs, il faut commencer par envoyer une trame de début. Une trame de début est simplement composée de quatre zéros.

Après la trame de début, chaque DEL nécessite quatre nombres pour la contrôler. Le premier nombre règle la luminosité globale de la chaine de DELs. Nous ne voulons pas trop rentrer dans les détails, c'est pourquoi nous règlerons toujours ce nombre à 255, ce qui correspond à la luminosité la plus forte.

Après ce premier nombre, les trois nombres restants sont compris entre 0 et 255, et définissent la luminosité du bleu, du vert et du rouge pour chaque DEL. Pour allumer une DEL en bleu, vous enverrez 255,0,0. Pour éteindre une DEL, vous enverrez 0,0,0. Pour allumer une DEL en blanc, vous enverrez 255,255,255.

Les DELs dans la bande sont simplement chaînées ensemble. La première couleur est envoyée à la première DEL, la seconde couleur à la seconde DEL, et ainsi de suite. Les DELs vont continuer à briller jusqu'à ce qu'une nouvelle trame de début soit envoyée (quatre zéros). Après la trame de début, la séquence recommence par la première DEL.

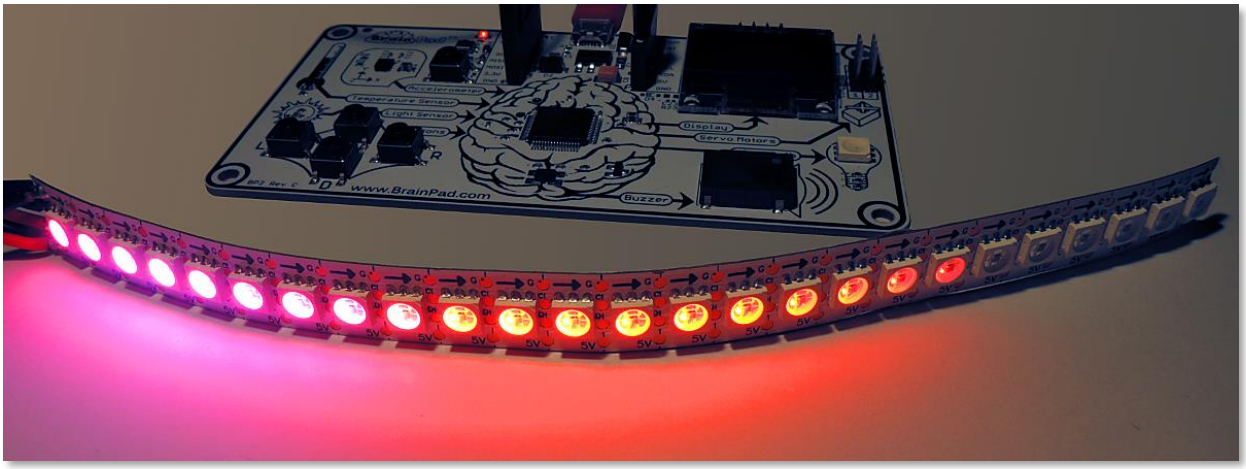
À cause du nombre de blocs nécessaires, il est plus facile de basculer en mode JavaScript et de copier, coller le code suivant dans l'éditeur de Microsoft MakeCode :

```
let center = 12

let startRed = 0
```

```
let startGreen = 0
let startBlue = 0
let red = 0
let green = 0
let blue = 0
let data: Buffer = pins.createBuffer(4)
let dummy: Buffer = pins.createBuffer(4)

pins.spiFrequency(1000000)

function SendStart() {
    data[0] = 0
    data[1] = 0
    data[2] = 0
    data[3] = 0
    pins.spiTransfer(data, dummy)
}

function SendRGB(red: number, green: number, blue: number) {
    data[0] = 255
    data[1] = blue
    data[2] = green
    data[3] = red

    pins.spiTransfer(data, dummy)
}

forever(function () {
    startRed = Math.constrain(startRed + Math.randomRange(-20, 20), 0, 255)
    startGreen = Math.constrain(startGreen + Math.randomRange(-20, 20), 0, 255)
    startBlue = Math.constrain(startBlue + Math.randomRange(-20, 20), 0, 255)

    SendStart()
```

```
    for (let index = 0; index <= 24; index = index + 1) {
        red = Math.constrain(startRed - Math.abs(index - center) * 16, 0, 255)
        green = Math.constrain(startGreen - Math.abs(index - center) * 16, 0, 255)
        blue = Math.constrain(startBlue - Math.abs(index - center) * 16, 0, 255)

        SendRGB(red, green, blue)
    }
})
```

Si vous faites tout correctement, vous devriez voir un spectacle de lumière colorée aléatoirement. Essayez de modifier le code ci-dessus pour voir l'impact sur l'éclairage.



CONCLUSION

Microsoft MakeCode et le BrainPad forment une bonne équipe pour apprendre la programmation. Il est facile de tester le code, même sans BrainPad, avec le simulateur. Vous pouvez ensuite charger le programme dans le BrainPad en copiant tout simplement le fichier téléchargé dans sa fenêtre.

Dans le prochain chapitre, nous allons aller plus loin et coder en utilisant C# ou Visual Basic dans Microsoft Visual Studio. "Pour aller plus loin" est une partie optionnelle, mais grandement recommandée une fois que vous vous sentez à l'aise avec MakeCode. Dans la plupart des cas, utiliser MakeCode est amplement suffisant.

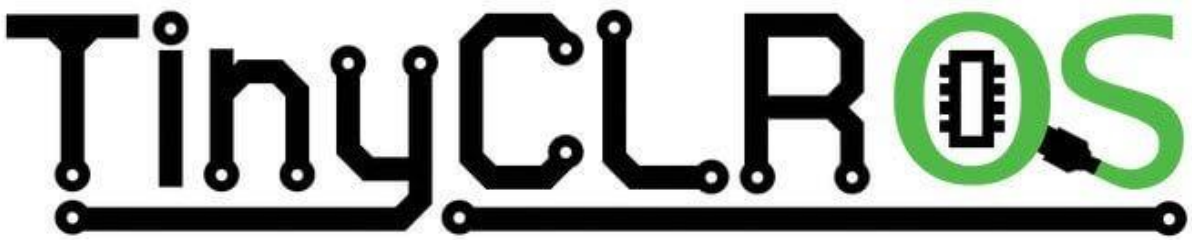
POUR ALLER PLUS LOIN - GO BEYOND

Vous êtes ici, car vous souhaitez apprendre à programmer comme un pro. Dans cette partie, vous utiliserez Microsoft Visual Studio pour programmer en C# ou en Visual Basic. Vous disposerez de possibilités de débogage dans un environnement de développement fonctionnel.

L'utilisation de Microsoft MakeCode n'est pas une condition préalable, mais elle est fortement recommandée. Après tout, il y a une raison si nous l'avons nommé **Start Making**. Quoi qu'il en soit, mettez votre ceinture de sécurité puisque vous êtes sur le point d'entreprendre une véritable carrière de codeur professionnel qui pourrait devenir votre futur métier !

TINYCLR OS™

Pour que BrainPad fonctionne avec .NET en C# et Visual Basic, vous aurez besoin de TinyCLR OS. C'est dans ce minuscule système d'exploitation que tient toute la magie. Il interprète les assemblys .NET compilés et contient des milliers de services tels que le threading, la gestion de la mémoire et le débogage. C'est la même chose que le .NET utilisé pour développer des logiciels sur les téléphones ou les ordinateurs. Cependant, TinyCLR OS est conçu pour fonctionner sur de petits ordinateurs comme le BrainPad, il s'agit donc d'un sous-ensemble de la norme .NET.



VISUAL STUDIO

En théorie, n'importe quel compilateur .NET peut être utilisé avec TinyCLR OS ; cependant, TinyCLR est uniquement pris en charge par Microsoft Visual Studio. L'édition communautaire (community) de Visual Studio est un environnement de programmation complet considéré comme l'un des meilleurs (sinon le meilleur) disponible sur le marché – et, cerise sur le gateau, c'est gratuit ! Pour installer Visual Studio, vous aurez besoin d'un PC récent avec Windows. La version Windows 10 est recommandée.

Notez que vous pourez réutiliser Microsoft Visual Studio et les compétences acquises lors de l'apprentissage de la programmation du BrainPad pour commencer à écrire des applications pour d'autres appareils, comme les smartphones, les PC et même les consoles de jeu Microsoft Xbox !

INSTALLATION DU SYSTEME

L'édition Microsoft Visual Studio Community est gratuite, mais ce n'est pas un jouet et ce n'est pas un petit logiciel. Vous aurez besoin de temps pour le configurer et le prendre en main. Des instructions complètes sont disponibles sur le site Web de la documentation du BrainPad à l'adresse :

https://docs.brainpad.com/go-beyond/system-setup.html.

En résumé, vous devrez télécharger et installer :

1. L'édition Microsoft Visual Studio Community. N'oubliez pas de sélectionner l'option "Développement de bureau .NET" lors de l'installation.
2. Le système de projet TinyCLR OS de GHI Electronics.

Si vous êtes perdu, vous pouvez toujours demander de l'aide sur le forum :
https://forums.ghielectronics.com/c/brainpad.

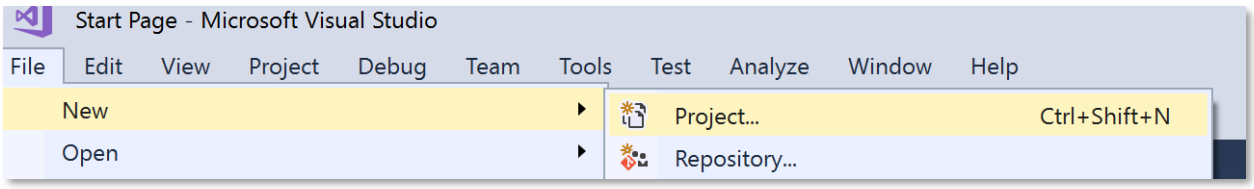
Maintenant que le PC est configuré, vous devez charger TinyCLR OS sur le BrainPad. Téléchargez le fichier du firmware TinyCLR OS à partir d'ici https://docs.brainpad.com/resources/downloads.html puis chargez-le sur le BrainPad comme si vous chargiez un fichier créé avec Microsoft MakeCode :

1. Maintenez le bouton de réinitialisation (RESET) enfoncé pendant environ trois secondes. Le Light Bulb devient vert.
2. Le PC détectera un périphérique de stockage et ouvrira une fenêtre.
3. Enregistrez le micrologiciel du système d'exploitation TinyCLR sur le périphérique de stockage BrainPad. Vous pouvez également faire glisser le fichier ou copier et coller le fichier dans la fenêtre BrainPad.

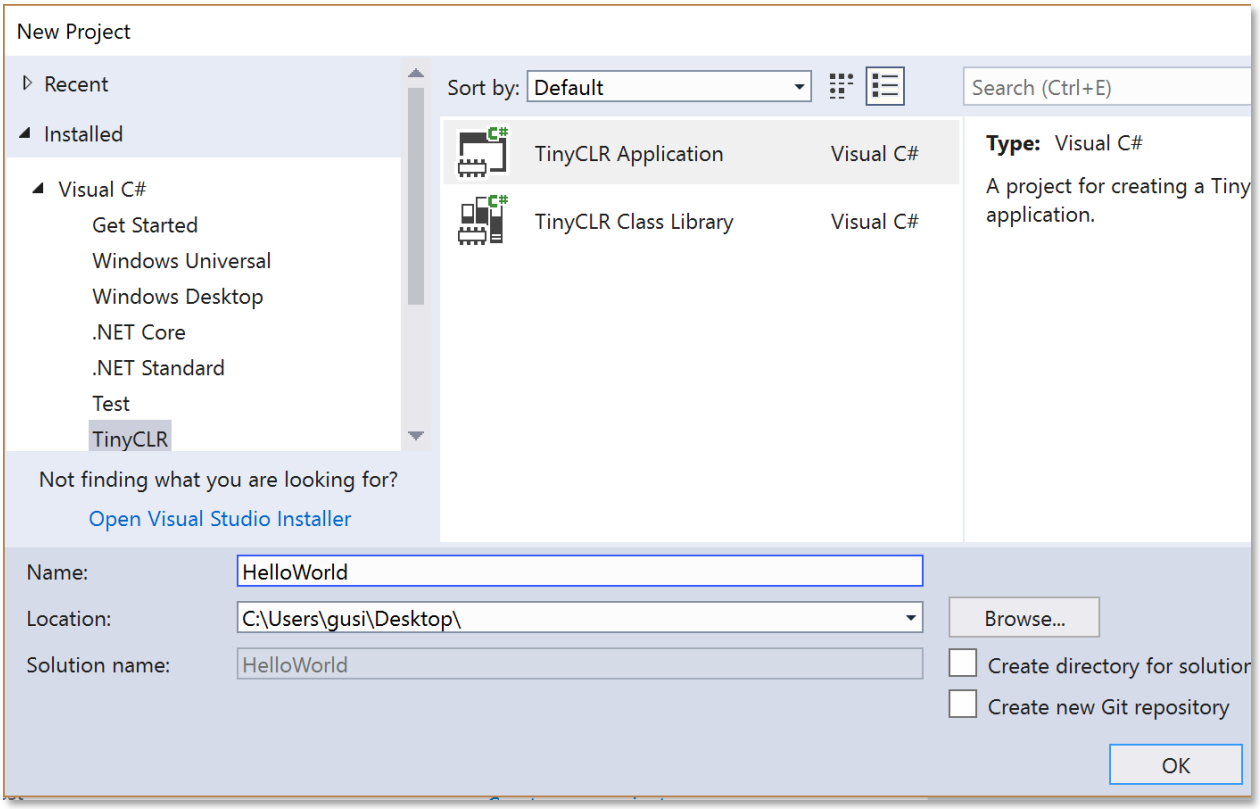
HELLO WORLD

Lors de la programmation d'un nouvel appareil, ou lors de l'utilisation de nouveaux outils de programmation, il est courant de démarrer avec un programme très simple pour s'assurer que tout fonctionne correctement. Ce programme est généralement appelé "Hello World" ("Bonjour le monde").

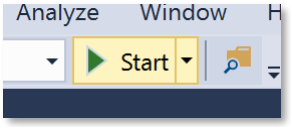
Ouvrez Microsoft Visual Studio et démarrez un nouveau projet. Dans le menu "Fichier", sélectionnez "Nouveau" puis "Projet" (Fichier (File)> Nouveau (New)> Projet (Project)).



Il devrait y avoir une option TinyCLR sous Visual C #. Si ce n'est pas le cas, vous n'avez pas installé le système de projet TinyCLR OS. Ici, j'ai nommé le projet "HelloWorld" et je l'ai placé sur mon bureau.



Vous pouvez maintenant charger ce programme vide sur votre BrainPad. Ça ne fera pas grand-chose, mais ça va vous permettre de vérifier que tout fonctionne comme prévu. Allez-y puis connectez le BrainPad et cliquez sur Démarrer.

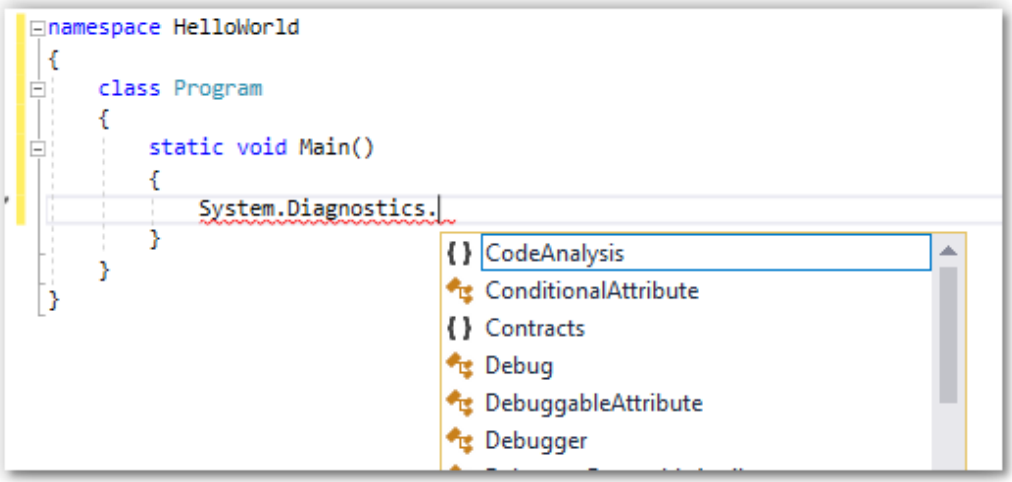


Observez l'activité affichée dans la fenêtre de sortie. Si cette fenêtre n'est pas visible, appuyez sur Ctrl + Alt + O sur votre clavier ou sélectionnez "Sortie" dans le menu "Affichage" (Affichage (View) > Sortie (Output)).

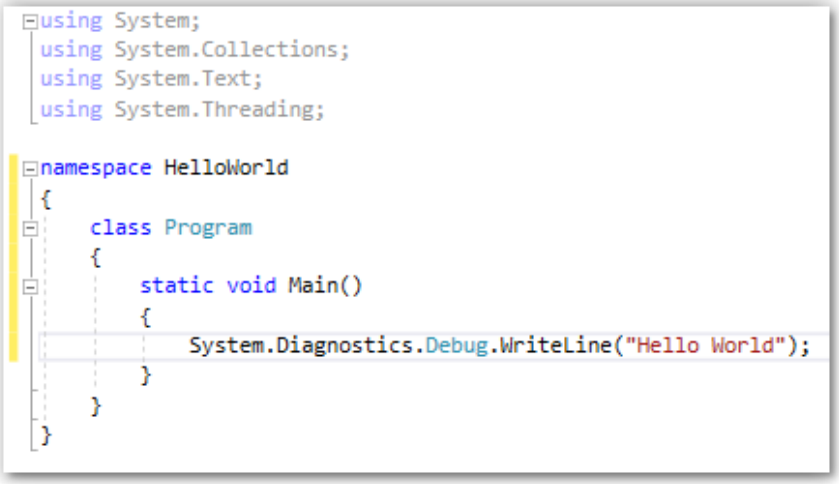
Maintenant, nous sommes prêts pour que BrainPad dise "Hello World". Nous allons d'abord afficher "Hello World" dans la fenêtre de sortie de Visual Studio. Puis, nous l'afficherons sur l'écran du BrainPad. Ajoutez la ligne ci-dessous au programme, juste sous l'accolade ouvrante ({) après Main ().

```
System.Diagnostics.Debug.WriteLine("Hello World");
```

Notez comment Visual Studio vous suggère automatiquement des mots.



Une fois terminé, le code devrait ressembler à ceci :



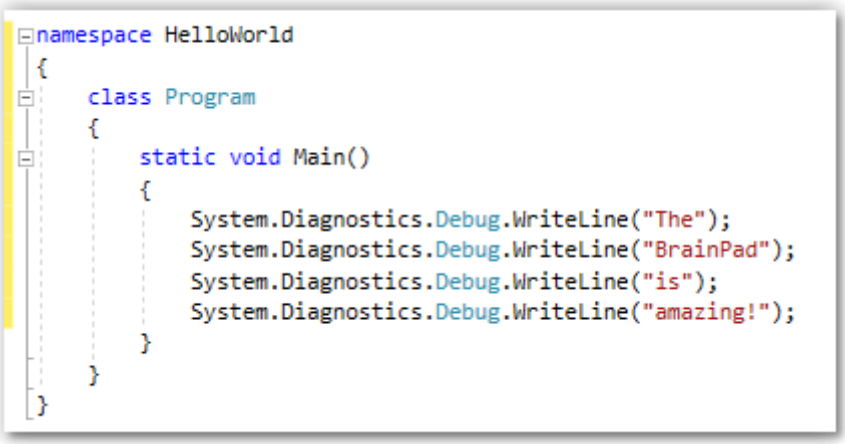
Démarrez le programme et observez à nouveau la fenêtre de sortie.

```
The debugging target runtime is loading the application assemblies and starti
Ready.

Done.

Waiting for debug commands...

'GHIElectronics.TinyCLR.VisualStudio.dll' (Managed): Loaded 'C:\Users\gusi\De
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
Hello World
The thread '<No Name>' (0x1) has exited with code 0 (0x0).
The program '[3] TinyCLR application: Managed' has exited with code 0 (0x0).
```

Avez-vous trouvé "Hello World" ? C'était un bon début, mais pas très excitant. Essayons d'imprimer plusieurs lignes. Remplacez la ligne que nous avons ajoutée auparavant avec ces quatre lignes.

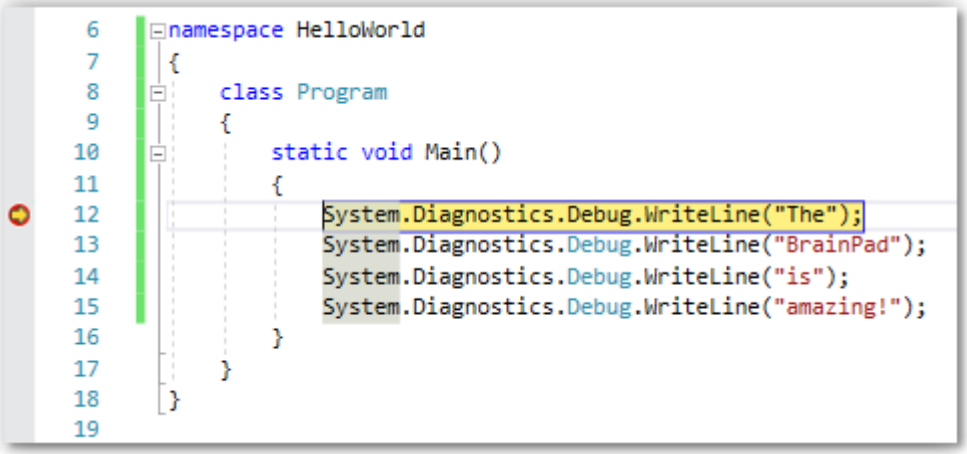
```
System.Diagnostics.Debug.WriteLine("The");
System.Diagnostics.Debug.WriteLine("BrainPad");
System.Diagnostics.Debug.WriteLine("is");
System.Diagnostics.Debug.WriteLine("amazing!");
```

Placez le curseur sur la première ligne et appuyez sur la touche de fonction F9. Cela va ajouter un point d'arrêt. Vous pouvez faire la même chose dans le menu Déboguer (Debug).



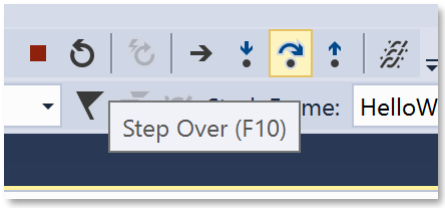
```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            System.Diagnostics.Debug.WriteLine("The");
            System.Diagnostics.Debug.WriteLine("BrainPad");
            System.Diagnostics.Debug.WriteLine("is");
            System.Diagnostics.Debug.WriteLine("amazing!");
        }
    }
}
```

Démarrez le programme comme précédemment. Dès que le point d'arrêt est atteint, l'exécution s'arrête sur cette ligne.



```
6   namespace HelloWorld
7   {
8       class Program
9       {
10          static void Main()
11          {
12              System.Diagnostics.Debug.WriteLine("The");
13              System.Diagnostics.Debug.WriteLine("BrainPad");
14              System.Diagnostics.Debug.WriteLine("is");
15              System.Diagnostics.Debug.WriteLine("amazing!");
16          }
17      }
18  }
19
```

Le programme s'arrêtera juste avant que cette ligne soit exécutée. Appuyez sur la touche F10 ou cliquez sur le bouton « Pas à pas principal » (Step Over).



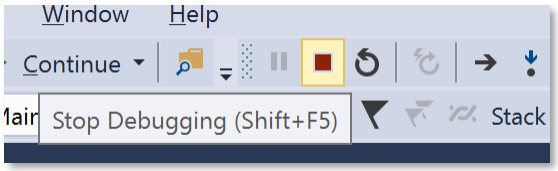
Cela permettra au programme d'exécuter une ligne de code, ou une étape. Puisque nous sommes sur la ligne qui affiche "The", vous la verrez dans la fenêtre de sortie. Appuyez sur (F10) tout en observant la fenêtre de sortie. C'est ce qu'on appelle le mode pas-à-pas, extrêmement utile lorsque vous essayez de déboguer vos programmes pour comprendre pourquoi quelque chose ne fonctionne pas comme prévu.

DES PROGRAMMES SANS FIN

Il est logique qu'un programme se ferme ou se termine sur un PC ou un téléphone. Cependant, dans les périphériques plus petits, appelés périphériques embarqués, les programmes ne se terminent généralement pas. Pensez à un four à micro-ondes par exemple. Il y a un petit "cerveau" à l'intérieur du four qui surveille l'appui sur les touches et fait fonctionner l'horloge. Sauf si vous débranchez le micro-ondes, ce programme est toujours en cours d'exécution. Cela s'appelle une boucle infinie et c'est souvent utilisé en programmation.

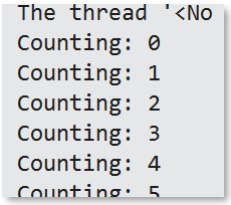
Créons un compteur fonctionnant "indéfiniment". Le BrainPad comptera une fois par seconde et ne s'arrêtera jamais. Voici à quoi devrait ressembler le code.

Notez que vous devez arrêter un programme en cours d'exécution avant de le modifier. Vous pouvez le faire en appuyant sur le bouton d'arrêt (Stop Debugging).

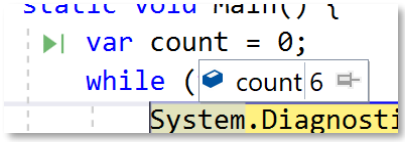



```
namespace HelloWorld {
    class Program {
        static void Main()
        {
            var count = 0;

            while (true)
            {
                System.Diagnostics.Debug.WriteLine("Counting: " + count);
                Thread.Sleep(1000); // Wait one second

                count = count + 1;
            }
        }
    }
}
```

La fenêtre de sortie affichera le compteur.

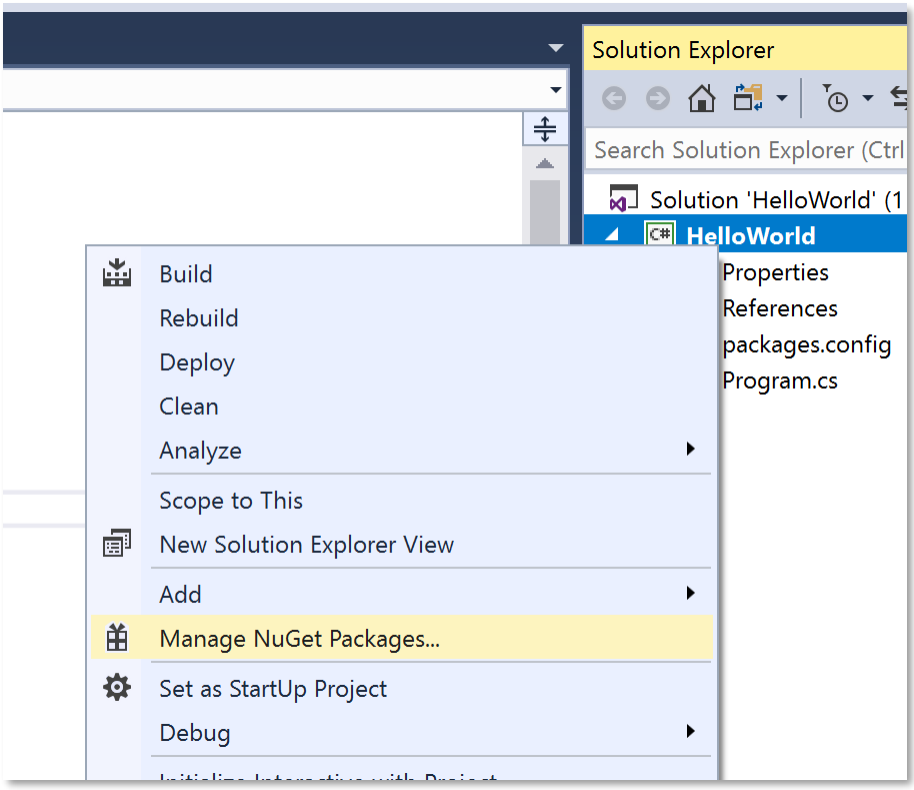


Pendant l'exécution du programme, insérez un point d'arrêt sur la ligne du compteur. À partir de là, vous pouvez parcourir le code comme nous l'avons fait auparavant. Maintenant, passez la souris sur notre variable "count", Visual Studio vous montrera la valeur actuelle de la variable. Lorsque j'ai arrêté le programme, elle avait la valeur 6. En passant, l'inspection des variables est une autre fonctionnalité très utile pour le débogage des programmes.

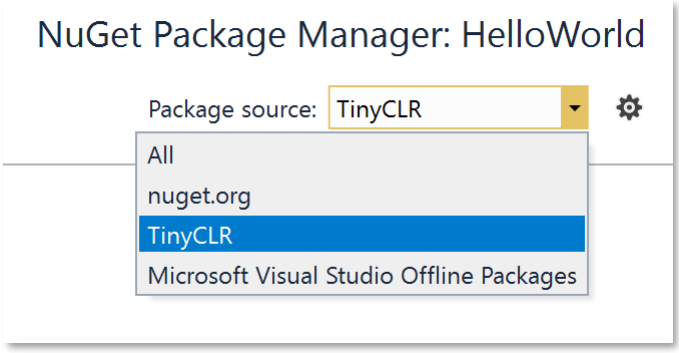


LES BIBLIOTHEQUES BRAINPAD

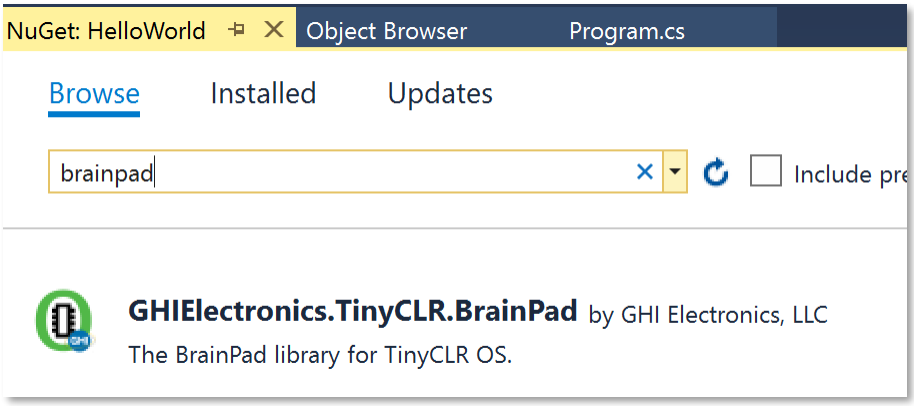
Jusqu'à présent, nous avons utilisé un système d'exploitation TinyCLR de base pour exécuter des programmes simples. Le BrainPad est livré avec des bibliothèques facilitant l'utilisation de toutes les entrées et sorties disponibles. Cliquez avec le bouton droit sur le projet dans la fenêtre de l'Explorateur de solutions, puis sélectionnez « Gérer les packages NuGet » (Manage Nuget Packages) ...



NuGet est un service en ligne hébergeant des bibliothèques. Vous pouvez également télécharger les bibliothèques et les héberger localement. Sélectionnez la provenance des bibliothèques. Par exemple, nous avons les bibliothèques hébergées localement dans un répertoire nommé TinyCLR.

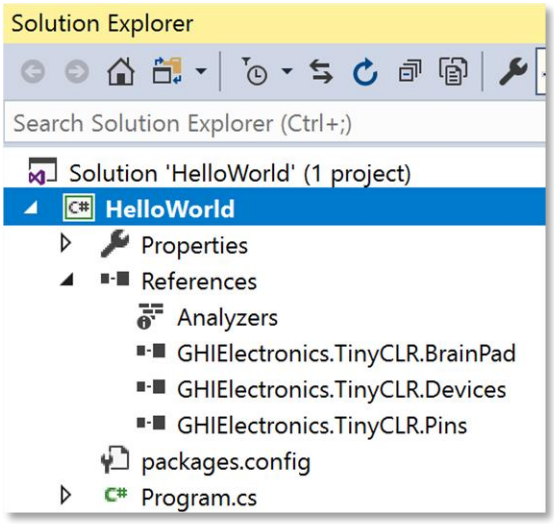


Sous « Parcourir » (Browse), entrez "brainpad" dans la barre de recherche.



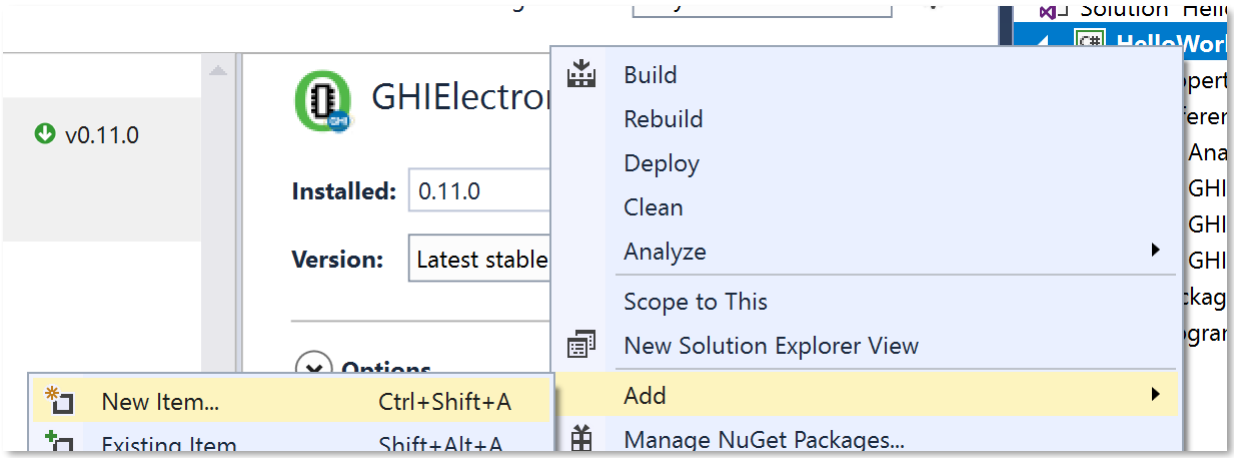
Cela fera apparaître la bibliothèque `GHIElectronics.TinyCLR.BrainPad`. Sélectionnez-là et cliquez sur installer (install).

Cette étape ajoutera les bibliothèques nécessaires sous Références (References) dans l'explorateur de solution.

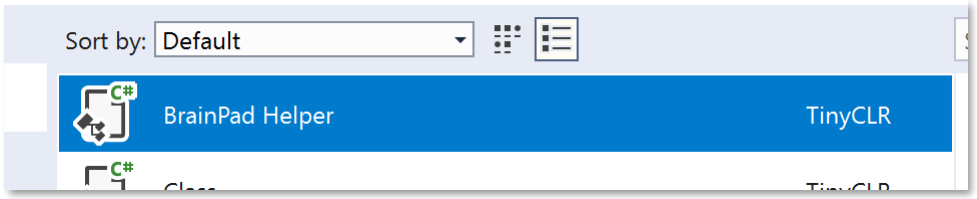


Pour faciliter l'ajout de bibliothèques à votre code, une classe d'aide est fournie. Faites un clic droit sur le projet, sélectionnez Ajouter (Add) > Nouvel élément (New Item). . .

Vous pouvez également utiliser le raccourci Ctrl + Maj + A.



Parmi les options disponibles, sélectionnez l'assistant BrainPad (BrainPad Helper) et cliquez sur le bouton Ajouter.



Le plaisir commence maintenant ! Revenez sur Program.cs, où nous avons précédemment ajouté le code du compteur. Changez le code pour afficher le compteur sur l'écran du BrainPad.

```
namespace HelloWorld
{
```

```
    class Program
    {
        static void Main()
        {
            var count = 0;

            while (true)
            {
                BrainPad.Display.DrawSmallText(0, 0, "Count: " + count);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(1);

                count = count + 1;
            }
        }
    }
}
```

La ligne qui affiche le compteur est similaire à la commande Debug.WriteLine que nous avons utilisée auparavant, sauf que cette fois-ci, elle permet un affichage sur l'écran du BrainPad. La deuxième ligne est nécessaire au rafraîchissement de l'écran. Les écrans sont beaucoup plus lents que les processeurs. Si nous rafraîchissions l'affichage chaque fois que nous dessinons, cela se déroulera très lentement. Au lieu de cela, le processeur écrit dans sa mémoire. Ce n'est que lorsque RefreshScreen est appelé que l'affichage sera actualisé / mis à jour. Pensez à un jeu vidéo où vous devez dessiner un navire, des obstacles, des ennemis ... etc. Vous allez placer tous ces dessins en mémoire, ce qui est rapide, et rafraîchir l'affichage seulement si c'est nécessaire.

La dernière ligne demande au Brainpad d'attendre une seconde.

BALLE REBONDISSANTE

Le texte contenu dans les bibliothèques BrainPad laisse à penser qu'il s'agit plus d'"Anglais" que de code. Ajoutez de la logique vous permet de créer des programmes incroyables. Ce livre n'est pas destiné à enseigner C # ou Visual Basic, mais cela ne veut pas dire que nous ne pouvons pas nous amuser un peu !

Revenons au programme original et dessinons un cercle de 5 pixels de diamètre. Nous allons le dessiner à l'emplacement 30,30. Sur les systèmes informatiques, les affichages commencent dans le coin supérieur gauche, à la position 0,0. Déplacer un objet vers la droite se fait en augmentant le premier nombre qui est l'abscisse x. 30,0 décrit un point à 30 pixels du bord gauche et tout en haut de l'écran.

```
namespace HelloWorld
{
    class Program
```

```
    {
        static void Main()
        {
            var x = 30;
            var y = 30;

            while (true)
            {
                BrainPad.Display.DrawCircle(x, y, 5);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(1);
            }
        }
    }
}
```

Et nous avons un cercle !

Augmentons la valeur des variables x et y chaque fois que le programme parcourt la boucle sans fin. En partant de la valeur 30, la variable prendra les valeurs 31, 32, 33, 34 ... etc. Les choses deviennent-elles plus excitantes ?

Le cercle s'est déplacé très lentement à cause du délai d'une seconde. Nous allons le changer en 0,01. Nous ajouterons également de la logique afin de faire rebondir le cercle c'est-à-dire la balle. Pour qu'elle rebondisse, nous devons utiliser de nouvelles variables, dx et dy. Celles-ci gèrent principalement sa direction. Nous vérifions ensuite si la balle est sur un bord. Si oui, nous inversons la direction.

```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            var x = 30;
            var y = 30;
            var dx = 1;
            var dy = 1;

            while (true)
            {
                x = x + dx;
                y = y + dy;

                if (x < 0 || x > BrainPad.Display.Width)
                {
```

```
                    dx = dx * -1;
                }

                if (y < 0 || y > BrainPad.Display.Height)
                {
                    dy = dy * -1;
                }

                BrainPad.Display.DrawCircle(x, y, 5);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(0.01);
            }
        }
    }
}
```

Résultat obtenu, le cercle est redessiné lorsqu'il se déplace sur l'écran :

Mais nous voulons avoir une balle rebondissante, alors effaçons l'écran avant de dessiner un cercle. Ajoutez cette ligne juste avant DrawCircle.

```
BrainPad.Display.Clear();
```

Voulez-vous accélérer le déplacement ? Le retard de 0.01 est très petit, donc le retard n'est pas le problème ici. Nous déplaçons simplement le cercle d'un pixel à chaque parcours de la boucle. Déplacez-le de 5 pixels et il sera 5 fois plus rapide. Ceci se fait en modifiant la valeur initiale de dx et de dy à 5.

```
var dx = 5;
var dy = 5;
```

Quelle balle rebondit sans faire de bruit ? Produisons un bip sur le BrainPad à chaque rebond. Ajoutez la ligne ci-dessous dans chaque instruction if…

```
BrainPad.Buzzer.Beep();
```

Ah, ah ! Maintenant, c'est une balle qui rebondit ! Mais le bruit est trop fort, surtout si vous êtes dans une salle de classe avec 30 BrainPads ! Modifiez le code pour produire les sons uniquement si le bouton bas est enfoncé.

```
if (BrainPad.Buttons.IsDownPressed())
{
    BrainPad.Buzzer.Beep();
}
```

Pouvez-vous donner un effet cool à la balle en la rendant plus grande ou plus petite quand elle se déplace ? Pensez à la façon dont nous changeons sa position dans les limites établies. Vous ferez exactement la même chose. Vous avez besoin d'une variable r pour le rayon et d'une variable dr pour la modification du rayon, en augmentant ou en rétrécissant.

```
var r = 3;
var dr = 1;
```

Nous allons augmenter la taille avec des limites comprises entre 1 et 8.

```
r = r + dr;

if (r < 1 || r > 8)
{
    dr = dr * -1;
}
```

Bien sûr, nous devons changer la ligne DrawCircle pour utiliser la variable r au lieu de 5.

```
BrainPad.Display.DrawCircle(x, y, r);
```

Voici le code complet. Mais avant de copier et coller le code, si vous avez nommé votre projet autrement que HelloWorld (pas d'espace entre les mots, sensible à la casse), vous aurez un espace de noms différent. Gardez votre espace de noms et ne copiez pas `HelloWorld`. Par exemple, si votre code est :

```
namespace SomethingElse
```

Gardez la ligne d'espace de noms, et copiez le reste du code, c'est-à-dire tout ce qui commence à partir de la classe Program et se termine par la dernière accolade droite (}). La dernière accolade fait partie de l'espace de noms.

```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            var x = 30;
            var y = 30;
            var dx = 5;
            var dy = 5;
            var r = 3;
            var dr = 1;

            while (true)
            {
                x = x + dx;
                y = y + dy;

                if (x < 0 || x > BrainPad.Display.Width)
                {
                    if (BrainPad.Buttons.IsDownPressed())
                    {
                        BrainPad.Buzzer.Beep();
                    }
```

```
                    dx = dx * -1;
                }

                if (y < 0 || y > BrainPad.Display.Height)
                {
                    if (BrainPad.Buttons.IsDownPressed())
                    {
                        BrainPad.Buzzer.Beep();
                    }

                    dy = dy * -1;
                }

                r = r + dr;

                if (r < 1 || r > 8)
                {
                    dr = dr * -1;
                }

                BrainPad.Display.Clear();
                BrainPad.Display.DrawCircle(x, y, r);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(0.01);
            }
        }
    }
}
```

UN ECLAIRAGE DE NOËL

Vous avez bien lu. Nous allons créer un éclairage de Noël, uniquement avec le Light Bulb. Pour cela, nous utiliserons une fonction parmi les milliers que contient TinyCLR OS. Cette fonction renvoie un nombre aléatoire. Contrairement à la réalité, dans un ordinateur, rien n'est vraiment aléatoire. Cela peut rendre difficile la génération d'un nombre ayant cette propriété. Heureusement, il existe une fonction intégrée qui facilite sa génération. Nous créons simplement un objet aléatoire et lui demandons ensuite une valeur.

Nous utiliserons ces nombres aléatoires pour définir les couleurs primaires rouges, vertes et bleues du Light Bulb. Au cas où vous ne le sauriez pas, ces trois couleurs peuvent créer n'importe quelle couleur.

```
static void Main()
{
    var rnd = new Random();

    while (true)
    {
        BrainPad.LightBulb.TurnColor(rnd.Next(100), rnd.Next(100), rnd.Next(100));
        BrainPad.Wait.Seconds(0.1);
    }
}
```

SUIVRE LA LUMIERE

Nous voulons placer une plante quelque part dans notre intérieur, et nous sommes curieux de savoir quelle quantité de lumière elle recevra. Nous voulons avoir le niveau de lumière tout au long de la journée sous la forme d'un graphique.

Le graphique est très simple. Nous allons essentiellement dessiner une ligne verticale qui commence au niveau de la quantité de lumière et se termine en bas de l'écran. La ligne se déplacera d'un pixel vers la droite à chaque passage dans la boucle.

```
static void Main()
{
    var x = 300;

    while (true)
    {
        x++;

        if (x > BrainPad.Display.Width)
        {
            x = 0;

            BrainPad.Display.Clear();
            BrainPad.Display.DrawSmallText(30, 0, "Light Level");
        }

        var light = BrainPad.LightSensor.ReadLightLevel() / 2;
        var y = BrainPad.Display.Height - light;

        BrainPad.Display.DrawLine(x, y, x, BrainPad.Display.Height);
        BrainPad.Display.RefreshScreen();
        BrainPad.Wait.Seconds(0.1);
    }
}
```

Vous voulez probablement que le programme s'arrête quand il atteint le côté droit de l'écran, mais dans cet exemple, nous allons effacer l'affichage et recommencer au début. Exécutons rapidement la boucle afin d'avoir immédiatement des résultats. Si vous voulez afficher deux heures avec une largeur d'affichage de 128 pixels, vous aurez besoin d'un délai d'une minute. Cela enregistrera 128 minutes sur la largeur de l'affichage.

Vous pouvez également garder le programme tel quel et couvrir le capteur de lumière avec votre main pour faire des dessins sympas !

Puisque nous sommes des experts codeurs, pouvez-vous deviner les réponses à ces questions ?

1. Pourquoi divisons-nous le niveau de lumière par 2 ?
2. Pourquoi définissons-nous la ligne de départ à BrainPad.Display.Height - light ?
3. Pouvez-vous deviner pourquoi x est initialisé avec 300 ? Que se passe-t-il si on utilise 0 à la place ?

Voici les réponses, mais promettez-moi que vous essaierez de trouver d'abord.

Le niveau d'éclairage est compris entre 0 et 100. L'affichage a une hauteur de 64 pixels. Si nous divisons le niveau en deux, la valeur maximale sera 100/2, ou 50. Avec un affichage de 64 pixels de hauteur, nous aurons 14 pixels disponibles en haut de l'écran. C'est là que nous écrirons "Light Level".

La deuxième réponse concerne le fait que nous voulons afficher des niveaux de lumière dans le sens croissant en partant du bas de l'écran. Les faibles valeurs des ordonnées y s'affichent en haut de l'écran, mais nous voulons que les faibles niveaux d'éclairage apparaissent en bas de l'écran. En soustrayant le niveau de lumière de la hauteur de l'affichage, nous inversons le graphique.

La dernière réponse concerne une astuce pour que tout se passe bien au lancement du programme. En effet, nous affichons le texte "Light Level" seulement lors de l'effacement de l'écran. Donc, en mettant x à une valeur élevée, nous serons automatiquement en dehors de la limite que nous avons définie et "Light Level" sera affiché sur l'écran. Essayez de donner à x la valeur 0, "Light Level" ne s'affichera pas à l'écran la première fois que le graphique est tracé. Il s'affichera seulement après. Vous pouvez aussi l'afficher à chaque passage dans la boucle, mais cela ralentira votre programme. Nous devons toujours réfléchir quand nous codons et faire uniquement ce qui est nécessaire.

MULTITACHE

Demander à un système de faire plusieurs choses à la fois est très complexe. Généralement, ce n'est pas convivial sur les petits systèmes. La bonne nouvelle est que TinyCLR OS gère le multitâche et qu'il est facile à utiliser. Il fonctionne exactement comme il le ferait sur n'importe quel système plus important, programmé en .NET. Le multitâche est appelé threading, où chaque thread est une tâche individuelle exécutant une partie du programme.

Vous ne savez pas pourquoi vous auriez besoin du multitâche ? Ni comment exécuter un compteur et l'afficher tout en faisant clignoter le Light Bulb ? Bien sûr, si vous incrémentez à chaque clignotement, c'est facile.

```
Static void Main()
{
    var count = 0 ;

    while (true)
    {
        BrainPad.LightBulb.TurnGreen() ;
        BrainPad.Wait.Seconds(0.1) ;
        BrainPad.LightBulb.TurnOff() ;
        BrainPad.Wait.Seconds(0.9) ;

        BrainPad.Display.DrawSmallText(0, 0, « Count : «  + count) ;
        BrainPad.Display.RefreshScreen() ;

        count = count + 1 ;
    }
}
```

Maintenant, changez le programme pour faire clignoter le Light Bulb une fois par seconde, mais en plus, comptez aussi vite que possible. Vous ne pourrez vraiment pas sans multitâche. Commencez par déplacer le code du clignotement dans sa propre méthode. Une méthode est un morceau de code qui effectue une tâche spécifique. Vous appelez cette méthode pour gérer cette tâche. Par exemple, le Light Bulb a une méthode appelée TurnGreen pour qu'il s'éclaire en vert..

```
static void Blink()
{
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}
```

Si vous appelez cette méthode, elle clignotera une fois par seconde ; Cependant, on n'en sortira jamais en raison de la boucle while. C'est correct, car nous allons appeler cette méthode à partir d'une tâche (thread). Êtes-vous prêt à écrire le code complexe qui crée une tâche ? C'est ici !

```
new Thread(Blink).Start();
```

Nous avons simplement dit au système de créer un nouveau thread pour la méthode Blink, puis de le démarrer.

STOP! Avant d'exécuter le code, nous avons besoin d'un dernier changement. Chaque thread du système doit pouvoir s'endormir. Cela aide le système à traiter ses tâches internes et permet aux autres threads de fonctionner correctement. Vous allez dire que nous n'avons qu'un seul thread. En fait nous en avons deux. Le système crée automatiquement un thread qui exécute Main (), la méthode principale. Si vous devez exécuter le code aussi vite que possible, ajoutez simplement le délai minimum requis.

```
BrainPad.Wait.Minimum();
```

Voici les deux méthodes, Main et Blink.

```
static void Blink()
{
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}

static void Main()
{
    new Thread(Blink).Start();

    var count = 0;

    while (true)
    {
```

```
        BrainPad.Display.DrawSmallText(0, 0, "Count: " + count);
        BrainPad.Display.RefreshScreen();
        BrainPad.Wait.Minimum();

        count = count + 1;
    }
}
```

Il y a deux boucles sans fin et elles fonctionnent toutes les deux simultanément. Bien que ce soit un exemple très simple, il montre les possibilités du multitâche. Comme exercice, ajoutez un autre thread qui émet un bip toutes les trois secondes.

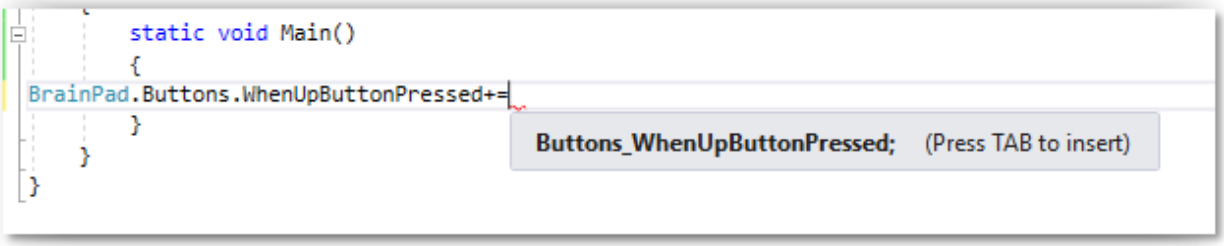
APPELLE-MOI

Nous avons déjà utilisé les boutons, vous savez donc comment ils fonctionnent. Cependant, nous les avons toujours utilisés dans une boucle où nous avons vérifié si le bouton avait été pressé. Le système peut vérifier des millions de fois si le bouton est enfoncé avant qu'il le soit réellement. C'est très mauvais pour les appareils sur piles. Pour économiser la batterie, un bon programme se mettra en veille quand il n'exécute pas de tâche. Si votre téléphone ne se mettait pas en veille, il ne fonctionnerait pas plus d'une heure sans être rechargé.

Ce sujet peut devenir assez technique, donc nous ne parlerons que des événements liés aux boutons. Au lieu de vérifier constamment les boutons, nous demanderons au système de nous appeler lorsqu'un bouton est enfoncé. Nous voulons émettre un bip à chaque pression sur le bouton haut. Nous allons aussi faire clignoter le Light Bulb.

```
static void Main()
{
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);

        if (BrainPad.Buttons.IsUpPressed())
        {
            BrainPad.Buzzer.Beep();
        }
    }
}
```

Si vous appuyez brièvement sur le bouton haut, rien ne se passera. Maintenez-le enfoncé et il émettra un bip une fois par seconde. Ceci est attendu, car la boucle contient des retards et le bouton est vérifié une fois par seconde. Vous pouvez diminuer le délai, mais cela ne résoudra toujours pas le problème. Au lieu de cela, nous allons maintenant utiliser les événements. Visual Studio peut générer automatiquement le code requis. Commencez par



taper "BrainPad.Buttons.WhenUpButtonPressed" et ajoutez "+ =" après. Vous pouvez maintenant appuyer sur la touche de tabulation pour générer la méthode d'événement.

La méthode d'événement générée ressemble à ceci :

```
private static void Buttons_WhenUpButtonPressed()
{
    throw new NotImplementedException();
}
```

Le code généré propose une ligne pour générer une erreur (exception). Remplacer cette ligne par la commande d'un bip.

```
private static void Buttons_WhenUpButtonPressed()
{
    BrainPad.Buzzer.Beep();
}
```

N'oubliez pas de supprimer le code qui vérifie le bouton dans la boucle principale.

```
static void Main()
{
    BrainPad.Buttons.WhenUpButtonPressed += Buttons_WhenUpButtonPressed;

    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}

private static void Buttons_WhenUpButtonPressed()
{
    BrainPad.Buzzer.Beep();
}
```

En passant, plusieurs événements peuvent appeler le même gestionnaire d'événements. Voici un code qui émet un bip lorsque l'un des quatre boutons est enfoncé.

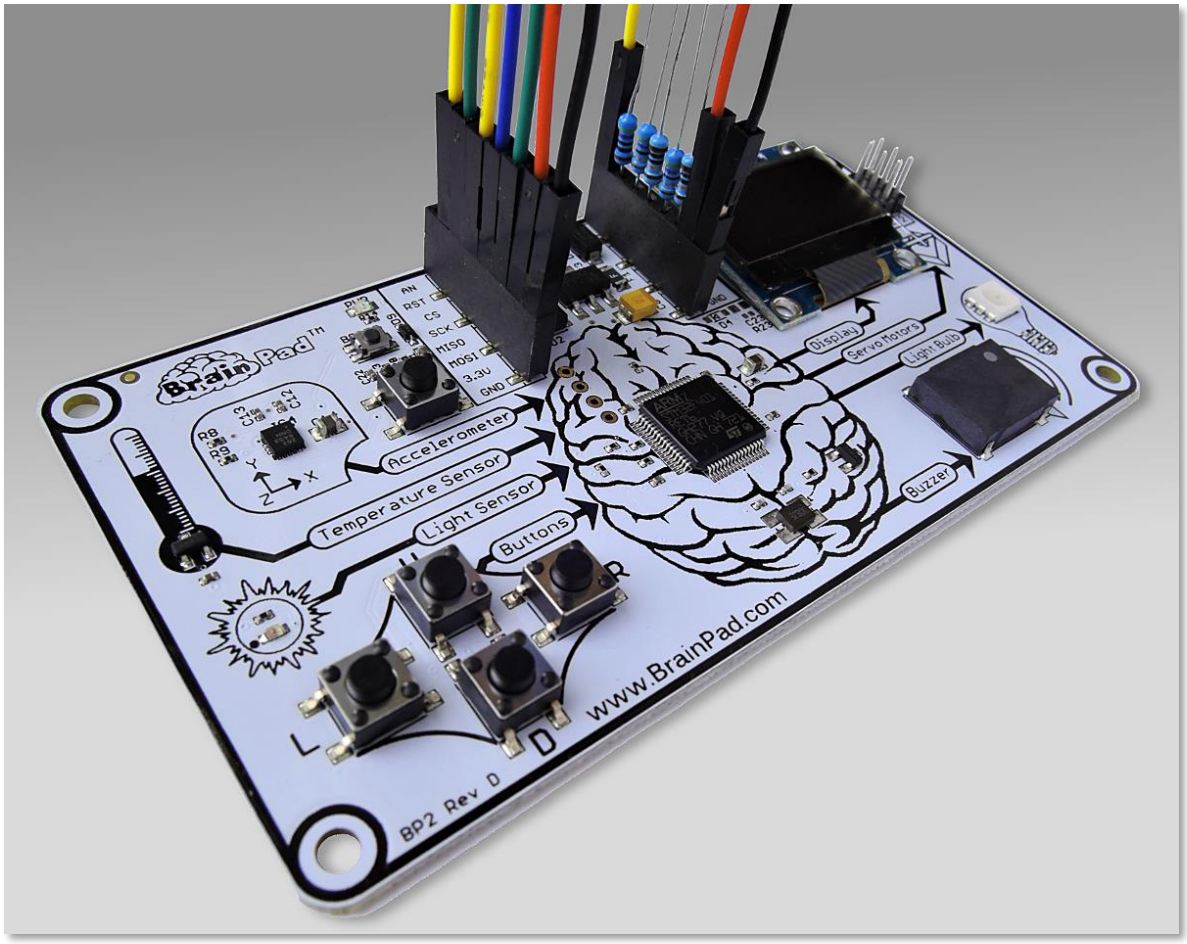
```
static void Main()
{
    BrainPad.Buttons.WhenUpButtonPressed += Beeper;
    BrainPad.Buttons.WhenDownButtonPressed += Beeper;
    BrainPad.Buttons.WhenLeftButtonPressed += Beeper;
    BrainPad.Buttons.WhenRightButtonPressed += Beeper;

    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}
```

```
private static void Beeper()
{
    BrainPad.Buzzer.Beep();
}
```

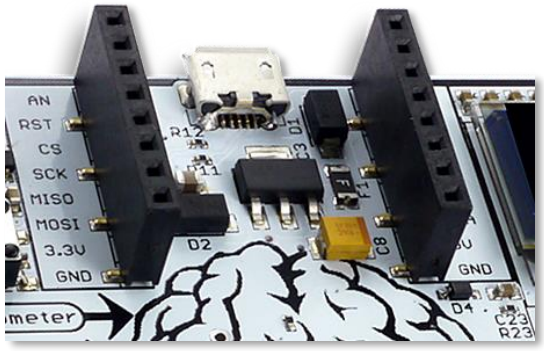
LA PEUR DES FILS

Il est normal que toute personne novice en électronique soit intimidée et effrayée par l'idée de câbler des circuits. Que faire si je suis électrocuté ? Que faire si j'endommage le circuit ?

Le BrainPad rend cela moins effrayant. Tout d'abord, il fonctionne sous une très basse tension de 5 volts (5V). Vous ne pouvez pas vous blesser en touchant 5V. En outre, le BrainPad a une protection intégrée et est conçu pour qu'il soit très difficile de l'endommager en câblant quelque chose de manière incorrecte. Cependant, vous devez toujours essayer de comprendre ce que vous faites. Il existe de nombreux cours et articles en ligne qui vous guideront dans la compréhension des circuits électroniques.

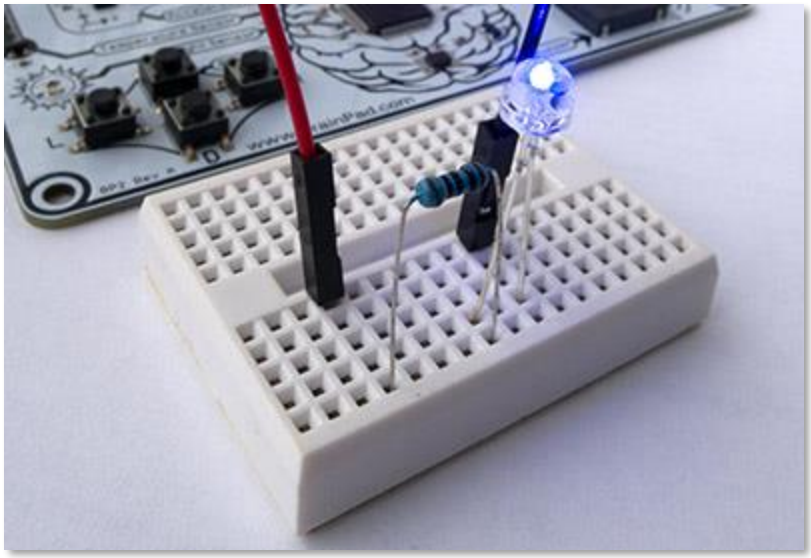


Comme cela a été mentionné dans le premier chapitre de ce livre, il existe plusieurs façons d'augmenter les possibilités du BrainPad, notamment en utilisant les connecteurs femelles situés près du connecteur USB.

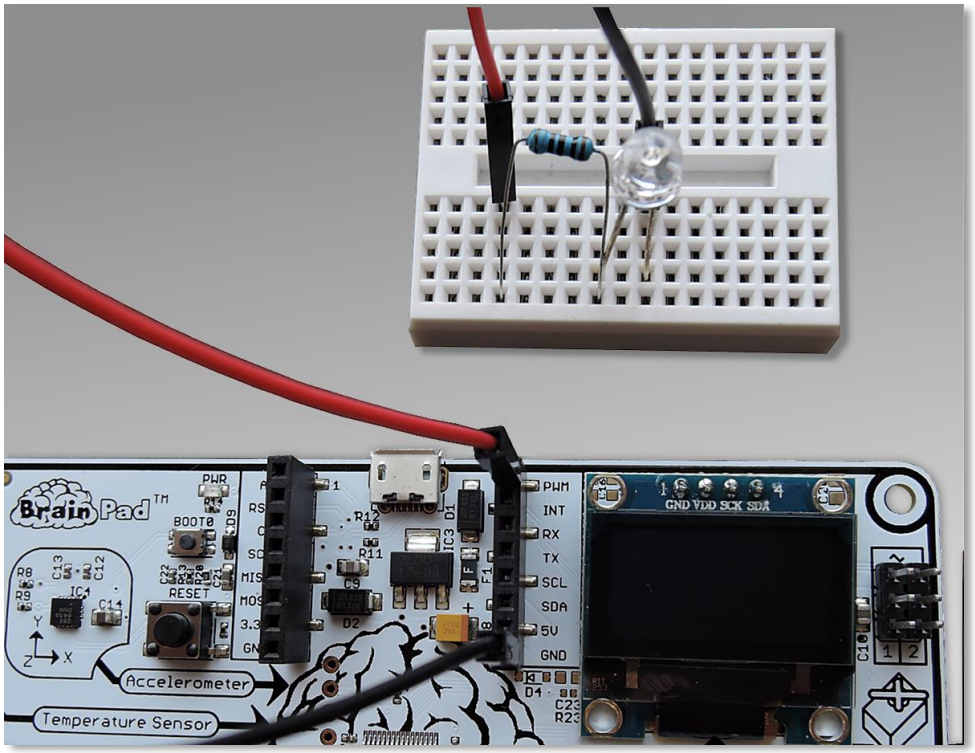


Ces broches ont de nombreuses fonctions et nous aurions besoin de plusieurs livres pour en faire le tour. Nous allons simplifier en connectant une simple DEL (Diode ElectroLuminescente.)

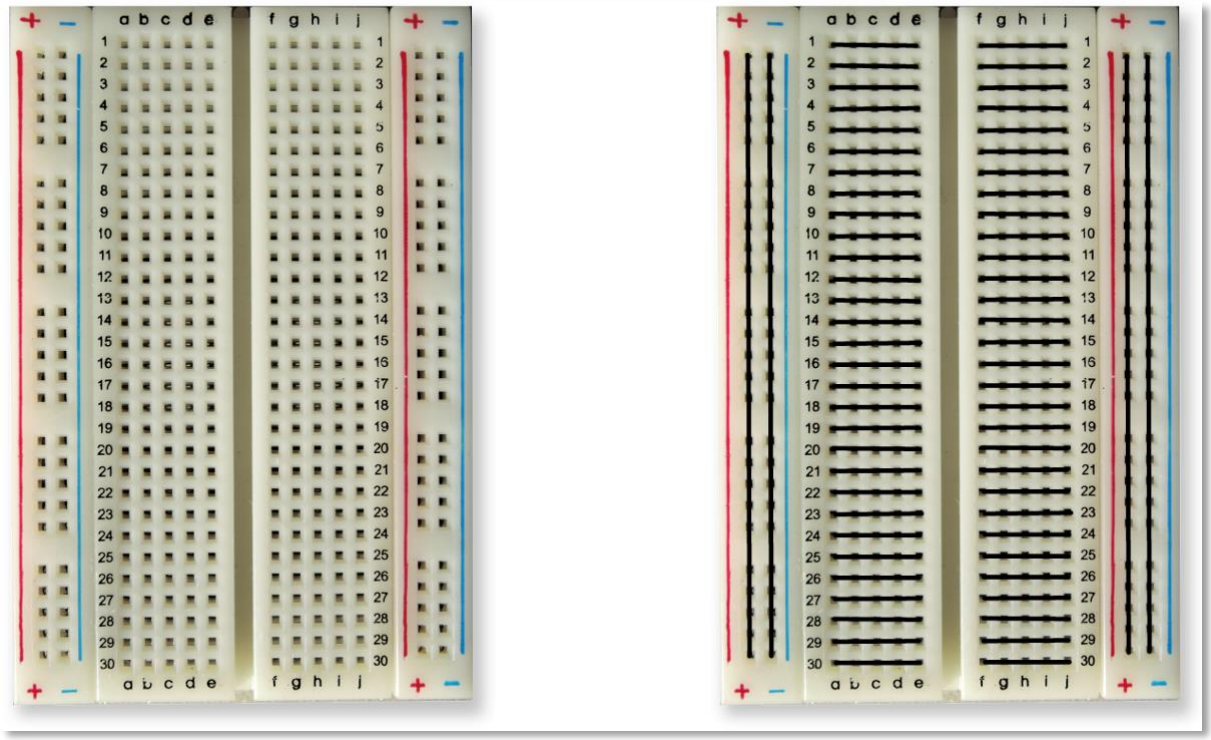
Les DELs sont très bon marché et existent dans différentes couleurs. Elles ont deux connexions avec l'une plus longue que l'autre. Certaines DELs s'éclairent dans plusieurs couleurs (comme le Light Bulb du BrainPad). Nous n'utiliserons pas ces DELs. Pour contrôler une DEL, une résistance de limitation de courant est nécessaire. Les valeurs courantes sont 220, 330 ou 470 ohms. Vous aurez également besoin de quelques fils et d'une planche de test (breadboard).



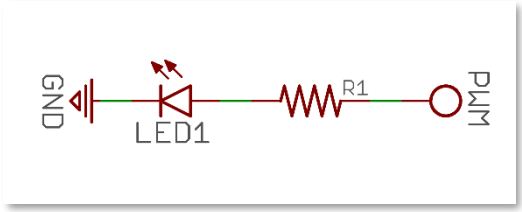
Nous pouvons utiliser l'une des broches GPIO (General Purpose Input Output) du connecteur d'extension pour contrôler la DEL. GPIO signifie entrée, sortie à usage général. Nous utiliserons la broche repérée PWM.



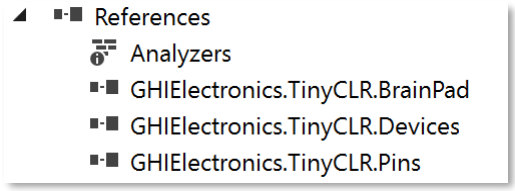
Les plaques d'essai ont des connexions internes où chaque rangée de trous est connectée électriquement. Dans l'image ci-dessous, la plaque d'essai a sur le côté droit des lignes noires montrant les connexions électriques internes. Tout ce qui est branché dans les trous reliés par une ligne noire sera connecté électriquement.



Un fil passera de la broche PWM à la plaque d'essai où le long fil de la DEL est branché. Le fil court de la DEL se connectera à la résistance, et la résistance à la broche GND (terre). Cela forme un cercle (ou un circuit) où les électrons trouveront un chemin entre PWM et GND. Lorsque la broche PWM est activée, la DEL s'allume.



Pour contrôler la broche PWM, avec un GPIO, nous devons inclure les bibliothèques Nugets GHIElectronics.TinyCLR.Devices et GHIElectronics.TinyCLR.Pins. Ces bibliothèques sont indispensables à la bibliothèque GHIElectronics.TinyCLR.BrainPad que nous devons toujours ajouter, aussi aucune autre action n'est nécessaire, car elles sont automatiquement ajoutées.



Nous aurons besoin d'avoir accès aux bibliothèques GPIO dans notre code. Ceci se fait avec une ligne.

```
using GHIElectronics.TinyCLR.Devices.Gpio;
```

Nous devons accéder à une broche à travers le contrôleur GPIO qui se trouve à l'intérieur du processeur.

```
var controller = GpioController.GetDefault();
var pwmPin = controller.OpenPin(BrainPad.Expansion.GpioPin.Pwm);
```

Les broches peuvent être des entrées ou des sorties, tout comme les entrées et les sorties du BrainPad. Les entrées vont dans le "cerveau" et les sorties sortent du "cerveau". Un exemple d'entrée est un bouton, comme les boutons trouvés sur le BrainPad. Un exemple de sortie est une DEL, comme le Light Bulb. Donc, la broche PWM doit être une sortie.

```
pwmPin.SetDriveMode(GpioPinDriveMode.Output);
```

Nous pouvons maintenant simplement "écrire" sur cette broche, pour la rendre haute (activer) ou basse (désactiver). Mettez cela dans une boucle avec quelques retards et vous avez une DEL clignotante.

```
while (true)
{
    pwmPin.Write(GpioPinValue.High);
    BrainPad.Wait.Seconds(0.1);
    pwmPin.Write(GpioPinValue.Low);
    BrainPad.Wait.Seconds(0.5);
}
```

Le code complet de la méthode principale Main devrait ressembler à ceci :

```
static void Main()
{
    var controller = GpioController.GetDefault();
    var pwmPin = controller.OpenPin(BrainPad.Expansion.GpioPin.Pwm);

    pwmPin.SetDriveMode(GpioPinDriveMode.Output);

    while (true)
    {
        pwmPin.Write(GpioPinValue.High);
        BrainPad.Wait.Seconds(0.1);
        pwmPin.Write(GpioPinValue.Low);
        BrainPad.Wait.Seconds(0.5);
    }
}
```

La connexion d'un bouton est similaire, sauf que la broche doit être une entrée au lieu d'une sortie. Sans entrer dans les détails, la broche doit être une entrée avec pull up. Cela la maintiendra au niveau logique haut jusqu'à ce que le bouton soit enfoncé.

```
pwmPin.SetDriveMode(GpioPinDriveMode.InputPullUp);
```

Pas besoin d'ajouter de résistance au bouton. Il suffit de le câbler entre l'un des GPIO et GND.

CONCLUSION

TinyCLR OS de GHI Electronics concède un style de programmation vraiment professionnel au BrainPad. Les connaissances que vous obtiendrez sont utiles pour coder n'importe quel appareil, des téléphones aux ordinateurs. La simplicité d'utilisation est due à notre système d'exploitation TinyCLR et au .NET Framework de Microsoft.

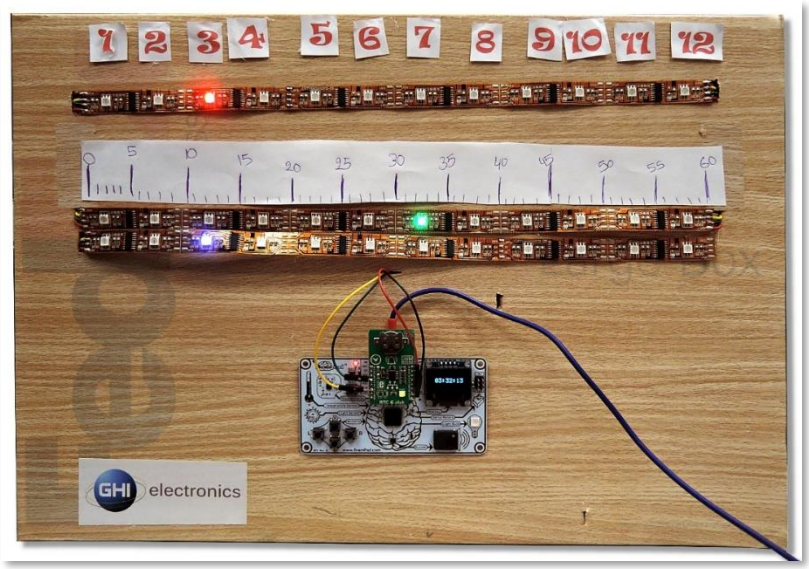
EXTENSION

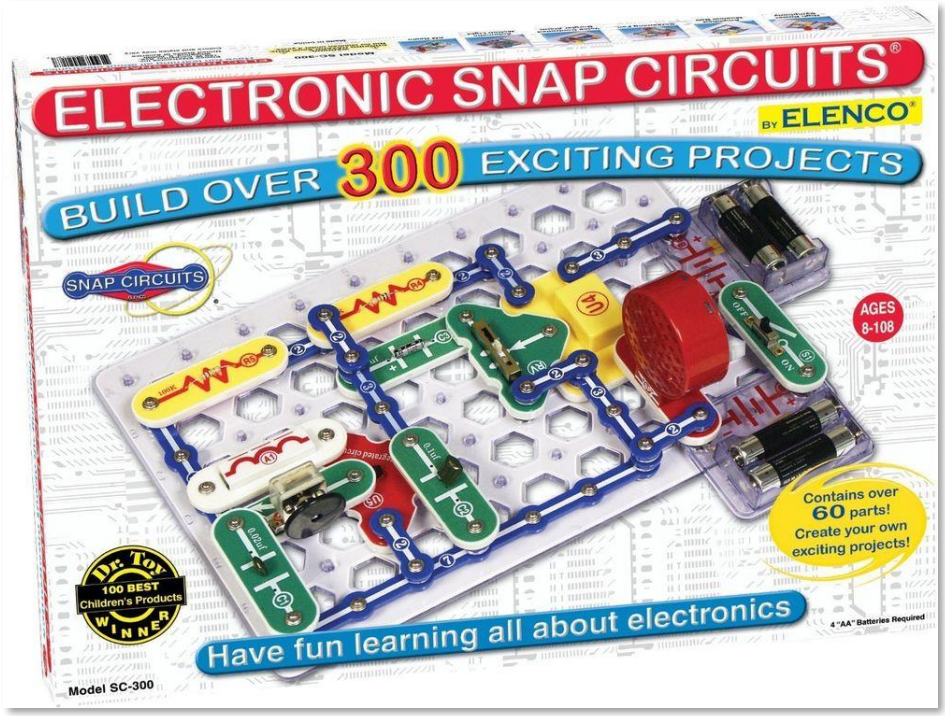
Le connecteur d'extension ouvre tout un monde de possibilités. Nous listons ici quelques options qui vous montrent ce qui est possible.

MIKROELEKTRONIKA CLICK BOARDS™

MikroElektronika propose des centaines de petits modules se connectant sur le BrainPad. Les modules peuvent être de simples capteurs d'humidité ou de température, comme des dispositifs plus sophistiqués tel que la reconnaissance vocale. Il existe également de nombreux actuateurs et des modules de commande comme le contrôle de moteur et d'éclairage. Vous pouvez les trouver sur le site Web https://www.mikroe.com/click.

Dans cet exemple de projet, nous avons créé une horloge linéaire qui utilise un module d'horloge temps réel (HTR) pour suivre l'heure même lorsque le BrainPad n'est pas alimenté (https://docs.brainpad.com/projects/linear-clock.html).





L'un des kits d'apprentissage électronique les plus populaires, fabriqué par Elenco (https://www.elenco.com) s'appelle Snap Circuits. Ces kits se trouvent chez les détaillants locaux ou en ligne et comprennent une variété de composants électroniques et des instructions pour la construction de nombreux projets.

Avec le BrainPad, vous pouvez ajouter de l'intelligence et de la programmation à vos projets électroniques. Cet exemple de projet est un compte à rebours qui lance une hélice. Regardez la vidéo. C'est l'un de nos préférés (https://docs.brainpad.com/projects/lift-off.html).

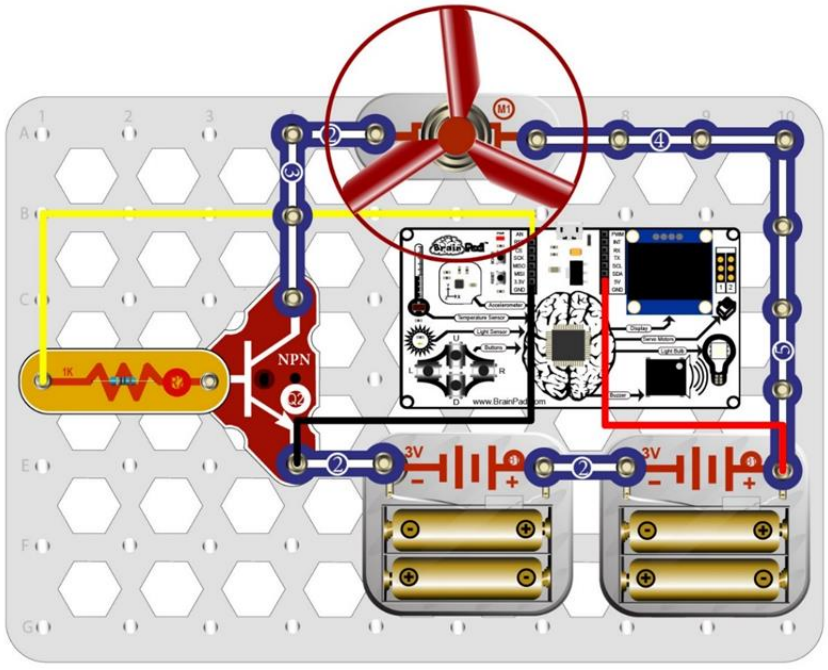
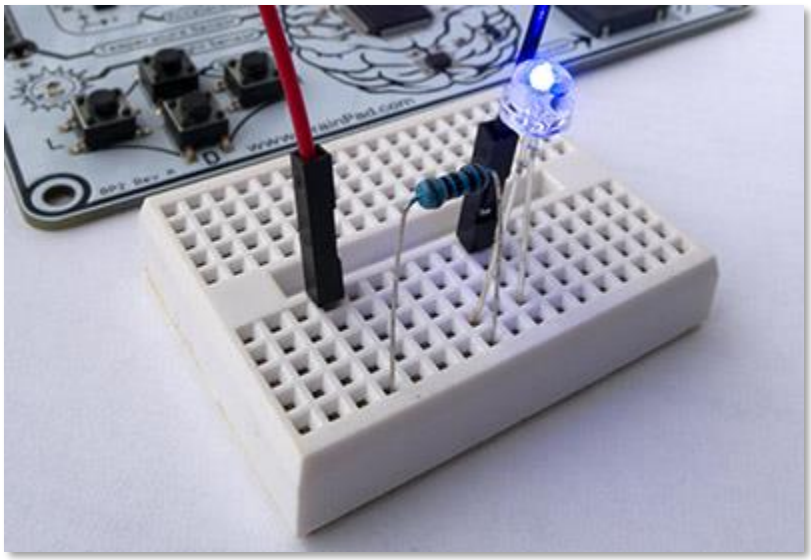


PLANCHE DE TEST (BREADBOARDS)

Voulez-vous concevoir vos propres circuits électroniques à partir de rien comme le font les ingénieurs ? Il existe plusieurs façons de créer des circuits compatibles avec le BrainPad. Bien que la conception de circuits nécessite un peu plus de connaissances, c'est la façon la plus créative et la moins coûteuse d'étendre les capacités de votre BrainPad. C'est aussi la plus éducative.

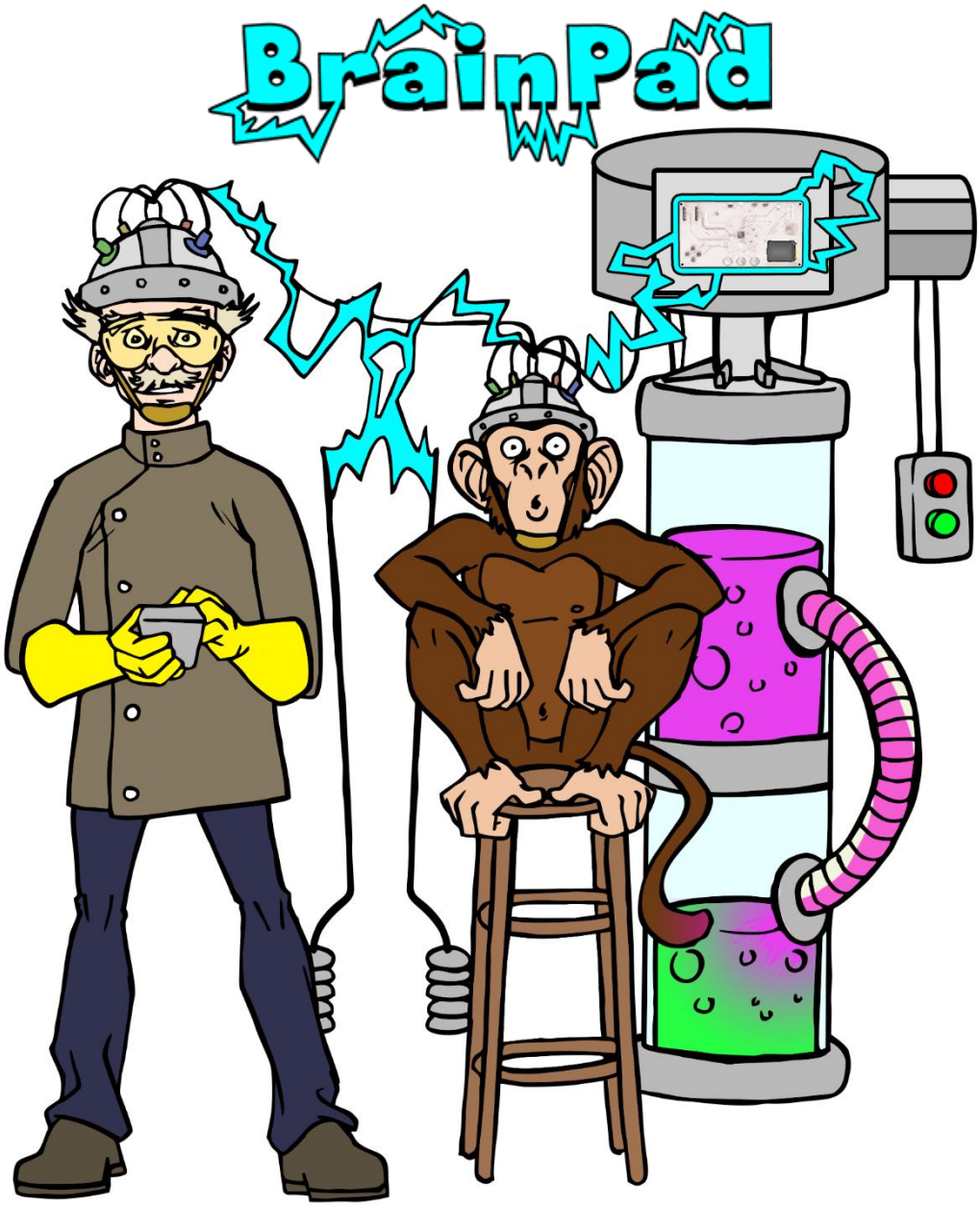
La façon la plus simple de commencer à créer vos circuits est d'utiliser des plaques d'essai sans soudure. Le fil et les composants se branchent simplement dans les trous de la plaque. Elle supporte les composants et les relie ensemble électriquement. Non seulement c'est le moyen le plus rapide de tester un nouveau circuit, mais la correction des erreurs est également plus simple.



Si vous voulez rendre un montage permanent, il y a plusieurs options, allant des cartes de prototypage préfabriquées, faciles à utiliser, à la conception de vos propres cartes de circuits imprimés. Les circuits imprimés peuvent être dessinés à la main ou en utilisant des outils gratuits de conception assistée par ordinateur. C'est ce qui rend le BrainPad si spécial - il est simple à mettre en oeuvre pour débuter, mais vous pouvez aller aussi loin que vous le souhaitez tout en apprenant à votre rythme.

Vous pouvez également ajouter un ou plusieurs capteurs à votre projet BrainPad. En cherchant sur le Web "kit de capteurs" (Sensor kit), vous pouvez trouver de nombreux assortiments de capteurs peu coûteux. Par exemple :

Certaines connaissances sont nécessaires pour utiliser ces modules, mais nous avons quelques projets faciles à réaliser qui vous permettront d'avancer dans la bonne direction. Une fois que vous vous serez familiarisé avec un type de capteur, il deviendra beaucoup plus facile de comprendre comment les autres fonctionnent.



www.BrainPad.com