Détection et analyse de la température - Préface

Ce document a été préparé pour être utilisé par les enseignants qui enseignent le cours de sciences décloisonnées SNC1W en Ontario, au Canada. Il peut être amélioré pour répondre aux besoins d'autres cours de sciences également. Les enseignants sont libres de l'adapter à leurs besoins. Cela peut être trouvé sur csintegration.ca

0. Pourquoi Python ?

La plupart des langages informatiques sont des langages "procéduraux". En tant que programmeur, nous élaborons une procédure (aussi appelée un algorithme) et le codons dans un langage informatique pour que l'ordinateur puisse l'exécuter et résoudre le problème. Tous les langages de programmation ont les trois constructions suivantes:

1. **Séquence:** exécution du programme de gauche à droite et de haut en bas comme si nous lisions un livre ou une page.
2. **Sélection:** Si une condition est vraie, faites ceci, sinon faites autre chose. Par exemple, si la note est supérieure à 49, imprimer "réussite", sinon imprimer "échec".
3. **Itération:** Boucle sur un bloc de code. Par exemple, calculer et imprimer les bulletins de 20 élèves dans une classe.

Ces constructions existant dans tous les langages de programmation, il est facile pour quelqu'un d'apprendre un nouveau langage s'il connaît déjà un langage de programmation. Python est un langage facile à apprendre. Contrairement à d'autres langages de programmation, il est plus "indulgent" envers les débutants en programmation. De plus, il possède tous les éléments linguistiques nécessaires que l'on retrouve dans les langages de programmation modernes. Python est enseigné dans les cours d'informatique de niveau secondaire dans les écoles secondaires de l'Ontario. Il est utilisé par de nombreuses industries de pointe dans tous les domaines, de la production cinématographique à l'analyse du marché boursier en passant par le "*Big Data*".

1. Pourquoi le Micro:bit ?

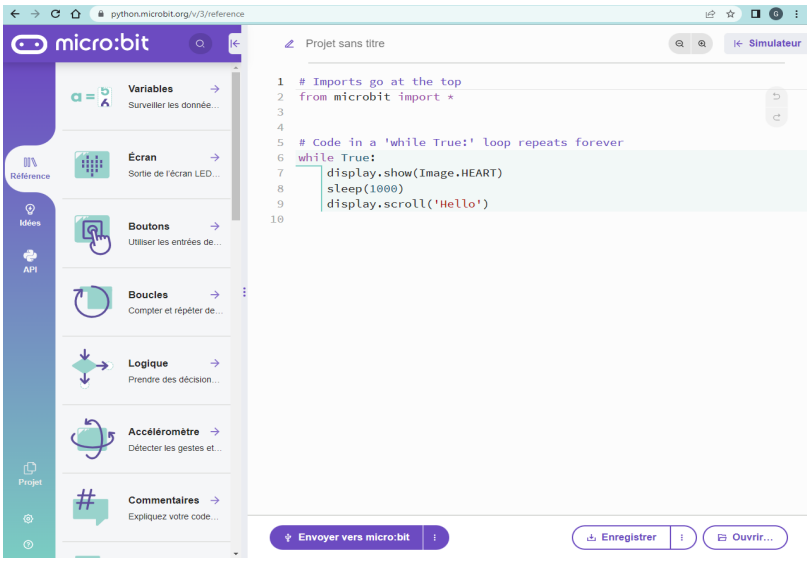
La programmation informatique permet aux enfants d'âge scolaire d'apprendre la programmation et la résolution de problèmes en utilisant un ordinateur. L'interactivité et les composants physiques offrent une représentation visuelle de la programmation. Il est facile d'enseigner que 1 allume une LED ou un moteur et que 0 l'éteint, plutôt que de dire que 1, 0 signifie des bits dans un écosystème informatique! Enseigner la programmation sans indices visuels ressemble aux mathématiques et de nombreux étudiants ont du mal à visualiser et à travailler avec des idées abstraites.

Le Micro:bit est spécialement conçu pour les élèves de 7e à 9e année et peut être étendu aux cours de 10e à 12e année également. De nombreux capteurs intégrés facilitent la connexion, la mesure et l'expérimentation avec le monde physique. Le Micro:bit est donc facile à intégrer dans les cours de sciences. Il est riche en fonctionnalités pour son prix et sa taille!

2. Pourquoi Micropython ?

Micropython est une version adaptée de Python pour fonctionner avec les microcontrôleurs Micro:bit. Micro:bit n'est pas un système informatique complet et ses ressources sont plutôt limitées. Micropython est donc conçu pour fonctionner avec des ressources limitées. Mais il est compatible avec la version "réelle" de Python et le code écrit avec Micropython fonctionnera avec Python. De plus, il est livré avec toutes les fonctionnalités nécessaires qu'un programmeur débutant trouvera utiles, par exemple, des codes d'exemple, une mise en évidence de la syntaxe, une auto-complétion, etc. Et l'interface où les étudiants peuvent écrire et tester leur code est maintenue propre et simple. Le plus important, c'est qu'il a un simulateur intégré où les étudiants peuvent d'abord essayer leur code avant de le connecter à un Micro:bit physique réel. Ce simulateur dispose de tous les périphériques (également appelés capteurs) que l'on trouve sur un appareil physique. Donc, si un code fonctionne sur le simulateur, il fonctionnera sur un appareil physique!

3. Interface de l'éditeur



Vous pouvez lancer l'éditeur Python Micro:bit tel que présenté ci-dessus en allant sur python.microbit.org. Il y a trois panneaux: gauche, milieu et droit. Le code est écrit et testé dans le panneau central. Le panneau de droite est l'endroit où nous trouvons le simulateur. Une fois étendu, sous le simulateur, nous trouvons des périphériques d'entrée et de sortie que nous trouvons sur un véritable appareil tel que le micro, les capteurs tactiles, etc. Lorsque le simulateur est actif, nous pouvons appuyer sur les boutons, tester le microphone, changer le niveau de lumière, etc. et voir comment

notre code répond à ces changements. Le panneau de gauche contient l'onglet de référence contenant des exemples de code que nous pouvons copier et coller ou faire glisser dans l'éditeur.

Le programme de l'éditeur fonctionne à partir d'un navigateur web, il n'est donc pas nécessaire de télécharger ou d'installer un programme séparé sur les ordinateurs de l'école.

4. Saisie, sauvegarde, téléchargement d'un programme

4.1 Dans le simulateur :

Lorsque nous lançons l'éditeur Python Micro:bit, un code d'exemple s'affiche montrant le code pour afficher une icône de cœur suivi du mot «*Hello*» qui défile sur l'écran à LED. Ce code bouclera tant que la condition est '*TRUE*' ou pour toujours tant que l'appareil est sous tension. Ce format de codage est typique de tous les microcontrôleurs conçus pour répéter un ensemble d'instructions tant qu'ils sont en marche.

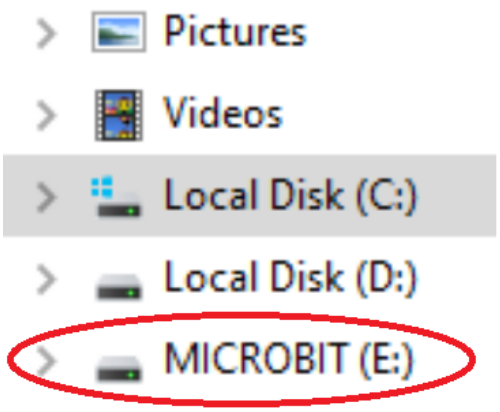
Essayons d'afficher d'autres icônes que nous pouvons afficher sur l'écran à LED. Supprimez le mot '*HEART*' et le point précédent. Pour voir quelles autres icônes sont disponibles, tapez le symbole point '.' et une liste déroulante apparaîtra de toutes les autres icônes que nous pouvons afficher sur les LED. Cette fonctionnalité s'appelle '*code completion*'. C'est pratique car cela réduit les risques d'erreurs et de fautes de frappe et facilite l'apprentissage et l'expérimentation.

Sauvegardons ce que nous avons tapé jusqu'à présent en appuyant sur le bouton "*Enregistrer*". Lors de la sauvegarde, il est toujours bon de donner un nom de fichier significatif comme "Icône heureuse", etc. Le fichier sera enregistré dans le dossier "*Downloads*".

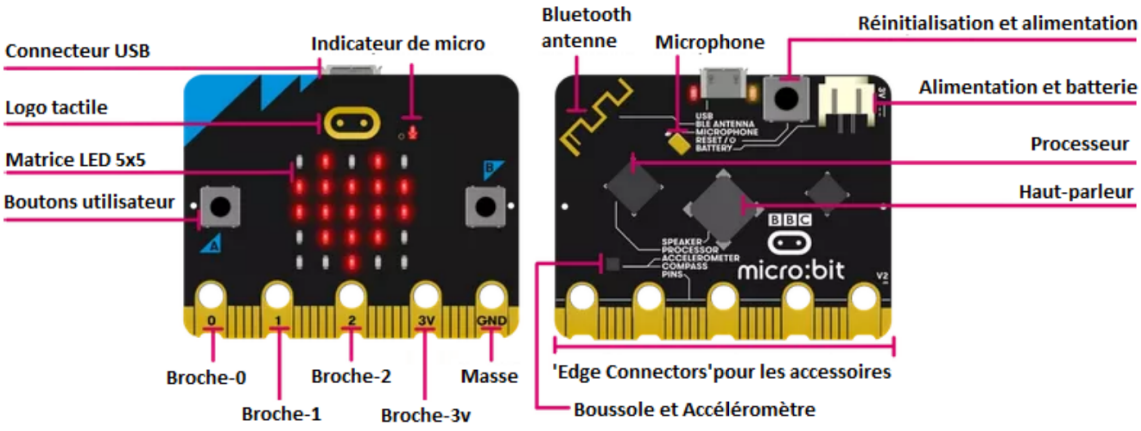
4.2 Sur les vrais Micro:bits :

Le *'Micro:bit Go Paquet'* d'ABRA Electronics est livré avec un câble micro USB, 4 piles AAA en deux paquets, un connecteur de batterie et des fils avec des pinces crocodile.

Lorsque nous connectons le Micro:bit à un ordinateur à l'aide d'un câble USB, il n'est pas nécessaire de connecter la batterie. La connexion USB alimente le Micro:bit. Le Micro:bit apparaîtra comme un lecteur externe sur l'ordinateur. Sur mon ordinateur, il apparaît sous le nom de 'MICROBIT (E:)'. Maintenant que nous avons un dispositif physique, nous pouvons «*envoyer*» ou télécharger le code sur le Micro:bit. Des fenêtres contextuelles nous demanderons de nous connecter en utilisant l'USB, puis de nous connecter au Micro:bit. Ensuite, le code sera envoyé au Micro:bit. Nous pouvons maintenant débrancher le câble USB de l'ordinateur et alimenter le Micro:bit à partir de la batterie. Il fonctionnera maintenant en mode autonome. Il est important de noter que le Micro:bit ne stockera que le dernier programme que nous avons envoyé depuis l'ordinateur. Il ne conservera aucun programme précédent. Nous devons conserver ces programmes sur l'ordinateur. Il serait donc judicieux de créer un dossier séparé sur l'ordinateur pour enregistrer les programmes plutôt que de les enregistrer tous dans le dossier de téléchargements par défaut.



4.3 Les broches sur un Micro:bit :



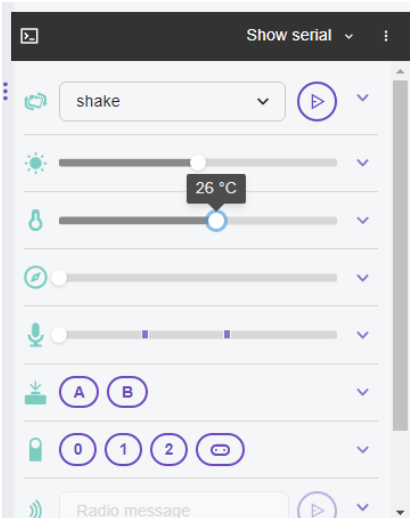
Les deux vues montrent les broches et autres composants d'un Micro:bit. Les broches d'alimentation sont 3V (3,3V pour être précis) et GND (pour la masse ou négatif). Les broches 0, 1 et 2 *(Pin-0, Pin-1, Pin-2)* sont utilisées comme broches d'entrée ou de sortie. L'entrée signifie que nous pouvons envoyer une tension dans le Micro:bit et la sortie signifie que nous pouvons en obtenir une tension. Des pinces crocodile peuvent être utilisées pour se connecter à ces broches. De petites encoches sur le bord facilitent la connexion des pinces.

La matrice de LED 5 x 5 affiche des icônes, des messages, etc. Elle est également utilisée comme capteur de lumière pour mesurer le niveau de lumière ambiante. Notre code peut interagir avec les boutons A et B. Le logo Micro:bit en haut peut également être utilisé comme entrée tactile dans notre code. À droite, nous verrons les composants internes tels que l'antenne, l'accéléromètre, le haut-parleur, la boussole et le commutateur de réinitialisation. Le commutateur de réinitialisation, comme son nom l'indique, sert à redémarrer le Micro:bit.

5. Capteur de température

En SNC1W, dans le cadre des attentes du '*Domaine A. Habiletés liées aux STIM, carrières et liens connexes*', il y a quelques attentes que cette activité abordera. Mesurer la température à l'aide du Micro:bit. Et éventuellement la consigner pour une analyse ultérieure.

5.1 Utilisation du simulateur :

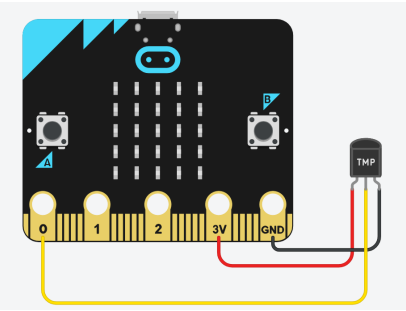


Tout d'abord, construisons ce thermomètre sur le simulateur. Lancez l'éditeur et vous aurez le code par défaut pour afficher le cœur et le mot *'Hello'*. Écrivons le code pour la température à la fin. Dans le panneau de gauche, faites défiler vers le bas et cliquez sur '*Temperature*' et vous verrez un extrait de code `display.scroll(temperature())` s'afficher.

Faites glisser et déposez cet extrait de code et appuyez sur le bouton de lecture sur le simulateur. Vous verrez la température affichée comme 21. Vous pouvez ajuster la température en déplaçant le curseur de -5°C à +50°C.

5.2 Utilisation d'un capteur de température LM 35 :

Le LM 35 est un capteur de température de précision qui produit une sortie de 10 millivolts / °C. Il possède trois broches. Il fonctionne de 0 à 100 °C. Lorsque vous faites face au capteur (le côté plat vers vous), la broche de gauche est l'alimentation, la broche de droite est la masse ou le négatif. La broche du milieu est la sortie du capteur. Après l'avoir alimenté comme indiqué, si nous utilisons un voltmètre sur la broche du milieu, il affichera 200 mV pour une température ambiante de 20°C. Nous allons utiliser le Micro:bit comme un thermomètre numérique. J'ai connecté la broche du milieu à la broche 0. Nous aurions également pu utiliser la broche 1 ou 2.

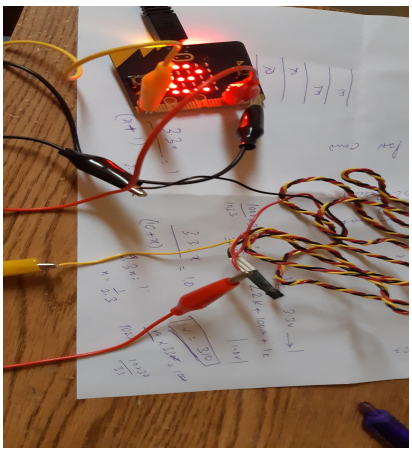


Le LM 35 est un capteur de température de précision qui produit une sortie de 10 millivolts / °C. Il possède trois broches. Il fonctionne de 0 à 100 °C. Lorsque vous faites face au capteur (le côté plat vers vous), la broche de gauche est l'alimentation, la broche de droite est la masse ou le négatif. La broche du milieu est la sortie du capteur. Après l'avoir alimenté comme indiqué, si nous utilisons un voltmètre sur la

broche du milieu, il affichera 200 mV pour une température ambiante de 20°C. Nous allons utiliser le Micro:bit comme un thermomètre numérique. J'ai connecté la broche du milieu à la broche 0. Nous aurions également pu utiliser la broche 1 ou 2.

`tempVoltage` sera un nombre de 0 à 1023 qui est ensuite converti en température. Cette valeur de température aura quatre décimales, ce qui est difficile à lire sur l'affichage LED défilant. J'ai donc converti cela en un entier en utilisant la fonction `int` pour une lecture plus facile.

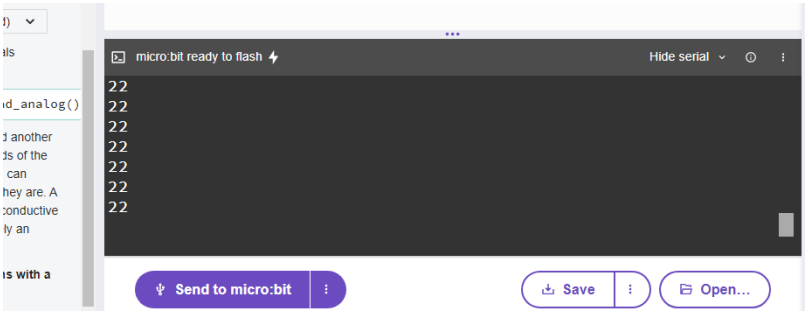
Sur l'image, le capteur LM 35 est près de la pince crocodile rouge. J'ai rallongé les broches du capteur en utilisant trois fils de raccordement longs, rouge, noir et jaune. En suivant un schéma de code couleur clair, le rouge pour le positif, le noir pour le négatif et d'autres couleurs pour les signaux, le câblage et le dépannage sont beaucoup plus faciles. Il est préférable de connecter les fils de raccordement aux broches du capteur, puis de connecter les pinces crocodile aux fils rallongés plutôt que de connecter les pinces crocodile directement aux broches du capteur, car les pinces crocodile pourraient endommager les broches du capteur.



La température était de 22°C. La température augmentera si nous pincions le capteur, mais n' atteindra pas la température corporelle de 37°C car les doigts ne montrent pas la température corporelle.

```
from microbit import *
while True:
    tempVoltage = pin0.read_analog()
    display.scroll(int((tempVoltage/1024.0)*3300/10))
    sleep(1000) # pause de 1000 millisecondes ou 1 seconde
```

Si vous ajoutez la ligne de code ci-dessous, avant sleep(1000), vous pouvez voir le même résultat dans l'éditeur également. Cliquez sur le bouton ***Show serial*** (Afficher la console série) situé sous l'éditeur pour ouvrir la console série.




```
print(int((tempVoltage/1024.0)*3300/10))
```

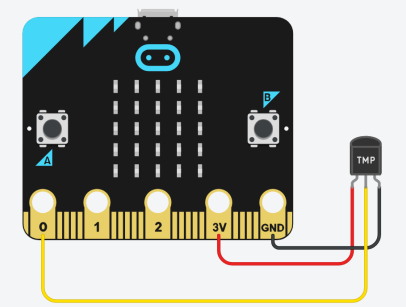
Le symbole '#' indique un commentaire pour l'utilisateur et est ignoré par le Micro:bit. La phrase '***pause de 1000 millisecondes ou 1 seconde***' est strictement destinée à l'utilisateur et le programme l'ignore complètement. Utiliser '#' est un excellent moyen de prendre des notes ou d'ajouter des commentaires pour une référence future.

Dans cette activité, nous avons appris comment obtenir la température à partir du simulateur et d'un véritable capteur LM 35 en construisant un thermomètre numérique. Dans un prochain laboratoire, nous apprendrons comment enregistrer la température.

6. Enregistrement de la température

Dans cette activité, nous allons revisiter le capteur de température, mais nous allons utiliser le Micro:bit pour enregistrer la température sur une période de temps. Nous allons également sauvegarder la température dans un fichier pour une analyse ultérieure. Cette activité répond aux attentes spécifiques : STIM Applications, carrières, et liens connexes.

A1.1: appliquer une démarche de recherche et les habiletés connexes pour effectuer des recherches afin d'établir des liens entre celles-ci et les concepts scientifiques à l'étude.



A1.2: appliquer une démarche expérimentale et les habiletés connexes pour effectuer des expériences afin d'établir des liens entre ses observations et conclusions et les concepts scientifiques à l'étude.

A1.3: appliquer un processus de design en ingénierie et les habiletés connexes pour concevoir, construire et tester des dispositifs, des modèles, des structures et/ou des systèmes.

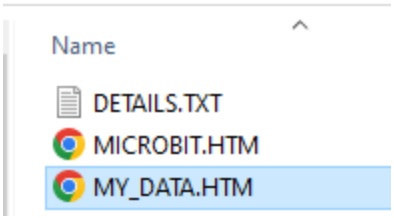
A1.4: appliquer des habiletés en codage pour examiner et modéliser des concepts scientifiques et des relations connexes.

Référez-vous à la section Utilisation d'un capteur de température LM 35. Connectez le capteur LM35 comme indiqué. La broche de gauche est +3V (Rouge), la broche de droite est GND ou négative (Noir), la broche du milieu est la sortie qui se connecte à la broche 0 du Micro:bit.

Le code suivant enregistre la température toutes les 5 secondes. Cela peut être modifié en heures (h), minutes (min), secondes (s) ou millisecondes (ms) dans l'instruction @run_every. Le Micro:bit n'étant pas un ordinateur, il n'a pas de concept de temps. Le temps est mesuré à partir du moment où il est alimenté. Vous pouvez trouver plus d'informations sur l'enregistrement dans la section *Référence* de l'éditeur.

```
from microbit import *
import log
log.set_labels('temp',timestamp=log.SECONDS)
# log every five seconds
@run_every(s=5)
def log_data():
    log.add({
        'temp' : temperature(),
    })
def temperature():
    tempVoltage = pin0.read_analog()
    return (int((tempVoltage/1024.0)*3300/10))

while True:
    sleep(100000)
```



Le Micro:bit crée un fichier journal appelé "MY_DATA.HTM" dans le lecteur MICROBIT(E:). Il s'agit d'une page web qui peut être ouverte dans un navigateur. Vous verrez les données capturées toutes les 5 secondes dans un tableau. Vous pouvez également visualiser les données sous forme de graphiques.

Les tableaux et graphiques suivants présentent les propriétés d'isolation de deux flacons remplis d'eau à environ 54°C pour commencer l'expérience. La baisse de température est enregistrée toutes les cinq minutes pendant environ une heure.

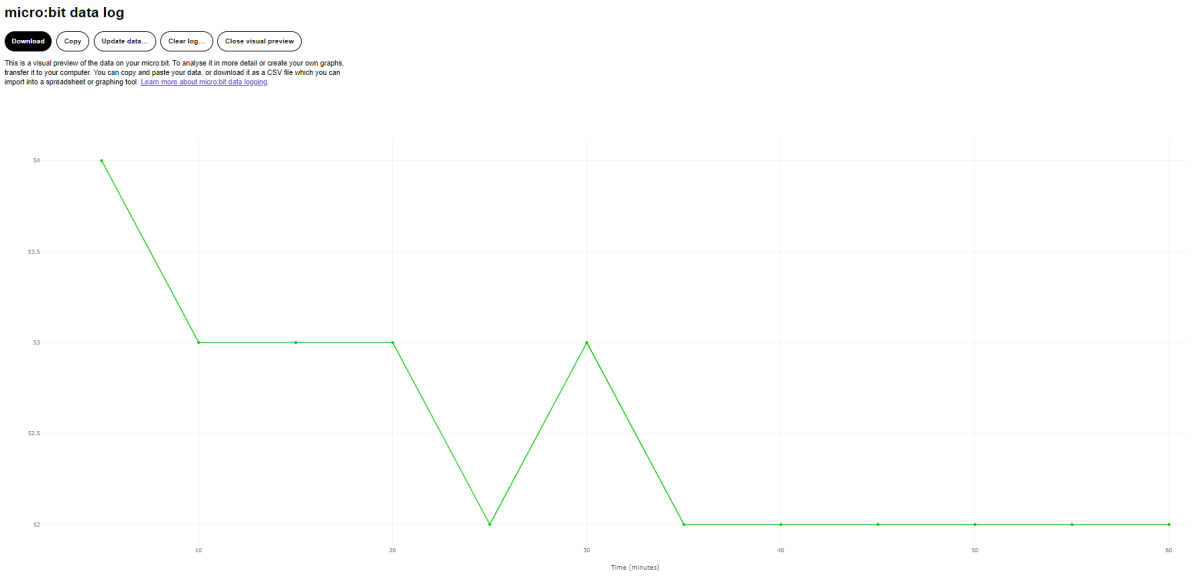
Le premier ensemble de données correspond au flacon de gauche, et le deuxième ensemble de données correspond au flacon de droite. On peut observer que les deux flacons perdent plus ou moins de la chaleur à la même vitesse. Pour tirer une conclusion, il serait nécessaire de continuer à enregistrer les données de température sur une période plus longue.

micro:bit data log

Download | Copy | Update data... | Clear log... | Visual preview

This is the data on your micro:bit. To analyse it and create your own graphs, transfer it to your computer. You can copy and paste your data, or download it as a CSV file which you can import into a spreadsheet or graphing tool. Learn more about micro:bit data logging.

Time (minutes)	temp
5.01	54
10.01	53
15.01	53
20.01	53
25.01	52
30.01	53
35.01	52
40.01	52
45.01	52
50.01	52
55.01	52
60.01	52



micro:bit data log

Download | Copy | Update data... | Clear log... | Visual preview

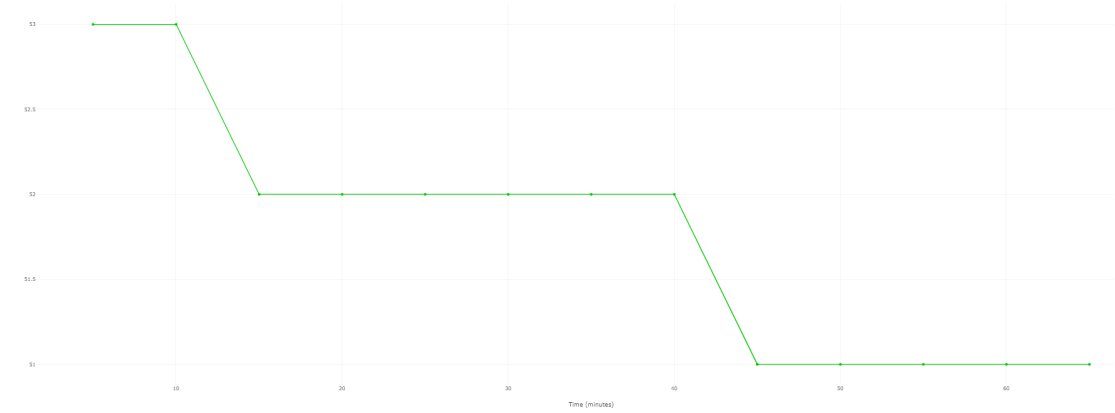
This is the data on your micro:bit. To analyse it and create your own graphs, transfer it to your computer. You can copy and paste your data, or download it as a CSV file which you can import into a spreadsheet or graphing tool. Learn more about micro:bit data logging.

Time (minutes)	temp
5.01	53
10.01	53
15.01	52
20.01	52
25.01	52
30.01	52
35.01	52
40.01	52
45.01	51
50.01	51
55.01	51
60.01	51
65.01	51

micro:bit data log

Download | Copy | Update data... | Clear log... | Close visual preview

This is a visual preview of the data on your micro:bit. To analyse it in more detail or create your own graphs, transfer it to your computer. You can copy and paste your data, or download it as a CSV file which you can import into a spreadsheet or graphing tool. Learn more about micro:bit data logging.



7. Conseils pour prendre soin de votre Micro:bit

- ★ Le connecteur Micro USB sur le Micro:bit est délicat. Les branchements répétés peuvent endommager le connecteur. Il est recommandé de laisser le câble USB connecté à l'extrémité du Micro:bit et de simplement le brancher et le débrancher du PC au besoin.
- ★ Le connecteur de la batterie est également délicat. Il est recommandé de laisser le câble de la batterie connecté au Micro:bit et de simplement retirer les 2 piles AAA lorsque l'appareil n'est pas utilisé.
- ★ Rappelez aux étudiants de télécharger le programme par défaut avant de rendre le Micro:bit. De cette manière, les enseignants pourront vérifier l'état du Micro:bit.

Annexes

A.1 Composants et liens utiles

Les composants proviennent du site abra-electronics.com, un détaillant en électronique basé à Montréal.

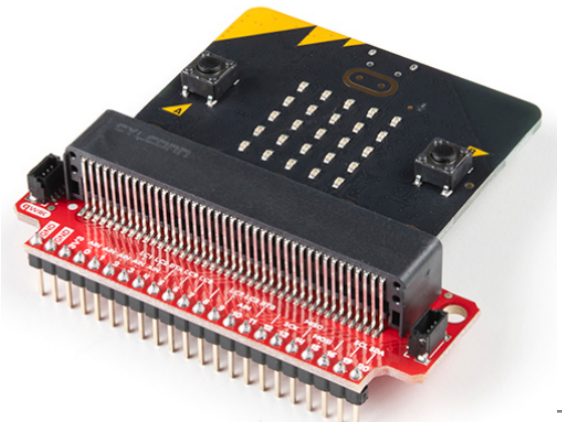
1. micro:bit v2 Go Paquet
2. LM35DZ CI, Capteur de Température
3. CDS004-5001 Lumière Photocellule
4. 10K Ohm Résistance
5. Fils sur Bobines 22 AWG Solide Cela vient dans une petite boîte avec six bobines de fils de différentes couleurs pour une distribution facile.
6. Pince-Dénudeur réglable 10-30 AWG Ceci est un outil pratique pour dénuder les fils. Utilisez l'encoche pour maintenir fermement le fil et tirez pour dénuder le fil. Il nécessite la bonne quantité de pression, une pression excessive risque de couper le fil.
7. CAPTEUR DE CO2 pour MICRO:BIT (MONKMAKES)

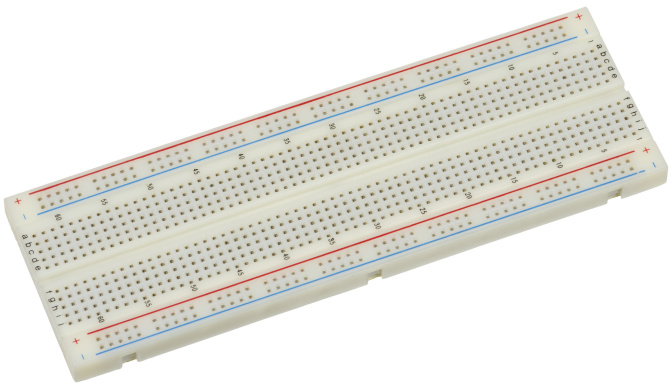
Lines utiles

- ★ BBC Micro:bit
- ★ Micro:bit pour les enseignants
- ★ Éditeur Micro Python
- ★ https://microbit-micropython.readthedocs.io/en/v2-docs/umentation

A.2. Pour aller plus loin

Jusqu'à présent, nous n'avons utilisé que trois broches sur le Micro:bit, sans compter les broches d'alimentation. Si nous construisons des projets plus avancés, tels qu'un robot suiveur de ligne, nous pouvons avoir besoin de plus de broches pour connecter le Micro:bit. Cela peut être réalisé en utilisant une Platine à déploiement Micro:bit, comme illustré ci-dessous, avec une Plaque Experimental (*breadboard sans soudure*). La carte de dérivation permet de connecter toutes les autres broches du Micro:bit et elle possède des broches d'en-tête qui peuvent être placées dans la breadboard pour ajouter plus de composants.





A.3. Introduction à Python

Python est un "langage de programmation de haut niveau" similaire à Java, C, C++, Visual Basic, etc. Haut niveau signifie que le code ressemble à des instructions en anglais et a du sens. Voici quelques caractéristiques de base de Python :

- ★ Python est sensible à la casse : "True" et "true" sont différents.
- ★ # est utilisé pour commencer un commentaire, et il est ignoré par Python. Il permet d'expliquer le code à l'utilisateur et facilite la compréhension du code.
- ★ Il y a une seule instruction Python par ligne.
- ★ Les constructions de sélection (if, else) et d'itération (while) se terminent par un deux-points ':' Et une fonction commence par un def. Par exemple, `def temperature():`
- ★ Les indentations correctes démarrent un bloc de code qui peut contenir une ou plusieurs instructions Python. Un bloc de code peut être comparé au début d'un paragraphe avec une indentation à droite. Dans l'exemple suivant, `display.scroll` ne s'exécutera que si le bouton A est pressé (ou si la condition est True), car il est indenté, ce qui signifie qu'il est contrôlé par la condition .`sleep(500)` s'exécutera indépendamment de la condition if.

```
if button_a.was_pressed():
      display.scroll('T=' + str(get_tempC()),100)
sleep(500)
```

- ★ Les variables sont utilisées pour stocker des valeurs pendant l'exécution du programme. Elles commencent par une lettre et ne doivent pas contenir d'espaces. Dans le code ci-dessous, il y a quatre variables : `'co2', 'tempC', 'rh', 'response'`. La variable `'co2'` fonctionne, mais `'2co'` en tant que nom de variable est une erreur car il commence par un chiffre. response est un nom de variable valide, mais my response est une erreur car il contient un espace.

```
co2 = 0 # initialiser la variable à 0
tempC = 0
rh = 0
response = ""
```

- ★ Il est bon d'initialiser les variables car elles peuvent contenir des valeurs indésirables provenant des exécutions précédentes du programme. L'initialisation à 0 est toujours une option. Comme on dit, un ordinateur est une machine GIGO, ce qui signifie "*Garbage In, Garbage Out!*" (Des entrées non valables produisent des sorties non valables !)

*****