Fiche - Programmation & Capteurs

Collecter des données avec le capteur d'humidité et de température

Matériel et outils nécessaires

- **Carte Micro:bit V2 et ses capteurs intégrés** : il s'agit de la carte programmable principale. Elle comprend un capteur de lumière (via l'écran LED), un capteur sonore, ainsi qu'un capteur de température intégré. *Prix indicatif : environ 19 € par carte Micro:bit.* (Voir les prix)
- **Câble micro-USB** : permet d'alimenter la carte et de la programmer depuis un ordinateur.
- **Batterie externe (optionnelle)** : utile pour un fonctionnement en autonomie si la carte est détachée de l'ordinateur. *Le boîtier de piles officiel Micro:bit est disponible pour environ 2,20 € par unité ici.*

Vous pouvez également acheter le kit Micro:bit V2 comprenant le câble USB et le boitier de piles pour 21 EUR par kit(ici) ou 177 EUR pour 10 kits (ici)

- **Capteur DHT22 (ou DHT11)** : ces capteurs sont populaires pour mesurer l'humidité et la température avec des microcontrôleurs. *Le DHT11 est bon marché et suffisant pour des projets simples, tandis que le DHT22 offre une meilleure précision et une résolution supérieure, pour un coût légèrement plus élevé.*
- **Ordinateur ou tablette** : utilisé pour écrire le code et le transférer vers la Micro:bit.
- **Environnement de programmation** : l'éditeur en ligne MakeCode est recommandé pour programmer facilement la carte Micro:bit.

Pour cette étape, il est recommandé de programmer entre 3 et 6 cartes Micro:bit afin de les répartir entre les élèves et de recueillir un plus grand volume de données. Il est possible de réaliser l'activité avec une seule carte, mais cela nécessitera soit **d'allonger la période globale de collecte**, soit **de réduire la durée de collecte par élève**, en passant de 7 à 3 jours environ.

Câblage et utilisation d'une carte Micro:bit

Suivez les étapes ci-dessous pour programmer, installer, enregistrer et récupérer des données environnementales à l'aide d'une carte Micro:bit.

Étape 1 : Câblage du capteur de température/humidité à la carte Micro:bit. Il existe deux types de capteurs **DHT11/DHT22** :

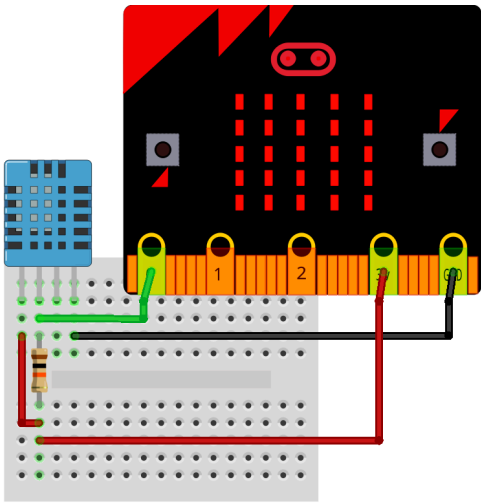
1. **Version sans carte PCB**, avec **4 broches** ;
2. **Version montée sur carte PCB**, avec **résistances de rappel intégrées** et seulement **3 broches**.

Nous vous recommandons d'utiliser **la version avec PCB**, plus simple à connecter. Pour la version **avec PCB** (3 broches) :

- **Vcc (+)** : à connecter à **3,3 V ou 5 V** (les deux tensions sont compatibles)
- **GND (-)** : à connecter à la masse (**GND**)
- **Data (OUT)** : à connecter à **n'importe quelle broche GPIO** de la Micro:bit

Pour la version **sans PCB** (4 broches) :

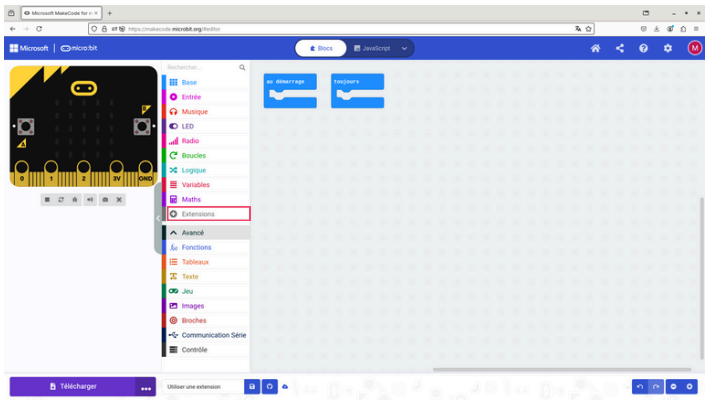
- Vous devez ajouter une **résistance de rappel** entre **Vcc et Data** pour maintenir la broche Data à l'état haut. *Une résistance comprise entre **220 Ω et 10 kΩ** fonctionne correctement sous 3,3 V ; au-delà, le capteur risque de ne pas répondre.*
- Vous pouvez aussi utiliser l'option **pull-up interne** de la Micro:bit : dans MakeCode, allez dans le menu **"Broche" > "Plus" > "Régler le levier en broche..."**. La Micro:bit possède des **résistances de rappel internes** d'environ **12–13 kΩ**.
- **Remarque** : la **troisième broche à partir de la gauche** (sur la version 4 broches) **n'est pas utilisée**.



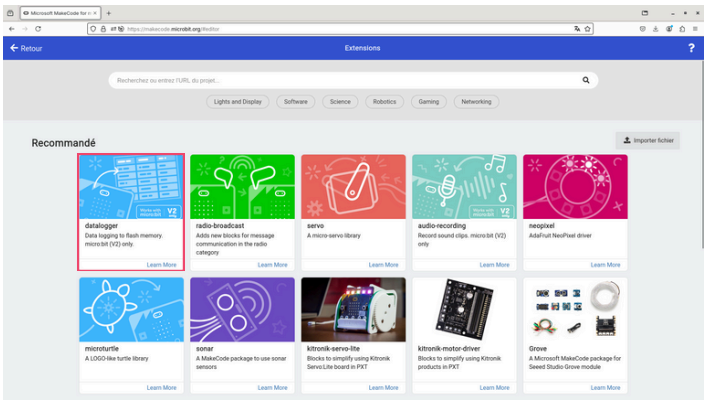
Connectez votre capteur DHT en suivant le schéma ci-contre.

<u>Étape 2 : Programmation de la Micro:bit</u>. À l'aide de votre câble USB, connectez la carte à votre ordinateur via le connecteur micro-USB. Une fois connectée, la carte Micro:bit apparaîtra sur l'ordinateur comme un lecteur amovible (par exemple, « MICROBIT »). Ouvrez l'éditeur MakeCode pour créer un programme qui collecte des données de lumière, de bruit et de température à l'aide des capteurs intégrés de la carte Micro:bit V2. Donnez un nom clair à votre projet avant de commencer.

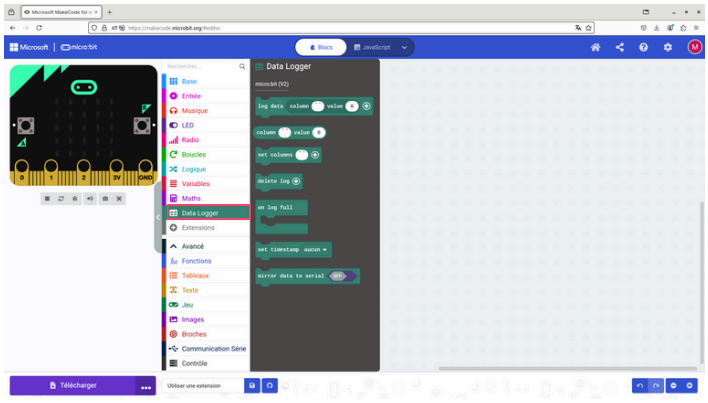
Une fois dans l'éditeur et après avoir créé votre nouveau projet, vous verrez apparaître l'écran par défaut « prêt à l'emploi ». Vous devrez alors installer une extension. Les extensions dans MakeCode sont des groupes de blocs qui ne sont pas inclus directement dans les blocs de base. Comme leur nom l'indique, elles ajoutent des blocs pour des fonctionnalités spécifiques. Il existe des extensions pour un large éventail d'usages : créer une manette de jeu, un clavier, une souris, contrôler un servomoteur, etc. Dans la colonne des groupes de blocs, cliquez sur **EXTENSIONS**. Dans la liste des extensions disponibles, recherchez l'extension **Datalogger**, qui sera utilisée pour cette activité. Cliquez sur l'extension souhaitée : un nouveau groupe de blocs apparaîtra sur l'écran principal. Faites de même pour le capteur de température/humidité en recherchant l'extension **DHT11/DHT22**.



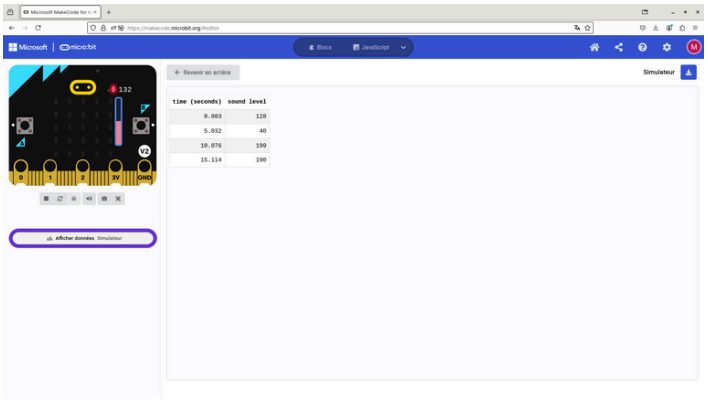
1 **Ouvrez le menu des extensions depuis l'éditeur MakeCode pour micro:bit.**



2 **Sélectionnez l'extension Datalogger dans la liste (vous pouvez également utiliser l'outil de recherche).**



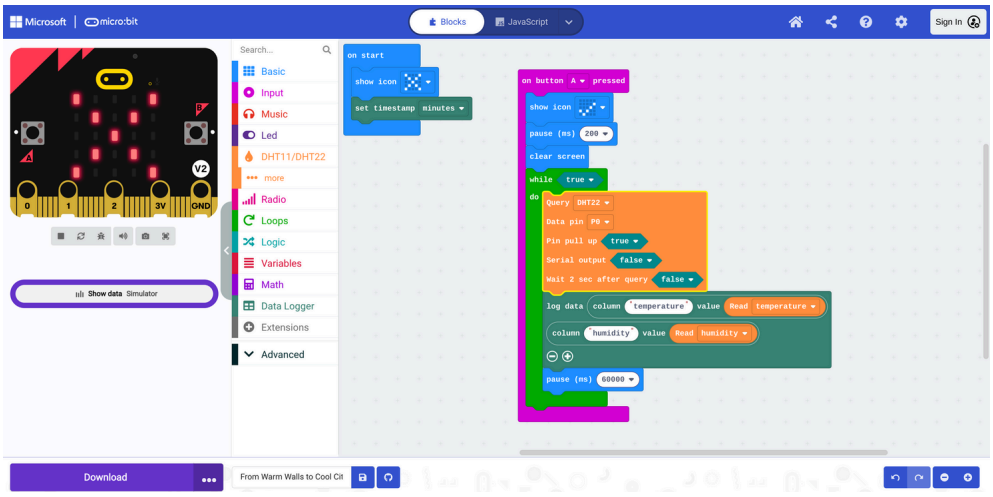
3 **L'extension datalogger apparaît dans la liste des blocs disponibles depuis votre éditeur.**



4 **Le datalogger vous permet d'accéder à un simulateur d'enregistrement de données.**

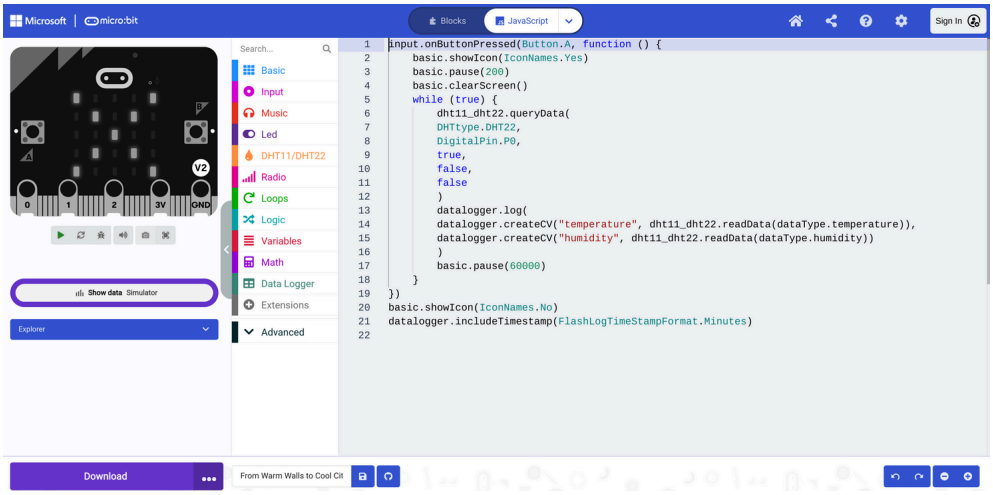
Ensuite, vous pouvez commencer à organiser les blocs en suivant le code fourni ci-contre (ajout d'une boucle infinie, enregistrement des données dans le datalogger, etc.).

Il est également possible de copier-coller directement le code dans l'éditeur **JavaScript**.



Une fois que votre programme fonctionne correctement dans le simulateur, transférez-le sur votre Micro:bit : cliquez sur « **Télécharger** » dans MakeCode pour générer un fichier **.hex**.

Ce fichier contient le programme compilé qui permettra à la carte de fonctionner.



Copiez le fichier **.hex** depuis votre dossier de téléchargement vers le lecteur amovible **« MICROBIT »**.

Une fois le fichier copié, la carte redémarrera automatiquement et exécutera le code.

Étape 3 : Positionner la Micro:bit et commencer à enregistrer les données

Une fois programmé, placez la Micro:bit dans un endroit où il pourra mesurer l'humidité et la température **sans obstruction**, afin de garantir des relevés fiables. Utilisez un ordinateur ou une **batterie externe** pour assurer l'alimentation continue du Micro:bit pendant toute la durée de l'enregistrement. Avant d'aller vous coucher, **appuyez sur le bouton A** de la carte Micro:bit pour démarrer l'enregistrement des données.

Étape 4 : Récupération des données et préparation de la carte pour la prochaine session

Chaque matin, pour éviter toute perte de données, **débranchez la Micro:bit de sa source d'alimentation** afin d'arrêter l'enregistrement. **Connectez-la ensuite à votre ordinateur** pour accéder au fichier généré pendant la nuit par le datalogger. Ce fichier s'appelle **MY_DATA.HTM** et se trouve sur le lecteur **MICROBIT**.

- **Copiez ce fichier** sur votre ordinateur.
- **Renommez-le** avec la date du jour et un identifiant clair (par exemple : BOARD1_NAME_YYYY-MM-DD.HTM).
- Une fois copié et renommé, **supprimez le fichier MY_DATA.HTM** de la carte Micro:bit pour libérer de l'espace et permettre un nouvel enregistrement.

Répétez cette opération chaque jour pour chaque carte utilisée. À la fin de la période de collecte, vous pourrez **centraliser tous les fichiers enregistrés** sur l'ensemble des Micro:bit.

Utiliser et comprendre le code

Voici le code Javascript utilisé pour programmer une carte micro:bit afin de collecter régulièrement des données sur l'humidité et la température :

```
input.onButtonPressed(Button.A, function () {
    basic.showIcon(IconNames.Yes)
    basic.pause(200)
    basic.clearScreen()
    while (true) {
        dht11_dht22.queryData(
        DHTtype.DHT22,
        DigitalPin.P0,
        true,
        false,
        false
        )
        datalogger.log(
        datalogger.createCV("temperature",dht11_dht22.readData(dataType.temperature)),
        datalogger.createCV("humidite",dht11_dht22.readData(dataType.humidity))
        )
        basic.pause(60000)
    }
})
basic.showIcon(IconNames.No)
datalogger.includeTimestamp(FlashLogTimeStampFormat.Minutes)
```

Comment le programme fonctionne ?

Ce programme mesure l'humidité et la température. **À intervalles réguliers** — par défaut toutes les minutes, mais cette fréquence peut être ajustée (toutes les 10 secondes, toutes les 5 minutes, deux fois par heure, etc.) — le programme enregistre les données dans un **datalogger**, à partir duquel il est possible de télécharger un **fichier .csv**.

Un fichier **.csv** (*Comma-Separated Values*) est un format de fichier texte utilisé pour stocker des données tabulaires, comme dans un tableau ou une feuille de calcul. Chaque ligne du fichier correspond à une ligne de données, et chaque valeur est séparée par un délimiteur — le plus souvent une virgule, mais parfois un point-virgule ou une tabulation.

Il est possible de récupérer les données d'un fichier .csv dans un tableur tel que **Excel** ou **LibreOffice Calc**. Dans Excel, ouvrez le logiciel, cliquez sur **Fichier > Ouvrir**, sélectionnez le fichier .csv, puis configurez les délimiteurs si nécessaire via l'outil d'importation. Dans LibreOffice Calc, le processus est similaire : cliquez sur **Fichier > Ouvrir**, choisissez le fichier, puis utilisez l'assistant d'importation pour définir le bon délimiteur (par exemple une virgule ou un point-virgule).

Dans les deux cas, les données s'affichent sous forme de tableau, prêtes à être analysées.

Initialisation de l'événement d'appui sur le bouton « A »: Lorsque l'utilisateur appuie sur le **bouton « A »** de la MicroBit, la fonction `input.onButtonPressed(Button.A, function () {...})` est exécutée.

Affichage de l'icône "Yes" pendant l'exécution: Avant de démarrer l'enregistrement des données, le programme affiche l'icône « **Yes** » (`basic.showIcon(IconNames.Yes)`) pendant **200 millisecondes** (0,2 seconde) pour indiquer que le processus d'enregistrement a démarré.

Pause de 200 millisecondes: Après avoir affiché l'icône « Yes », le programme attend **200 millisecondes** en utilisant `basic.pause(200)`.

Nettoyage de l'écran: Après la pause de 200 millisecondes, l'écran est effacé avec `basic.clearScreen()`, qui prépare l'écran pour ce qui suit sans être encombré d'images.

Boucle de collecte de données infinie : Le programme entre dans une boucle infinie `while (true).` Cela signifie que les données seront collectées et enregistrées sans fin jusqu'à ce que la MicroBit soit éteint ou redémarré.

Interrogation du capteur: Les blocs `dht11_dht22.queryData()` et `dht11_dht22.readData(...)` permettent de sélectionner le type de module et de lire les données du capteur (il est recommandé de respecter un délai entre chaque requête : au moins 1 seconde pour le DHT11 et 2 secondes pour le DHT22). Une requête doit être effectuée au préalable pour obtenir les valeurs de température et d'humidité. Ce bloc vérifie également la somme de contrôle des données renvoyées par le capteur. En cas d'erreur dans la somme de contrôle, les relevés de température et d'humidité retourneront -999, et le bloc « Dernière requête réussie ? » indiquera `false`.

Enregistrement des données dans le datalogger : A chaque itération, le programme enregistre les valeurs des capteurs de la MicroBit :

- **Température**: le bloc `dht11_dht22.readData(dataType.temperature),` récupère la température actuelle en degrés Celsius.
- **Humidité**: le bloc `dht11_dht22.readData(dataType.humidity)`, récupère l'humidité relative actuelle.

La température est mesurée en degrés Celsius (**°C**) et l'humidité relative en pourcentage.

Ces valeurs sont enregistrées dans le **datalogger** sous forme de variables nommes (respectivement, « température » et « humidité »). Cela se fait via la fonction `datalogger.log()` :

```
datalogger.log(
        datalogger.createCV("temperature", dht11_dht22.readData(dataType.temperature)),
        datalogger.createCV("humidite", dht11_dht22.readData(dataType.humidity))
)
```

La fonction `createCV` permet de créer un « CV » (valeur de contexte) pour chaque capteur, et la fonction `datalogger.log` permet d'enregistrer ces valeurs dans un fichier sur la MicroBit.

Pause de 60 000 millisecondes avant la lecture suivante: Après chaque enregistrement, le programme attend 60 000 millisecondes (1 minute) avant de relire les valeurs du capteur. Ceci est réalisé avec `basic.pause(60000)`.

Horodatage des données (inclus via `datalogger.includeTimestamp`) : En dehors de la fonction liée au bouton, la commande `datalogger.includeTimestamp(FlashLogTimeStampFormat.Minutes)` est utilisée pour inclure un horodatage avec chaque enregistrement de données. Le format d'horodatage est en minutes, ce qui signifie que chaque enregistrement aura un indicateur de temps basé sur les minutes écoulées depuis le démarrage du programme.

Affichage de l'icône "No" avant l'exécution: Avant que l'utilisateur n'appuie sur le bouton « A », le programme affiche une icône « No » (`basic.showIcon(IconNames.No)`) pour indiquer que la MicroBit attend l'action de l'utilisateur.