Datasheet - Programming & Sensors

Measure environmental data: light, noise, temperature

Materials and tools needed

- Micro:bit V2 board and its integrated sensors: this is the main programmable board. It includes a light sensor (via the LED screen), a sound sensor, and an integrated temperature sensor. Approximate price: around €19 per Micro:bit board. (https://www.kubii.com/fr/cartes-micro-controleurs/3091-carte-microbit-bbc-v2-5051259252585.html?mot_tcid=1436612e-e738-4468-b49f-58c52c92a4d4)
- Micro-USB cable: allows you to power the board and program it from a computer.
- External battery (optional): useful for autonomous operation if the board is detached from the computer. The official Micro:bit battery box is available for around €2.20 per unit https://www.kubii.com/fr/alimentations/4237-1913-support-de-pile-officiel-pour-microbit-3272496317253.html?mot_tcid=693572de-fca1-4287-bbd1-df4c014e258b#/appareil-sans.

You can also buy the Micro:bit V2 kit including the USB cable and the battery box for 21 EUR per kit (https://www.kubii.com/fr/kits-micro-controleurs/3092-kit-microbit-go-v2-5051259252592.html?mot_tcid=e92c2317-81d6-4102-8e90-e56faeb2fe68) or 177 EUR for 10 kits (https://www.kubii.com/fr/kits-micro-controleurs/3093-kit-microbit-club-v2-5051259252615.html?mot_tcid=97a4ea0c-3489-461e-ad35-4aec28defa2d)

For this step, we recommend programming at least 3 to 6 micro:bit boards to share among students and collect more information and data. You can do this with a single board, but you will need to either extend the overall collection period or reduce the collection period per student from 7 days to 3.

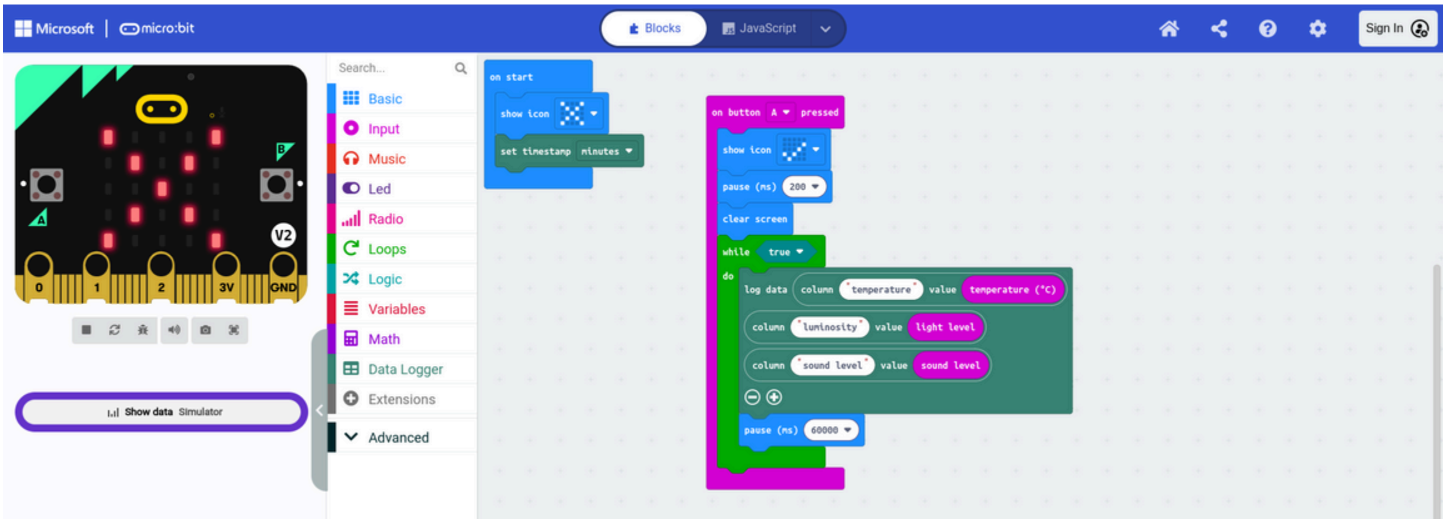
Wiring and Using a Micro:bit Board

Follow these steps to program, place, record, and retrieve environmental data using the micro:bit.

Step 1 - Program the Micro:bit board

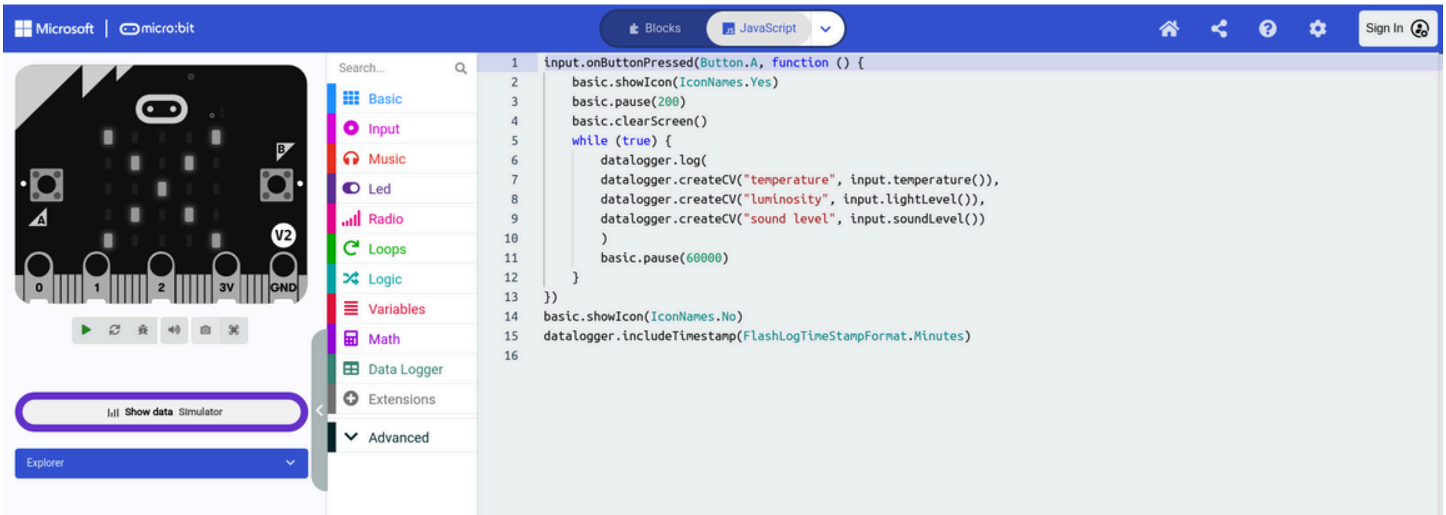
Connect the Micro:bit board: Connect the micro:bit board to the computer you created the program on using the MakeCode editor. Once connected, the micro:bit board will appear on the computer as a removable disk (e.g., "MICROBIT"). Write the program: Open the MakeCode editor to create a program that collects noise data using the built-in sensors of the Micro:bit V2 programming board. Give your project a clear name before you begin. Once in the editor, after creating your new project, you will be taken to the default "out of the box" screen and will need to install an extension. Extensions in MakeCode are groups of code blocks that are not directly included in the base MakeCode code blocks. Extensions, as the name suggests, add blocks for specific functionality. There are extensions for a wide range of very useful features, adding gamepad, keyboard, mouse, servo, and robotics capabilities, and much more. In the block display columns, click the EXTENSIONS button. In the list of available extensions, find the Datalogger extension that will be used for this activity. Click the extension you want to use and a new block group will appear on the main screen.

Then you can start organizing your blocks by following the code provided below (add an infinite loop, save the data to the datalogger...).



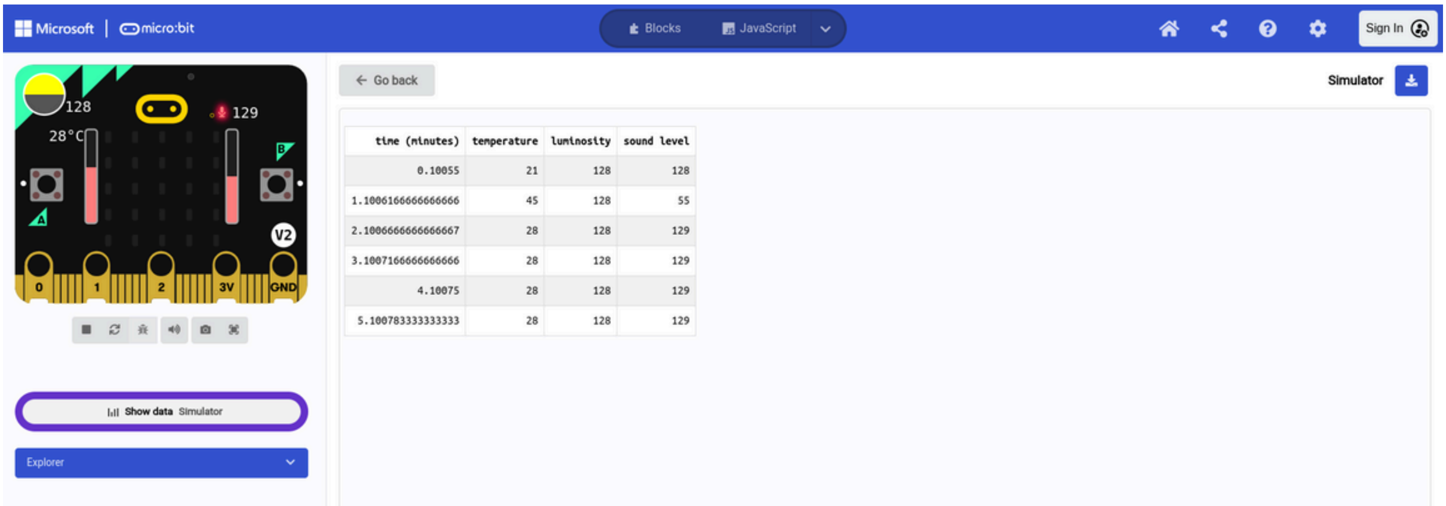
```
on start
    show icon
    set timestamp minutes

on button A pressed
    show icon
    pause (ms) 200
    clear screen
    while true
    do
        log data column "temperature" value temperature (°C)
                 column "luminosity" value light level
                 column "sound level" value sound level
        pause (ms) 60000
```

You can also copy and paste the code into the Javascript editor.



```javascript
input.onButtonPressed(Button.A, function () {
    basic.showIcon(IconNames.Yes)
    basic.pause(200)
    basic.clearScreen()
    while (true) {
        datalogger.log(
        datalogger.createCV("temperature", input.temperature()),
        datalogger.createCV("luminosity", input.lightLevel()),
        datalogger.createCV("sound level", input.soundLevel())
        )
        basic.pause(60000)
    }
})
basic.showIcon(IconNames.No)
datalogger.includeTimestamp(FlashLogTimeStampFormat.Minutes)
```

Test the program using the MakeCode simulator.



time (minutes)	temperature	luminosity	sound level
0.10055	21	128	128
1.1006166666666666	45	128	55
2.1006666666666667	28	128	129
3.1007166666666666	28	128	129
4.10075	28	128	129
5.100783333333333	28	128	129

Once your program is working correctly on the simulator, transfer it to your Micro:bit: click “Upload” in MakeCode to generate a .hex file. This file contains the compiled program that will enable the board to work.

Copy the .hex file from your download folder to the “MICROBIT” removable drive.

Once the file is copied, the board will automatically reboot and execute the code.

Step 2: Place the Micro:bit and start recording data

Once programmed, place the micro:bit to collect the data you need—near your bed in an area where it can accurately record light, noise, and temperature without obstruction. Use a computer or power bank to continuously power the micro:bit while recording. Make sure the board is placed in exactly the same position each night to record comparable data.

Before going to bed, press the "A" button on the MicroBit to start recording data via the program.

Step 3: Retrieve data and prepare the board for the next recording session

Every morning, to avoid any data loss, we recommend that you unplug the micro:bit from its power source to stop data recording and connect the micro:bit to your computer to access the file compiled overnight by the datalogger (which will be called “MY_DATA.HTM, available on the micro:bit reader).

Copy this file to your computer and rename it with today's date (e.g., BOARD1_NAME_YYYY-MM-DD.HTM).

After copying and renaming the file, delete the MY_DATA.HTM file from the MicroBit board to free up space and allow for new data recording.

Repeat the process for the next session, that is, the next evening before going to bed.

At the end of the collection period, you will be able to retrieve all the files collected on the various micro:bit boards. Once opened, the data files will be accessible in HTML format. They will provide all the collected data and allow you to download them in .csv format.

Use and understand the code

Here is the JavaScript code used to program a micro:bit board to regularly collect light, noise, and temperature data:

```
input.onButtonPressed(Button.A, function () {
    basic.showIcon(IconNames.Yes)
    basic.pause(200)
    basic.clearScreen()
    while (true) {
        datalogger.log(
        datalogger.createCV("temperature", input.temperature()),
        datalogger.createCV("luminosite", input.lightLevel()),
        datalogger.createCV("niveau sonore", input.soundLevel())
        )
        basic.pause(60000)
    }
})
basic.showIcon(IconNames.No)
    datalogger.includeTimestamp(FlashLogTimeStampFormat.Minutes)
```

How does the code work?

This program measures ambient noise, temperature, and brightness. Every minute (the interval can be changed to 10 seconds, 5 minutes, twice an hour, etc.), the program compiles the information into a data logger, from which a .csv file can be downloaded.

A .csv (Comma-Separated Values) file is a text file format used to store tabular data (such as in a table or spreadsheet). Each line in the file represents a row of data, and each value within a row is separated by a delimiter (often a comma, but sometimes a semicolon or a tab). It is possible to retrieve data from a .csv file into a spreadsheet program such as Excel or LibreOffice Calc. In Excel, open the program, click File > Open, select the .csv file, and configure delimiters if necessary using the import tool. In LibreOffice Calc, follow a similar process: click File > Open, select the file, and use the import wizard to choose the delimiter (for example, comma or semicolon). In either case, the data is displayed in table format, ready for analysis.

Initializing the button "A" press event: When the user presses the "A" button on the MicroBit, the function input.onButtonPressed(Button.A, function () {...}) is triggered.

Displaying the "Yes" icon during execution: Before starting data recording, the program displays the "Yes" icon (basic.showIcon(IconNames.Yes)) for 200 milliseconds (0.2 seconds) to indicate that the recording process has started.

Pause for 200 milliseconds: After displaying the "Yes" icon, the program waits for 200 milliseconds using basic.pause(200).

Clear Screen: After the 200 millisecond pause, the screen is cleared with basic.clearScreen(), which prepares the screen for what follows without being cluttered with images.

Infinite data collection loop: The program enters an infinite while (true) loop. This means that data will be collected and recorded endlessly until the MicroBit is turned off or restarted.

Recording data in the datalogger: At each loop iteration, the program records the values of the MicroBit sensors:

- temperature: input.temperature(), which retrieves the current temperature in degrees Celsius.
- luminosite: input.lightLevel(), which measures the ambient light level.
- sound level: input.soundLevel(), which captures the ambient sound level.

Sound level and light level measure a relative value and do not have standard units like decibels (dB) for sound level or lux (lx) for brightness. More precisely, the sensor measures the perceived intensity. This value is a numerical estimate (from 0 to 255), where 0 represents the minimum value (complete silence/total darkness) and 255 the maximum value (very loud noise/intense light). Temperature is measured in degrees Celsius (°C). These values are recorded in the datalogger as variables with respective names ("temperature", "brightness", "sound level"). This is done via the datalogger.log() function:

```
datalogger.log(
datalogger.createCV("temperature", input.temperature()),
datalogger.createCV("luminosite", input.lightLevel()),
datalogger.createCV("sound level", input.soundLevel())
)
```

The createCV function creates a "CV" (context value) for each sensor, and the datalogger.log function saves these values to a file on the MicroBit.

60,000 millisecond pause before next reading: After each recording, the program waits 60,000 milliseconds (1 minute) before reading the sensor values again. This is achieved with basic.pause(60000). You can change the pause duration to capture more or less data (e.g., every minute).

Data timestamp (included via datalogger.includeTimestamp): Apart from the button-related function, the datalogger.includeTimestamp(FlashLogTimeStampFormat.Minutes) command is used to include a timestamp with each data record. The timestamp format is in minutes, meaning each record will have a time indicator based on the minutes elapsed since the program was started.

Displaying the "No" icon before execution: Before the user presses the "A" button, the program displays a "No" icon (basic.showIcon(IconNames.No)) to indicate that the MicroBit is waiting for user action.