Datasheet - Programming & Sensors

Collect data with the humidity and temperature sensor

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)

- DHT22 (or DHT11) Sensor: These sensors are popular for measuring humidity and temperature with microcontrollers. The DHT11 is inexpensive and sufficient for simple projects, while the DHT22 offers better accuracy and higher resolution, for a slightly higher cost.
- Computer or tablet: Used to write code and transfer it to the Micro:bit.
- Programming environment: The online editor MakeCode is recommended for easy programming of the Micro:bit board.

For this step, it is recommended to program between 3 and 6 Micro:bit boards in order to distribute them among the students and collect a larger volume of data. It is possible to carry out the activity with a single board, but this will require either extending the overall collection period or reducing the collection time per student, from approximately 7 days to 3 days.

Wiring and Using a Micro:bit Board

Follow the steps below to program, install, record, and retrieve environmental data using a Micro:bit board.

Step 1: Wiring the temperature/humidity sensor to the Micro:bit board.

There are two types of DHT11/DHT22 sensors:

1. Version without PCB board, with 4 pins;
2. PCB mounted version, with integrated pull-up resistors and only 3 pins.

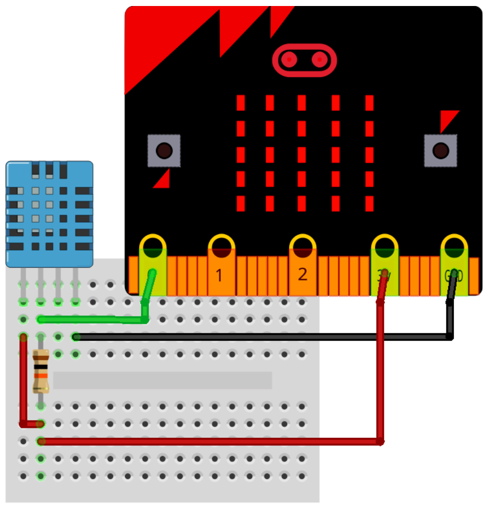
We recommend using the PCB version, which is easier to connect. For the PCB version (3 pins):

- Vcc (+): connect to 3.3 V or 5 V (both voltages are compatible)
- GND (-): to be connected to ground (GND)
- Data (OUT): connect to any GPIO pin of the Micro:bit

For the version without PCB (4 pins):

- You need to add a pull-up resistor between Vcc and Data to keep the Data pin high. A resistor between 220 Ω and 10 kΩ works well at 3.3 V; above that, the sensor may not respond.
- You can also use the Micro:bit's internal pull-up option: in MakeCode, go to "Pin" > "More" > "Set lever to pin...". The Micro:bit has internal pull-up resistors of about 12–13 kΩ.
- Note: The third pin from the left (on the 4-pin version) is not used.

Connect your DHT sensor following the diagram opposite.



Step 2: Programming the Micro:bit.

Using your USB cable, connect the board to your computer via the micro-USB connector. Once connected, the Micro:bit board will appear on your computer as a removable drive (e.g., “MICROBIT”). Open the MakeCode editor to create a program that collects light, noise, and temperature data using the Micro:bit V2’s built-in sensors. Give your project a clear name before you begin.

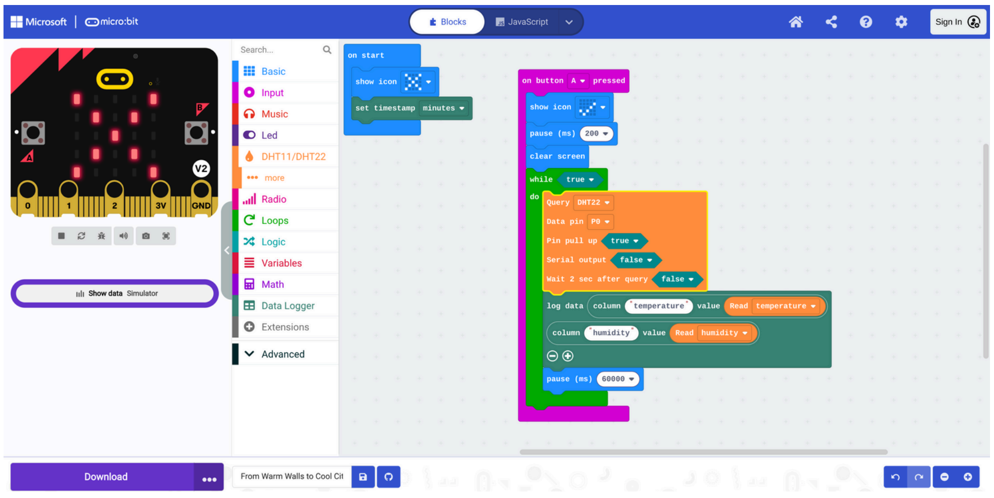
Once in the editor and after creating your new project, you will see the default "ready to use" screen. You will then need to install an extension. Extensions in MakeCode are groups of blocks that are not directly included in the basic blocks. As their name suggests, they add blocks for specific functionalities. There are extensions for a wide range of uses: creating a gamepad, a keyboard, a mouse, controlling a servo motor, etc. In the block groups column, click on EXTENSIONS. In the list of available extensions, find the Datalogger extension, which will be used for this activity. Click on the desired extension: a new block group will appear on the main screen. Do the same for the temperature/humidity sensor by searching for the DHT11/DHT22 extension.

Next, you can start organizing the blocks by following the code provided opposite (adding an infinite loop, saving data to the datalogger, etc.). It is also possible to copy and paste the code directly into the JavaScript editor.

Once your program is working correctly in 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.

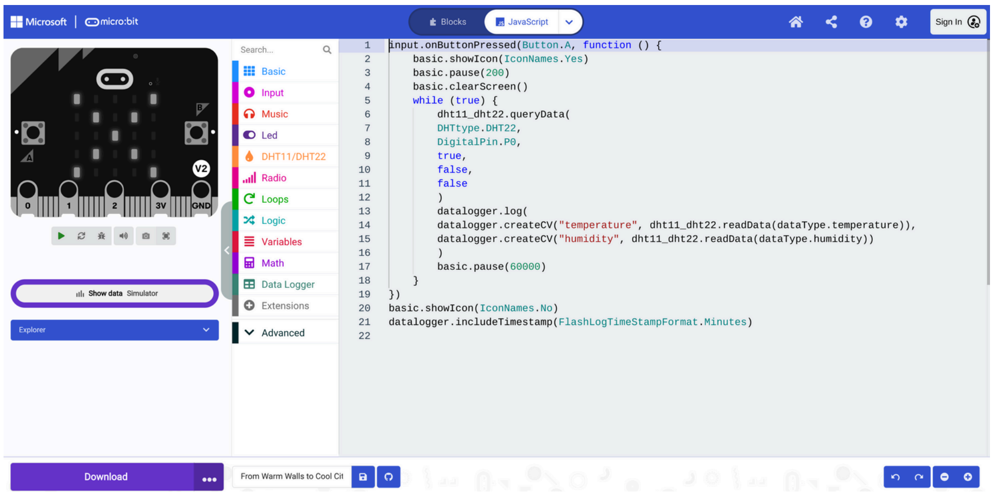
Copy the .hex file from your download folder to the "MICROBIT" removable drive.

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



```
on start
    show icon
    set timestamp minutes

on button A pressed
    show icon
    pause (ms) 200
    clear screen
    while true
    do
        Query DHT22
        Data pin P0
        Pin pull up true
        Serial output false
        Wait 2 sec after query false
        log data column "temperature" value Read temperature
                 column "humidity" value Read humidity
        pause (ms) 60000
```



```
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("humidity", dht11_dht22.readData(dataType.humidity))
        )
        basic.pause(60000)
    }
})
basic.showIcon(IconNames.No)
datalogger.includeTimestamp(FlashLogTimeStampFormat.Minutes)
```

<u>Step 3: Position the Micro:bit and start recording data</u>

Once programmed, place the Micro:bit in a location where it can measure humidity and temperature without obstruction to ensure reliable readings. Use a computer or power bank to ensure continuous power for the Micro:bit while recording. Before going to bed, press button A on the Micro:bit board to start data logging.

<u>Step 4: Retrieving data and preparing the map for the next session</u>

Every morning, to prevent data loss, unplug the Micro:bit from its power source to stop recording. Then connect it to your computer to access the file generated overnight by the datalogger. This file is called MY_DATA.HTM and is located on the MICROBIT drive.

- Copy this file to your computer.
- Rename it with today's date and a clear identifier (for example: BOARD1_NAME_YYYY-MM-DD.HTM).
- Once copied and renamed, delete the MY_DATA.HTM file from the Micro:bit board to free up space and allow for a new save.

Repeat this process every day for each board used. At the end of the collection period, you can centralize all the files saved on all Micro:bits.

Use and understand the code

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

```
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)
```

How does the program work?

This program measures humidity and temperature. At regular intervals—by default, every minute, but this frequency can be adjusted (every 10 seconds, every 5 minutes, twice an hour, etc.)—the program records the data in a datalogger, from which it is possible to download a .csv file.

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 corresponds to a row of data, and each value is separated by a delimiter—most often a comma, but sometimes a semicolon or 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 software, click File > Open, select the .csv file, and then configure the delimiters if necessary using the import tool. In LibreOffice Calc, the process is similar: click File > Open, choose the file, and then use the import wizard to set the correct delimiter (for example, a comma or a semicolon).

In both cases, 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 executed.

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).

Clearing the 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.

Sensor Query: The dht11_dht22.queryData() and dht11_dht22.readData(...) blocks are used to select the module type and read the sensor data (it is recommended to respect a delay between each query: at least 1 second for the DHT11 and 2 seconds for the DHT22). A query must be made beforehand to obtain the temperature and humidity values. This block also verifies the checksum of the data returned by the sensor. If there is an error in the checksum, the temperature and humidity readings will return -999, and the "Last query successful?" block will return false.

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

- Temperature: The dht11_dht22.readData(dataType.temperature) block retrieves the current temperature in degrees Celsius.
- Humidity: The dht11_dht22.readData(dataType.humidity) block retrieves the current relative humidity.

Temperature is measured in degrees Celsius (°C) and relative humidity in percentage.

These values are recorded in the datalogger as named variables (respectively, "temperature" and "humidity"). This is done via the datalogger.log() function:

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

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.

Pause for 60,000 milliseconds before reading again: After each recording, the program waits 60,000 milliseconds (1 minute) before reading the sensor values again. This is achieved with basic.pause(60000).

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 "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.