

# The micro:bit micro-controller

## UBMS Snap! Programming Summer 2023

July 10, 2023

### Tail of the Finch: the micro:bit

- The "brain" of the Finch 2.0 robot is a *microcontroller*, a small single-chip computer in an *embedded system* to control a single function or a set of functions, the BBC micro:bit.
- Even cooler than the device is the fact that you can simulate the micro:bit online - here is the Python editor: [python.microbit.org](https://python.microbit.org).
- But the easiest way to program the micro:bit (and, by implication, also the robot) is with Microsoft's MakeCode, which is like a micro-version of Scratch or Snap! [makecode.microbit.org/](https://makecode.microbit.org/).

### Demonstration: MicroPython editor

- Open [python.microbit.org](https://python.microbit.org) on the editor/simulator/debugger:
- This is actually not "Python" but "MicroPython" - a Python version for microcontrollers written in C++ (info).
- Parts of this platform interface:
  1. sidebar with reference, mini-project ideas, API (data)
  2. editor pane for Python scripting
  3. simulator for simulating a physical micro:bit
  4. extractor for sending .hex file to micro:bit
  5. menu to save .py or .hex files to your PC
- The editor opens to a sample animation on the display:



Figure 1: micro:bit v2 in the tail of a Finch 2.0 robot Encased micro:bit v1



Figure 2: Encased micro:bit v1

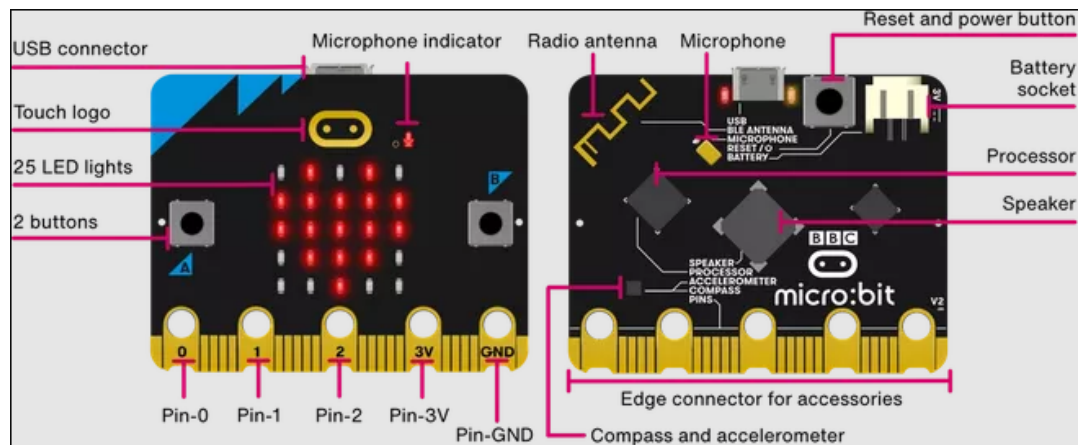


Figure 3: micro:bit v2 front and back

## Get creative with the micro:bit Python Editor

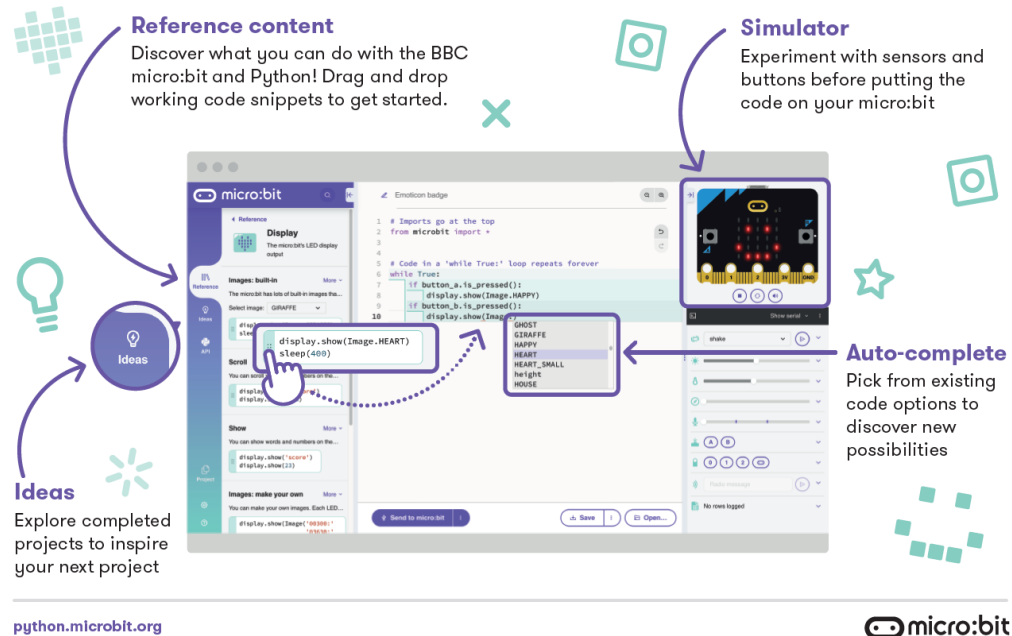


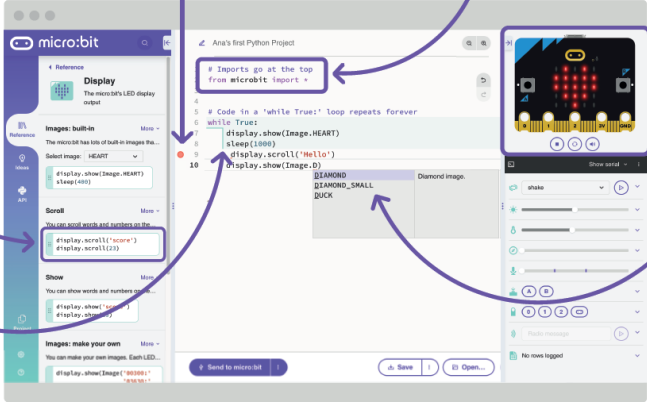
Figure 4: Microbit Python editor and debugger (source: microbit.org)

## Debugging with the micro:bit Python Editor

**Reference code snippets**  
Drag and drop working code snippets to develop your coding and improve accuracy

**Syntax errors highlighted with a red dot**  
Hover over the dot to see a helpful message

**Import micro:bit library**  
Always keep `'from microbit import *'` at the start of your program.  
*This imports the library that allows access to all the amazing sensors, inputs, and outputs on the micro:bit.*



[python.microbit.org](https://python.microbit.org)


 micro:bit

Figure 5: Microbit Python editor and debugger (source: microbit.org)

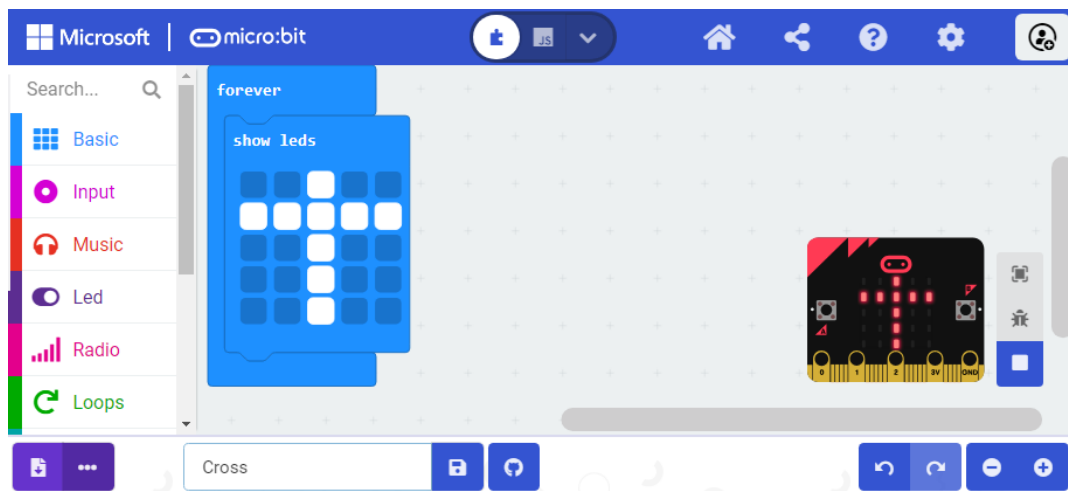


Figure 6: Microbit Microsoft Makecode editor (and debugger)

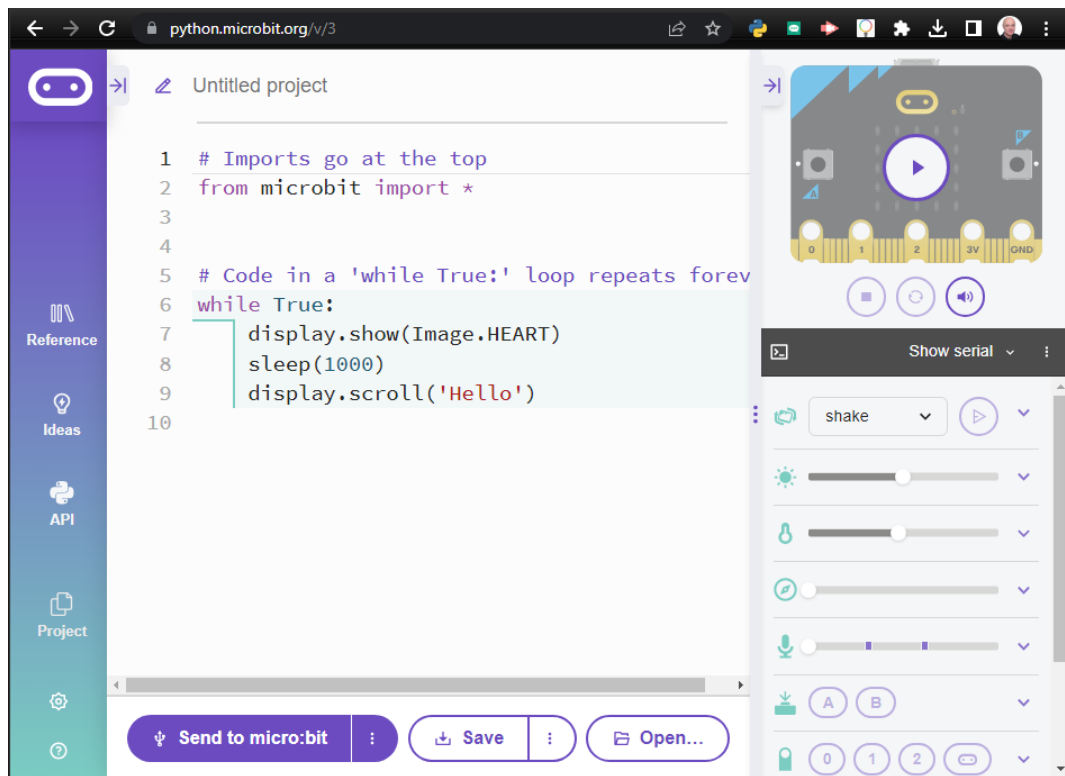


Figure 7: micro:bit Python editor v3

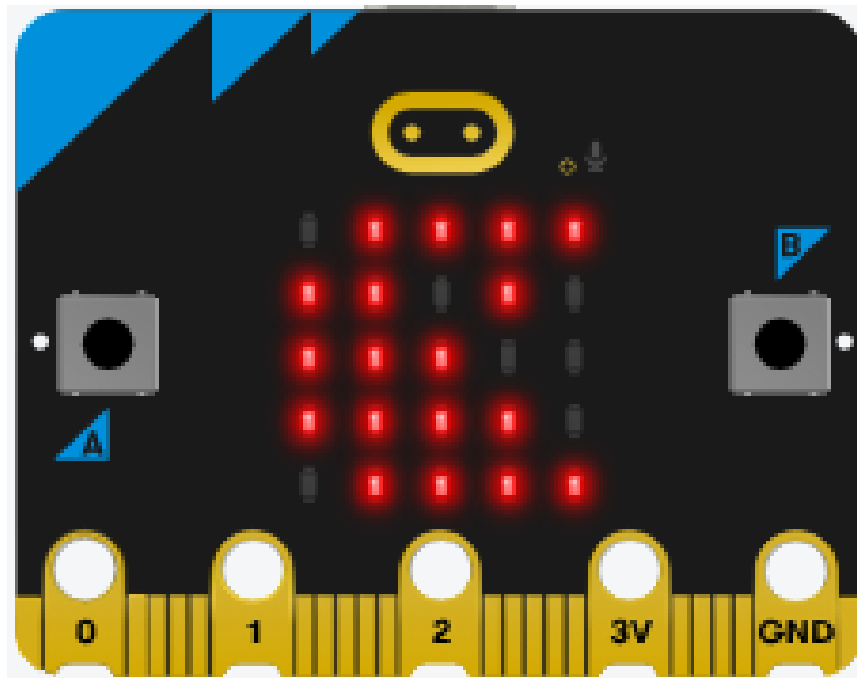
```

from microbit import *
while True:
    display.show(Image.HEART)
    sleep(1000)
    display.scroll('Hello')

```

- After importing the `microbit` library, these functions are called:
  1. `display.show` to show images on the 5 x 5 diode display
  2. `sleep` to pause for a given number of microseconds ( $100 \mu s = 1 s$ )
  3. `display.scroll` to scroll a string argument on the display

## Practice 1: Pacman



- Open the simulator and run it for the sample program that appears when you open [python.microbit.org](https://python.microbit.org).
- Change the image from `HEART` to `pacman`. Run the simulator again and check the display for the debugging message.

- Open the **Reference > Display** in the sidebar and find the correct image descriptor for "pacman". Fix the code and run it on the simulator.

## Practice 2: Alarm Clock with four hands

- Look for **Display > Images**: make your own in the reference manual.
- Create a clock that moves a hand clock-wise changing position once per second, and that makes a 'Ba-ding' sound when the hand changes.
- Solution video (micro:bit v1).
- Solution code (Python).

## Practice 3: Frère Jacques



1. Open the **Ideas > Frère Jacques** tab in the sidebar.
2. Load the first code shown by clicking on **Open** below the code block.
3. In the simulator, run the code.



4. In the code editor, complete the code for the lullaby. If you can't read music, this is the sequence of the notes shown in the image:

F G A F - F G A F - A B C - A B C - C D C B A F - C D C B A F - F C F - F C F

The black crotchets are quarter length (:4), the open crotchets are 2 x quarter length (:8), and the crotchets with the bars between them are 1/2 quarter length (:2).

5. Check using the simulator that you succeeded.
6. Save the final code as .hex and .py on your PC.

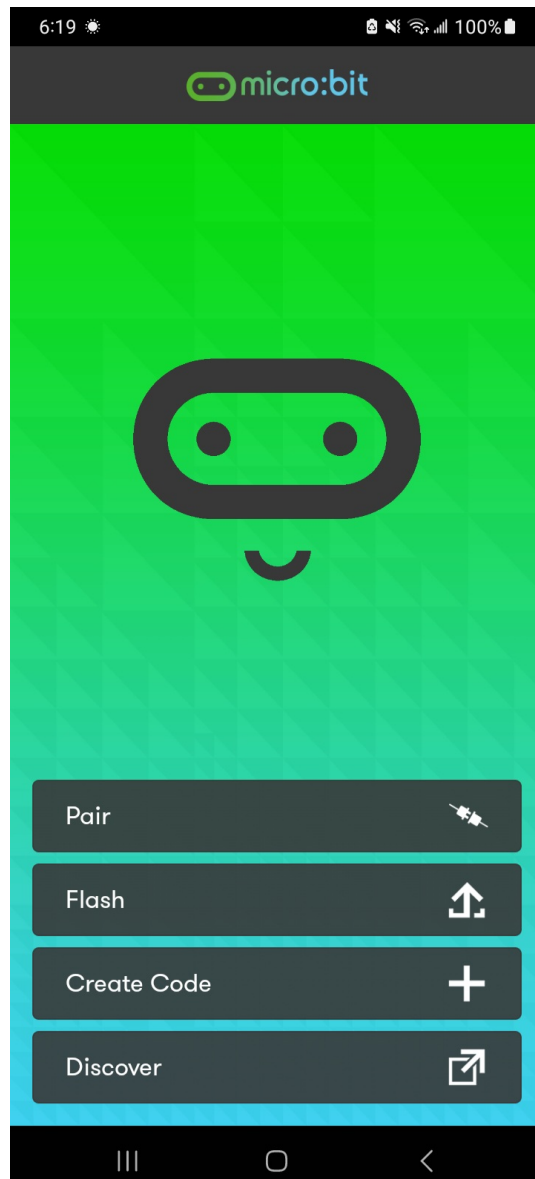
Solutions:

- video with Finch,
- video with micro-bit v1
- Python code

## More activities with the BBC micro:bit

### Also on the [Android] phone

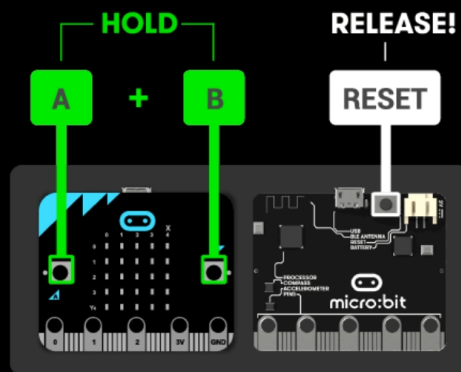
Samsung app for micro:bit (also other Android phones):



8:37

100%

## How to pair your micro:bit



### Step 1

Hold the A and B buttons then press and release RESET. Keep holding A and B until the screen fills up

Let's do this

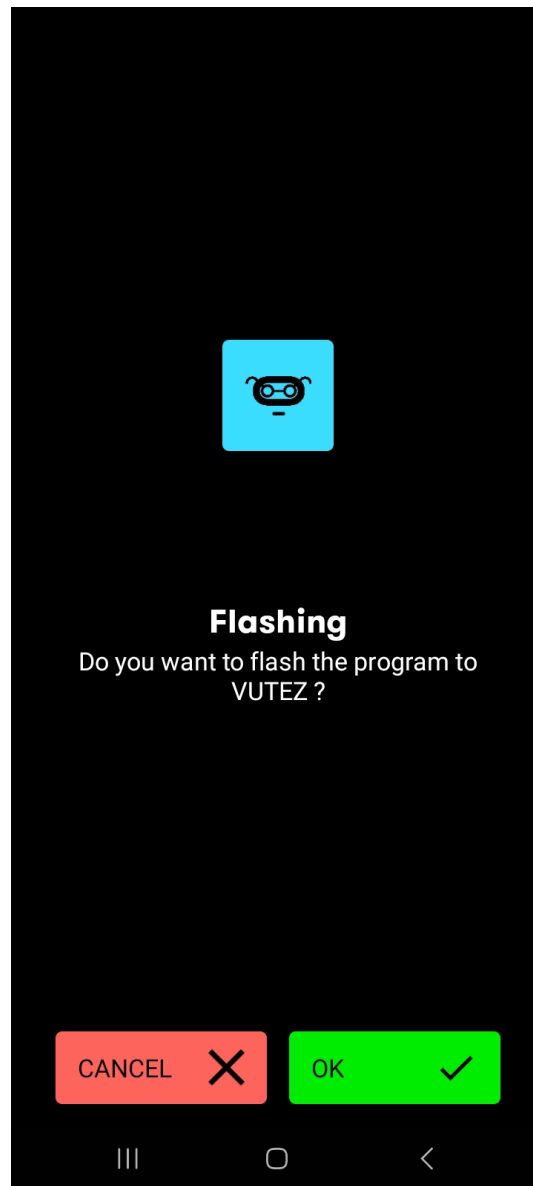


CANCEL



NEXT





- Pair with any micro:bit.
- Code in MakeCode, JavaScript or Python.
- Simulate code in app and flash to micro:bit.