The BBC micro:bit - from the UK to the World

[Extended Abstract]

Lancaster University

Micro:bit Educational Foundation

Microsoft Research

ABSTRACT

The micro:bit rocks!

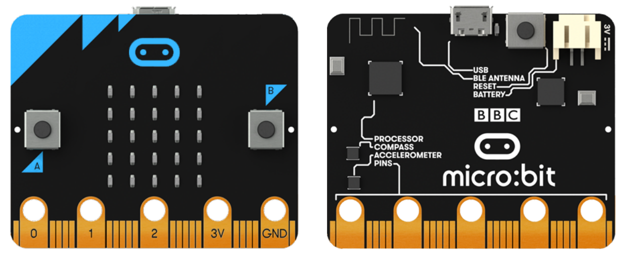
1. INTRODUCTION

The micro:bit is a small programmable and embeddable computer designed, developed and deployed by the BBC and partners (including ARM, Microsoft and Lancaster University) to approximately 800,000 UK middle school students in 2015-2016. Part of the BBC's Make It Digital Campaign, the BBC described the micro:bit as its "most ambitious education initiative in 30 years, with an ambition to inspire digital creativity and develop a new generation of tech pioneers." [**?**]

Figure 1 shows (a) the front and (b) the back of the micro:bit, which measures 4cm x 5cm. Like the Arduino Uno, the micro:bit is a single-board microcontroller that can be programmed via a host computer (usually a laptop or desktop) and then embedded in projects where it runs on battery power. In contrast to the Uno, which has no built-in sensors, the micro:bit board hosts a variety of sensors (temperature, accelerometer, magnetometer, light level), a 5x5 LED matrix, two user-defined buttons, as well as Bluetooth Low Energy (BLE) communications.[1].

The design of the micro:bit hardware was driven by the first two objectives of the BBC micro:bit project: (B1) to provide a simple creative experience for physical computing, wearable and Internet of Things (IoT) projects; (B2) to supply a device that can continue to provide learning opportunities as the user's expertise grows.

On the hardware side, the micro:bit's built-in sensors, buttons and LED display allow many projects to be completed with no additional hardware or wiring. The holes on micro:bit's edge connector allows additional external sensors and actuators to be connected via crocodile clips. The micro:bit's BLE capabilities introduces networking to the picture, and enables streaming of data and command/control operations among the micro:bit, smartphones, laptops, as well as other micro:bits. As with Arduino, an ecosytem of micro:bit shields (hardware peripherals) that accommodate the micro:bit's edge connector expands its capabilities greatly.[2]



(a) (b)

Figure 1: The micro:bit: (a) front, with two buttons, 5x5 LED display, and edge connector (bottom); (b) back, with processor, accelerometer, compass, Bluetooth, USB and battery ports.

The design of the micro:bit coding tools also was oriented towards a simple starting experience with room for progression. In particular, the coding objectives of the project were: (B3) to give students an exciting, engaging introduction to coding; (B4) to stimulate curiosity about how computing technologies can be utilized to solve problems that students identify.

Based on user trials with a micro:bit prototype with students in Years 5 and 7 (3rd and 5th grade in the US, respectively), the BBC focused on delivering a web app based on the popular Blockly framework [1] to permit students to create scripts via drag-and-drop operations in a web browser, and see the execution of their scripts via a simulator. Text-based coding via scripting languages also was identified as an important feature. As the micro:bit would be incorporated into standalone projects, it also was essential for the user's program to be compiled and installed in non-volatile storage on the micro:bit where it could be run via battery power.

The solution delivered by the BBC's partners evolved from the initial design to include:

- support for Blockly, JavaScript, Python and C++;
- an efficient C++ runtime for the micro:bit created by Lancaster University;
- a web app (`http://makecode.microbit.org`) with Blockly

[1]The micro:bit has a whopping 16kB of RAM and 256kB of Flash memory, compared to the Uno's 2kB of RAM and 32kB of Flash

[2] `http://microbit.org/assets/documents/AccessoryGuideSummer18.pdf`

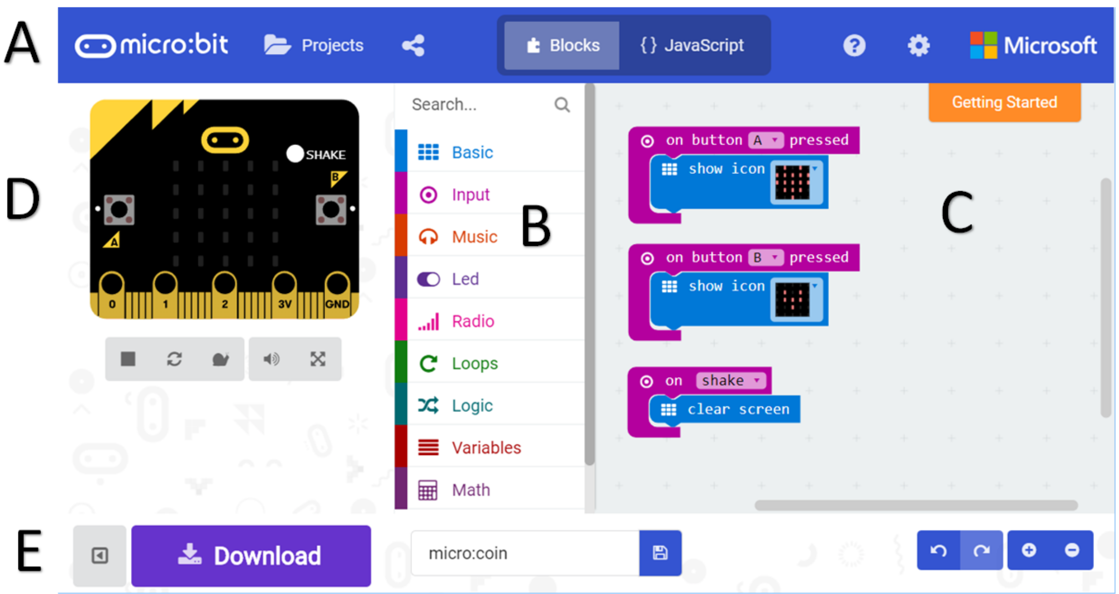


Figure 2: MakeCode web app for the micro:bit

and JavaScript editors, micro:bit simulator, and a compiler to machine code, linked against a pre-compiled C++ runtime;

- a Python compiler and read-eval-print loop (REPL) that resides *on the micro:bit* (via `https://micropython.org/`), supported by a simple web app (`http://python.microbit.org`) and an installable application (`https://codewith.mu/`);

- ARM's DAPlink firmware makes the micro:bit appear as USB pen drive on most operating systems, enabling a simple file copy operation to install a user's program on the micro:bit (no device drivers needed).

MakeCode, MicroPython, and the C++ runtime are all open source.[3]

Figure 2 shows a screen snapshot of the MakeCode web app for the micro:bit with five main sections: (A) menu bar with access to projects and examples, and switching between Blockly and JavaScript editors; (B) Blockly toolbox of micro:bit API categories; (C) Blockly programminng canvas with a simple reactive program; (D) micro:bit simulator for execution of the user's program in browser; (E) download button, which invokes the in-browser compiler to produce a binary executable.

The event-based program shown in section (C) displays a large heart when the A button is pressed, displays a small heart when button B is pressed, and clears the display when the user shakes the micro:bit (shake detection is implemented using the accelerometer; in the simulator, the shake event is fired using a virtual button). In addition to event-based APIs, direct access to the micro:bit's sensors via polling is possible. [takes a few minutes to code and deploy this simple program]

The BBC micro:bit project also called for partners to develop content and to "train the trainers" (educators) around the micro:bit computing system.

In the remainder of this paper, we focus on the primary promise of the BBC micro:bit, which was to deliver a simple physical computing experience for beginners and a progression path for users to follow as their expectations increase. [micro:bit education foundation founded in the fall of 2016] We draw from two full years of full deployment of the micro:bit in the UK, as well as deployments in Europe, the United States, and Asia. There are approximately two million micro:bits now in the market and many hardware, content, and education partners participating.

Overview: Section 2 on physical computing;

2. CONTEXT: PHYSICAL COMPUTING

As discussed in the Introduction, the micro:bit is a device with similarities to the Arduino family of printed circuit boards. Such *physical computing* devices are designed to be placed in and interact with our physical environment. Physical computing lives in the spaces between computing and many other disciplines: art, industrial design, health, environmental monitoring; it has close ties to cyber-physical systems, embedded systems, and IoT. The National Science Foundation defines cyber-physical systems as those that "integrate sensing, computation, control and networking into physical objects and infrastructure, connecting them to the Internet and to each other."[2]

The benefits of using physical computing as an introduction to computing include:

- *broad reach* because of diverse applications of physical computing – leverage fine arts, music, design, etc. in projects;

- *increased motivation* because of tangible visible out-

[3] At `https://github.com/microsoft/pxt`, `https://github.com/micropython/micropython`, and `https://github.com/lancaster-university/microbit-dal`, respectively.

come (rather than virtual on screen);

- *learning by doing* as there are many ways to achieve goal (no single correct solution)
- *natural division of labor* for more complex projects (design, hardware, software, ...)
- *full system view of computing*: hardware and software working together.

2.1 Wiring and Arduino

To help explain the BBC micro:bit, it's very instructive to understand Hernando Barragan's 2003 Master's thesis, "Wiring: Prototyping Physical Interaction Design", the inspiration for the Arduino system [?]. His objective was to make it easier for non-technical creators, such as artists and designers, to leverage electronics in their their work by simplifying the hardware and programming experience. In particular, he said of existing work: "Current prototyping tools for electronics and programming are mostly targeted to engineering, robotics and technical audiences." Of Wiring's design, he identified the following key concepts:

- a simple cross-platform integrated development environment (IDE) to create so-called "sketches";
- simplified application programming interfaces (APIs) to access a microcontroller's resources;
- leverage open source compiler/linker toolchain, transparent to the end user;
- a bootloader to make it easy to upload a compiled sketch to the microcontroller;

Also key to Wiring was openness of both the hardware and software comprising the system.

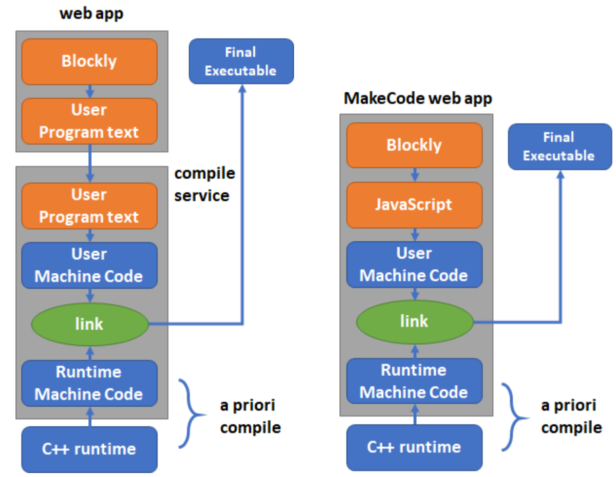
But, still some issues:

- reliance on the C language and C compiler (needs to be installed)
- very poor experience in IDE
- USB bootloader requires device drivers on some systems

2.2 The BBC micro:bit

Main points:

- *A Visible Computer*: BBC micro:bit inherits the raw PCB nature of Arduino (everything is visible to the end user).
- *No Wiring*: makes starting easy
- *Small Size*:
- *Scripting via Web App*: XYZ
- *No Install*: XYZ As shown in Figure 3(a), in the BBC design the text of a user's program (whether derived from Blockly or produced directly by the user) is submitted to a compile service that returns a final executable to be copied onto a micro:bit (connected to the host computer by USB) via a specialized loader application. avoiding the need for a compile service for user code (as shown in Figure 3(b));



(a) (b)

Figure 3: Web and compiler designs: (a) initial BBC design; (b) final design, as implemented in MakeCode.

- *Extensible*: via edge connector and layered APIS (package system too).

From this perspective, the micro:bit can be seen as a starter device for physical computing, embedded systems and cyber-physical systems, as it has sensing, computation, control and networking capabilities built in. The micro:bit is not properly an IoT device, having no built-in way to connect over IP, but it can be connected to other devices with IP connectivity.

3. PROJECTS

In this section we present a sampling of projects that illustrate the capabilities of the micro:bit.

3.1 React to an event

3.2 Embed for STEM

3.3 Communicate over radio

3.4 Control actuator

4. PLATFORM

4.1 Hardware

4.2 C++ Runtime

4.3 Microsoft MakeCode

4.4 MicroPython

5. PARTNERS

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] N. Fraser. Ten things we've learned from blockly. In *Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, BLOCKS AND BEYOND '15, pages 49–50, 2015.

[2] NSF. Cyber-physical systems. 2018.