The BBC micro:bit Coded by Microsoft Touch Develop

Thomas Ball, Jonathan Protzenko, Judith Bishop, Michał Moskal,
Jonathan de Halleux, Michael Braun
Microsoft Research
One Microsoft Way
Redmond, WA, USA
tball, protz, jbishop, micmo, jhalleux, v-braum @microsoft.com

ABSTRACT
The chance to influence the lives of a million children comes once in a generation. With the partnership between the BBC and several technology companies, a small device, the BBC micro:bit will be given to a million children in the UK in 2016. Moreover, using the micro:bit will be part of the curriculum. This demo describes the BBC micro:bit together with its software platform, based on Microsoft's Touch Develop. The demo will illustrate the architecture of the micro:bit and the software engineering hurdles that had to be overcome to enable it to be used by children. Evaluation of studies of the software platform are available and early anecdotal evidence of the hardware. A video about the micro:bit is available at aka.ms/bbcmicrobit.

CCS Concepts
• **Hardware~Sensor devices and platforms** • Applied computing~E-learning • Software and its engineering~Compilers

Keywords
K-12 education; BBC micro:bit; Touch Develop, devices, cloud

1. INTRODUCTION
Computer scientists are continually looking at new ways to engage and retain the interest of young students in the K-12 years. In recent years, there have been several waves of new initiatives to get and keep children aged 8-13 (middle school) engaged in computer science, for example: coding [6], computational thinking [10], games [11], robots [12] and storytelling [2] All of these are successful when led by dedicated and qualified teachers.

If a device can be made small enough to be cheaply distributed to millions of children, and if the accompanying software is engaging, intuitive and progressive, then there is a chance of making a leap in capturing the minds of an entire generation. The challenge is how to scale out an experiment to influence an entire country of students, or even globally. Two significant success at the coding level have been:

ICSE '16 Companion, May 14-22, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4205-6/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2889160.2889179

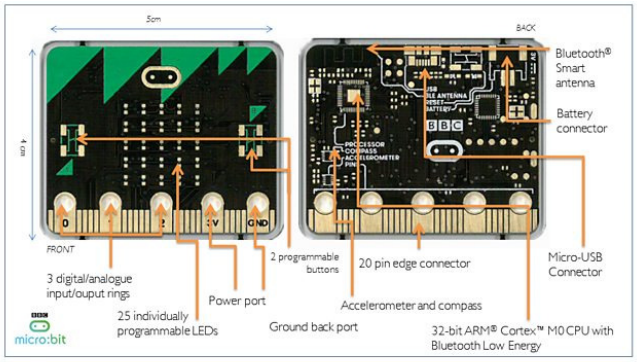

5cm
BACK
4 cm
0
1
2
3V
GND
FRONT
Bluetooth® Smart antenna
Battery connector
PROCESSOR COMPASS ACCELEROMETER PINS
USB BLE ANTENNA RESET BATTERY
BBC
2 programmable buttons
20 pin edge connector
Micro-USB Connector
3 digital/analogue input/ouput rings
Power port
Accelerometer and compass
25 individually programmable LEDs
Ground back port
32-bit ARM® Cortex™ M0 CPU with Bluetooth Low Energy
micro:bit

Figure 1 The BBC micro:bit front and back

1. code.org which initially took up the challenge of getting the K-12 students to code using a variety of online tools, and subsequently has started training teachers in the USA. [7]
2. CAS (Computing at School) in the UK is an established community of mainly teachers who create curriculum for formal computer science courses nationally [3].

There is evidence that students and children are enticed by activities where they can see, touch and change “the computer”, in addition to seeing code on a screen, as in .NET Gadgeteer [4]. The growth of interest in Arduino, Raspberry Pi and other small computers has been considerable in the developer world. However, rolling these out in schools presents the two challenges of cost and training. Gadgets, internet of things, and the maker culture are all attractive goals, but they need to be at a cost that schools can meet with a low barrier for entry in terms of skills required by teachers.

On the other hand, any initiative for large scale roll out of a new wave of computing should acknowledge that children learn and grown up very fast, so that a progression of tools within the same basic platform is highly desirable.

Taking up these challenge, a multi-partner initiative led by the BBC, the UK Department of Education, and including Microsoft, is providing a million small devices called the BBC micro:bit, to middle schoolers in the UK in 2016.

This demo describes the device, the programming environment, design considerations, and early evaluation. As the first paper on the BBC micro:bit, it also serves as a landmark and reference for interested academics and teachers to join the movement. Specifically, the demo addresses the software engineering of Microsoft's Touch Develop platform, and how it was adapted to work on a very small device. There is also an accompanying video[1].

2. THE BBC micro:bit
The BBC micro:bit[2] is a pocket-sized, codeable computer, designed to allow children to get creative with technology. The BBC

[1] http://www.aka.ms/bbcmicrobit

[2] https://www.microsoft.co.uk

announced the micro:bit on July 7, 2015, teachers were trained from August 2015, and devices were made available to schools from February 2016.

The micro:bit measures 4cm by 5cm and is available in a range of colours. Its design is intended to make it fun and easy to use. Something simple can be coded in seconds – like lighting up its LEDs to display a pattern – with little prior knowledge of computing. The emphasis is on imagination and creativity.

The micro:bit is powered by an ARM Cortex-M0 Processor[3] and has 256K non-volatile flash (for program and static data) and 16K volatile RAM (for stack, heap). It connects to other devices, sensors, kits and objects, and is intended as a companion rather than a competitor to devices such as Arduino, Galileo, Kano, littleBits and Raspberry Pi, acting as a spring-board to more complex learning. Key features of the micro:bit include:

- **25 red LEDs** to light up, flash messages, create games and invent digital stories
- **Two programmable buttons** to provide input;
- **An on-board motion detector** or 'accelerometer' to detect forces acting on the device.
- **A built-in compass** or 'magnetometer' to sense direction.
- **Bluetooth Smart Technology** to connect other micro:bits and devices, kits, phones, tablets, cameras and so on;
- **I2C/SPI capabilities** to work with any other sensor or display;
- **Five Input and Output (I/O) rings** to connect the micro:bit to devices or sensors using crocodile clips or 4mm banana plug to power devices such as robots and motors.

3. PROGRAMMING THE micro:bit

To bring the BBC micro:bit to life, Microsoft developed an enhanced version of their popular Touch Develop[4] web application and hosting service. All micro:bits share a dedicated website on Microsoft Azure and users can choose from a range of online code editors available from most modern web browsers.

Microsoft supplied two languages/editors – Touch Develop, a semi-structured text-based language, and the Block Editor, a graphical coding language. The Touch Develop web app supports all the code editors built for the micro:bit, runs the micro:bit simulator, and compiles programs to ARM machine code. Advanced users can use C++ to directly program against the run-time system and interface with not-yet-supported hardware sensors.

3.1 Languages, Editors, Compilers

We extended Touch Develop to support a *progression* of languages with accompanying browser-based editors. The Block Editor provides an introduction to structured programming via blocks that can be snapped together (Figure 2). Unlike other similar offerings [5] [9], Touch Develop's Block Editor is strongly-typed and the programs convert seamlessly to a next level of complexity in learning [8].

Touch Develop's classic mode shown in Figure 4 features a statically-typed scripting language with syntax-directed editor. The language subset contains: while and for loops; if-then-else conditional statement; functions; local and global variables; integer, boolean, string and image types; operations over values of the above types; user-defined event handlers and libraries.

Browser-based compilers from the Block Editor to Touch Develop and then to ARM assembly and machine code automate the transition from a visual language to a text-based language, and then to binary language of the ARM-based micro:bit. The first compiler allows a student to convert a Block Editor script into a Touch Develop script with a single press of a button (Figure 2).

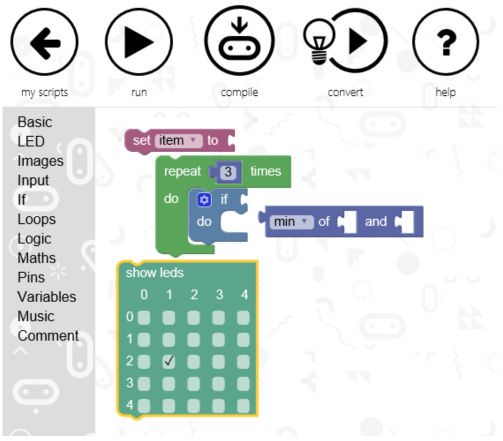


Figure 2. The Block Editor

3.2 Compile and Flash

Figure 3 shows the flow of compilation for the micro:bit. When a student has her Block Editor or Touch Develop script (Step 1) ready, she can connect her micro:bit to a computer via a USB cable, so it appears as a mounted drive. Compilation from Touch Develop to the micro:bit proceeds all within the confines of the web browser. The student is prompted to save the ARM binary program to a file (Step 2) which she then simply drags to the micro:bit mounted drive, which flashes the micro:bit device (Step 3) with the new program.

In more detail, the Touch Develop script is first compiled to ARM Thumb assembly code, which is then translated into machine code. The machine code is then injected into a pre-compiled runtime file (which conveniently uses a text-based hex format). The runtime file comes with metadata specifying the addresses of runtime functions, which helps with linking. The in-browser assembler handles not only the 12 or so instructions that the compiler generates, but also the remaining 100 or so, which allows users to write inline assembly in their Touch Develop scripts if they so desire.

3.3 The Simulator

Before a student compiles her script for the micro:bit, she can run it using the Touch Develop micro:bit simulator, all within the confines of a web browser. The simulator has support for the LED screen, buttons, as well as compass, accelerometer, and digital I/O pins. To run a student's Touch Develop script in the web browser, Touch Develop compiles it into JavaScript, the scripting language built into all web browsers.

The working of the simulator can be seen in the following popular example of Rock-Paper-Scissors (Figure 4). The code in Touch Develop is shown on the left and a picture of the device on the right. The micro:bit accelerometer is simulated with either the accelerometer of the phone or tablet or the mouse pointer of a desktop computer. The simulator image tilts to visualize the simulated forces. The example also illustrates how succinct programs are in Touch Develop. In particular, output to the LED display can be easily created on the simulator, as is shown in Figure 2 and illustrated in the fourth program line. It is not necessary to reference individual LEDs by number on a grid.

[3] http://www.arm.com/products/processors/cortex-m/cortex-m0.php

[4] http://www.touchdevelop.com

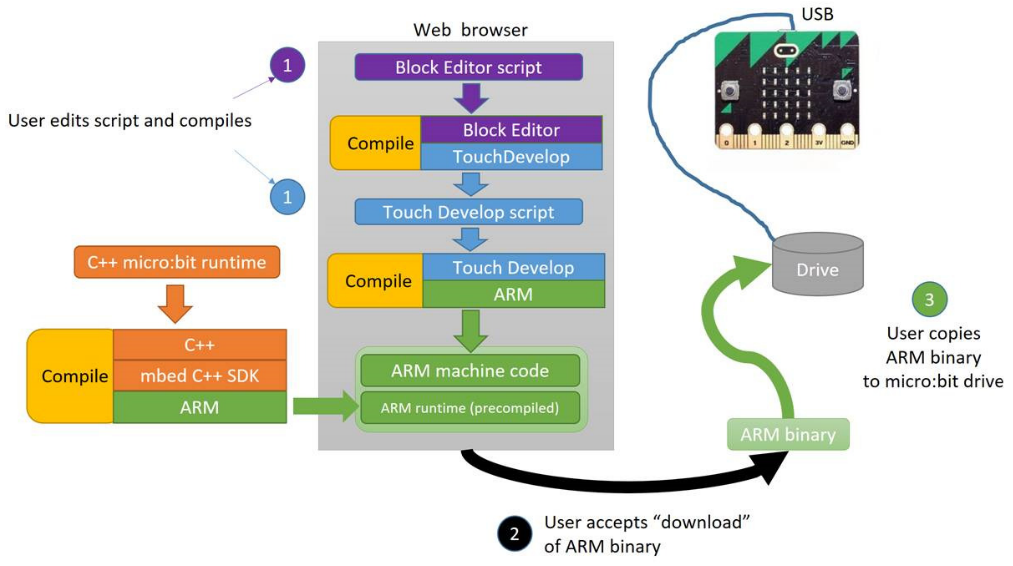


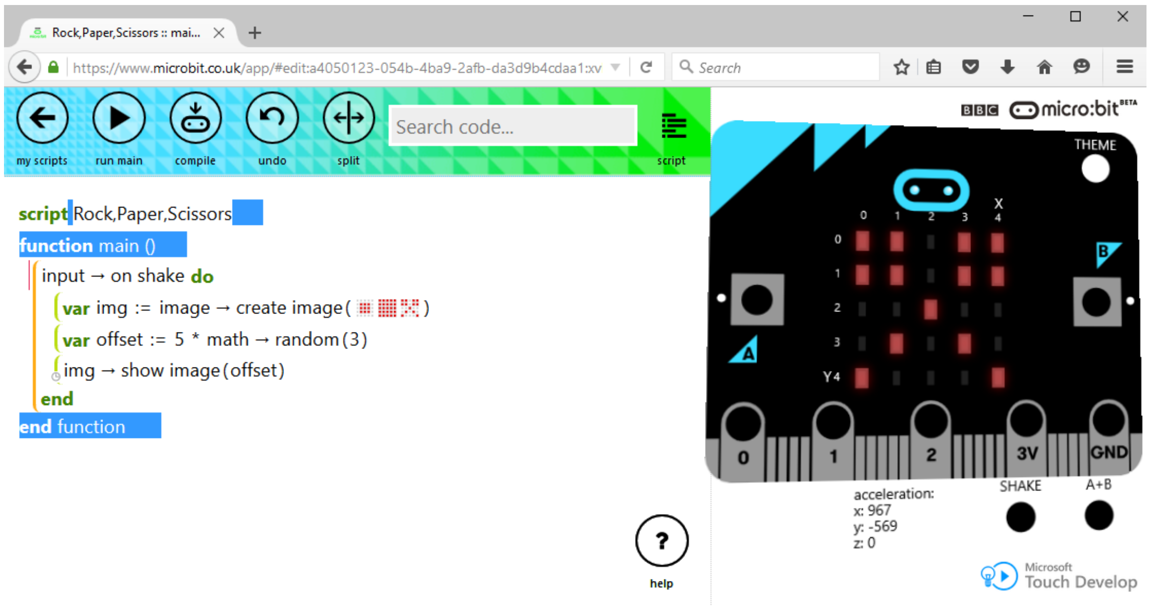
Figure 3 The compilation flow for the micro:bit

3.4 The Libraries

Onboard, the micro:bit is programmed in C++. The **C++** micro:bit library provides access to the hardware functions of the micro:bit, as well as a set of helper functions, such as displaying a number/image/string on the LED screen. The Touch Develop micro:bit library mirrors the functions of the C++ library. When a Touch Develop script is compiled, the calls to Touch Develop micro:bit functions are replaced with calls to the corresponding C++ functions. The C++ run-time system exposes hardware features; and takes care of mundane tasks, such as:

- blinking the LEDs several times per second, in order to ensure that the battery is not drained within a few hours;
- managing an event bus where clients can listen for, and dispatch events;
- "de-bouncing" the buttons, that is, making sure only one "button pressed" event is generated, even though the physical button may bounce off briefly;
- setting up a lightweight thread scheduler;
- abstracting away the specific communication protocols of the on-board accelerometer, thermometer and magnetometer.

TouchDevelop is strongly typed, which is good, because well-typed TouchDevelop programs generate well-typed C++ programs, hence alleviating the need for a dynamic type mechanism. Given the underlying constraints and the small amount of memory, dynamic types would likely put a low limit on the size of programs one can write.



```
script Rock,Paper,Scissors
function main ()
  input → on shake do
    var img := image → create image( )
    var offset := 5 * math → random(3)
    img → show image(offset)
  end
end function
```

```
acceleration:
x: 967
y: -569
z: 0
```

Figure 4 The micro:bit simulator

TouchDevelop, in its traditional JavaScript code generation scheme, is garbage-collected. We did not implement a garbage collector for the micro:bit; rather, we rely on reference-counting. We claim that this is "good enough" for an embedded device; by the time users generate cyclic data structures, they will likely be advanced enough to switch to C++ and handle memory themselves.

Users can define records with fields in TouchDevelop ("objects"); we map these to ref-counted C++ structs. Users can write event handlers that capture local variables; we use C++11's lambda-capture for that purpose.

4. DESIGN CONSIDERATIONS

In dealing with schools and children, many compromises had to be made to ensure ease of use, hardware safety, and privacy in the cloud, authorization and sheer roll out of devices at scale. These topics will be covered in another paper.

Providing young students with an understandable programming model was a challenge, and it took us several iterations to come up with the current semantics. For instance, students can use a busy-loop, active-polling programming model that is easier to understand and reason about, but does not scale well to larger programs. The alternative is an evented, reactive-based programming model that relies on cooperative threading. Naturally, this latter model comes with its own set of problems, such as making sure students' code yields often enough or defining clear semantics for handling button events. We mitigate these difficulties using a variety of mechanisms, such as warnings in the simulator if a loop has been visited a great number of times.

5. EVALUATION

We have been collecting and evaluating data on Touch Develop since 2011 [1] based on over half a million users and over 150,000 scripts. The median size for scripts was found to be 24 lines, but up to 100 lines was common. The target audience of the micro:bit is children who will write short scripts, making heavy use of libraries to accomplish complex tasks. With the more customized libraries (as described in Section 3.4) we estimate that micro:bit users will create satisfying apps in less than 50 lines.

We are entering the test phase of the hardware and transfer technology. Our prime success metrics will include ease of use and robustness, which we can only measure once the micro:bits are in the field. To date, we have trained 600 teachers in the UK prior to the launch and are working through the feedback.

6. RELATED WORK

We can compare the micro:bit to other coding gadgets such as the Arduino and the Raspberry Pi according to processing capacity, output, input and software. Firstly, the micro:bit is self-contained with sufficient onboard sensors and buttons for input, as well as LEDs acting as output. Even though it can be connected to external devices, it works perfectly on its own, lowering the barrier to entry into programming.

The micro:bit features a ARM Cortex-M0 processor which is more powerful than the AVR in the Arduino, meaning that any valid C++ program can run unmodified on the micro:bit using the proper GCC toolchain. However, the micro:bit is less powerful than a Raspberry Pi, which is a complete computer that runs a full-fledged operating system, while the micro:bit runs one program at a time. Typical micro:bit programs link to the *run-time system*, which provides various drivers for the display, buttons and extension ports, as well systems support for memory allocation and cooperative threading.

The software platform for the micro:bit has some similarities to Pocket Code [9] and App Inventor [5]. The advantages of Touch Develop are its progression for lifelong coding, as discussed in Section 3.1, and its platform independence. Considerable thought went into a seamless linking of the software to the hardware, providing a unified experience, which is essential for children and teachers who are new to coding.

7. ACKNOWLEDGMENTS

Nikolai Tillmann was the inspiration behind the Touch Develop system. Steve Hodges has guided the design of the hardware. Joe Finney and his team at the University of Lancaster have provided the runtime system for the micro:bit. Clare Riley has been the indefatigable force behind the project from its start.

8. REFERENCES

[1] Thomas Ball, Sebastian Burckhardt, Jonathan de Halleux, Michał Moskal, Jonathan Protzenko, and Nikolai Tillmann, Beyond Open Source: The TouchDevelop Cloud-based Integrated Development Environment, MOBILESoft, 83 - 93, 2015.

[2] Quinn Burke and Yasmin B. Kafai. 2012. The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. SIGCSE Technical Symposium, 433-438, 2012.

[3] CAS: http://www.computingatschool.org.uk/

[4] Steve Hodges, James Scott, Sue Sentance, Colin Miller, Nicolas Villar, Scarlet Schwiderski-Grosche, Kerry Hammil, and Steven Johnston. 2013. .NET gadgeteer: a new platform for K-12 computer science education. SIGCSE Technical Symposium, 391-396, 2013.

[5] J. Liu, C.-H. Lin, P. Potter, E. P. Hasson, Z. D. Barnett, and M. Singleton, Going mobile with App Inventor for Android: a one-week computing workshop for K-12 teachers, SIGCSE Technical Symposium, 433–438, 2013.

[6] Orni Meerbaum-Salant , Michal Armoni , Mordechai (Moti) Ben-Ari, Learning computer science concepts with Scratch, Computer Science Education , Vol. 23, Iss. 3, pp239-264, 2013

[7] Hadi Partovi. 2015. A comprehensive effort to expand access and diversity in computer science. ACM Inroads 6, (3) 67-72, 2015

[8] J. Protzenko, Pushing Blocks All The Way To C++, In Blocks and Beyond Workshop, Atlanta, Georgia, 2015

[9] W. Slany, A mobile visual programming system for Android smartphones and tablets, VL/HCC, 265–266, 2012

[10] Amber Settle, Baker Franke, Ruth Hansen, Frances Spaltro, Cynthia Jurisson, Colin Rennert-May, and Brian Wildeman. 2012. Infusing computational thinking into the middle- and high-school curriculum. ITiCSE, 22-27, 2012

[11] Linda Werner, Shannon Campe, and Jill Denner. Children learning computer science concepts via Alice game-programming. SIGCSE Technical Symposium, 427-432, 2012.

[12] Teruya Yamanishi*, Kazutomi Sugihara, Kazumasa Ohkuma and Katsuji Uosaki, Programming instruction using a micro robot as a teaching tool, Computer Applications and Engineering Education, 23, (1), 109–116, 2