Multi-Platform Computing for Physical Devices via MakeCode and CODAL

Thomas Ball
Microsoft Research, Redmond
United States
tball@microsoft.com

Judith Bishop
Stellenbosch University
South Africa
judithbishop@outlook.com

Joe Finney
Lancaster University
United Kingdom
j.finney@lancaster.ac.uk

ABSTRACT

As the Internet of Things becomes commonplace, modern software must encompass the sensors, actuators and controllers that make up these physical computers. But can non-experts program such systems? Can such software development be undertaken by anyone, especially programmers who are learning or who are not aiming to be technical experts? We describe the motivation and principles behind Microsoft MakeCode and CODAL, two symbiotic frameworks which have many innovative engineering features for physical computing. Together, these two technologies highlight a new approach to software development for embedded computing devices which provides accessible programming languages and environments that reduce the complexity of programming embedded devices without compromising the flexibility or performance of the resulting software.

KEYWORDS

Physical computing, internet of things, programming languages, runtime layer, micro:bit, Typescript, MakeCode, CODAL

1 INTRODUCTION

Physical computing deals with embedded hardware and software capable of interacting directly with the real, physical world through sensors, actuators and controllers. It has become a prominent domain in recent years due to: (1) the increase of embedded technology in our everyday environments (2) its popularity as a means for educators to promote a combination of science, engineering, computing and creativity.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3183463

Although distinct, these two application domains share many requirements – most notably that their primary users are unlikely to be trained experts in software development. The complexity associated with embedded software development presents a high barrier for entry to potential users. Many sensor and other devices are still programmed in low level languages such as C, and specialist hardware and software are required to physically transfer programs onto them.

In recent years we have started to see a move towards reducing the complexity required to program physical computing devices, with notable examples including the Arduino[1] and Raspberry Pi[2]. Although the spread of electronic boards has made a great contribution to physical computing, neither of these has truly made a breakthrough in usability. The potential user is still exposed to technologies that are difficult to learn – Arduino adopts a C/C++ user programming language, and the Raspberry Pi brings with it the complexities of an embedded Linux operating system.

Other technologies are emerging that attempt to place high level scripting languages "on the metal", such as Micropython [3], Espruino[4], JerryScript[5] and Embedded Lua[6]. These languages bring considerable benefits, but still require a significant degree of computing knowledge, or backup manuals [3]. Moreover, they also have performance overheads and battery lifetime impact associated with virtual machine (VM) based solutions. Often, they do not suit the soft real-time asynchronous programming characteristics of physical computing devices.

In the remainder of this paper we introduce two technologies that aim to lower this barrier to entry, namely, *Microsoft MakeCode* – a web-based environment for coding physical computing devices, and *CODAL* - a lightweight C++ runtime designed to act as a cross-compile target for high level languages such as MakeCode.

2 THE TECHNOLOGIES

Microsoft MakeCode[7] is an open source[8] web-based environment for coding physical computing devices. The system runs entirely within the browser without any software installation. This approach allows MakeCode to work online and offline with any modern web browser on almost any operating system running on desktops,

[1] https://www.arduino.cc/en/Guide/Introduction
[2] https://www.raspberrypi.org/
[3] http://docs.micropython.org/en/latest/wipy/index.html
[4] https://www.espruino.com
[5] http://jerryscript.net/
[6] http://www.eluaproject.net/
[7] https://makecode.com/
[8] https://github.com/microsoft/pxt

laptops, tablets, and even smartphones. MakeCode supports a broad spectrum of programming languages including an extensible visual block-based language, TypeScript[9] (a superset of JavaScript) and C++. Together they enable users to progress simply from one language to another as their knowledge, interest and confidence grows.

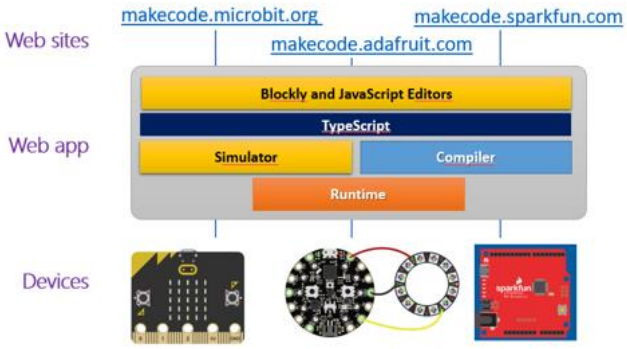

Figure 1: The MakeCode Technology Stack

Unlike traditional VM-based architectures, MakeCode contains an *in-browser compiler* that translates these high-level languages into native code for the target device, as outlined in Figure 1.

TypeScript is used as a common language by MakeCode to enable a seamless translation between its different languages. Furthermore, complementing the web editor and compiler is a fully functional device simulator used to reduce application development time. Physical computing devices supporting MakeCode typically present themselves as a USB mass storage device (flash drive) and/or Bluetooth. Once an application has been written, it can be downloaded onto those devices by saving a file from the web application onto the device.

Underpinning MakeCode is a lightweight component-oriented C++ runtime called CODAL [10, 11, 12] (the **C**omponent **O**riented **D**evice **A**bstraction **L**ayer). Designed for resource constrained physical computing devices, it enables easy programming of devices in C via either synchronous or asynchronous, event-based programming. It consists of a tailored non-preemptive scheduler, asynchronous procedure calls, optimized type safe memory allocation, strongly typed device drivers, an asynchronous event bus and a structured object-oriented API to device hardware. Together, these features provide a substrate that raises the level of abstraction of physical computing devices to make it easier to support high-level languages such as Typescript. The CODAL architecture is shown in Figure 2.

Together, MakeCode and CODAL provide users with Blockly and JavaScript editors to create programs, with the ability to convert back and forth. An in-browser compiler quickly creates an executable file to download to the physical device. A simulation of the physical device allows the user's program to be tested even without the device being present. A self-guided "Getting Started" experience introduces the basic features of the programming environment, and a set of projects for making and coding.

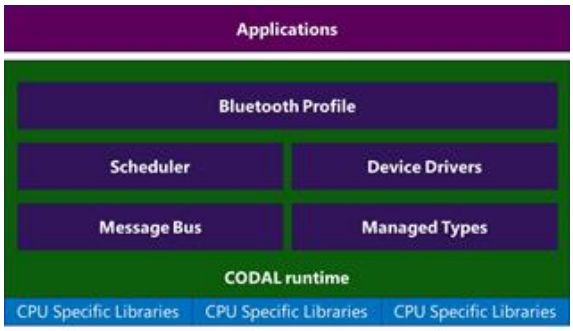

Figure 2: CODAL Architecture

3 IMPACT

MakeCode and CODAL now support a range of popular embedded processors, from 8-bit CPUs such the Atmel AVR CPU used in Arduino platforms to 32-bit ARM Cortex processors. Exemplar devices include the BBC micro:bit, Adafruit Circuit Playground Express and Seeed Studio Grove Zero platforms.

MakeCode is the primary editor for the BBC micro:bit, an education oriented device available worldwide and in 17 languages via the not for profit Microbit Education Foundation [13]. The micro:bit has national deployments in three countries, and over 750,000 devices have been delivered into UK schools alone, providing a coverage of 90% of all UK mainstream high schools. Studies of these deployments show that 90% of students and teachers agree that the micro:bit helped convince them that anyone can code, and 88% agree that the BBC micro:bit helped them to see that coding in a physical environment is not as difficult as they thought it was.

In the maker world, projects abound[14]. We are beginning to collect data on usage which can be shared. Early indications are that the low barrier to entry is a significant attractor for using MakeCode.

REFERENCES

[1] Thomas Ball, Jonathan Protzenko, Judith Bishop, Michal Moskal, Jonathan de Halleux, Michael Braun, Steve Hodges, Clare Riley: Microsoft touch develop and the BBC micro:bit. *ICSE (Companion Volume)* 2016: 637-640

[2] Nikolai Tillmann, Michal Moskal, Jonathan de Halleux, Manuel Fähndrich, Judith Bishop, Arjmand Samuel, Tao Xie: The future of teaching programming is on mobile devices. *ITiCSE 2012*: 156-161

[3] Gordon Williams. 2017. *Making Things Smart: Easy Embedded JavaScript Programming for Making Everyday Objects into Intelligent Machines*. Maker Media, 2017

[9] https://www.typescriptlang.org
[10] https://github.com/lancaster-university/codal
[11] https://lancaster-university.github.io/microbit-docs/
[12] https://lancaster-university.github.io/microbit-docs/

[13] www.microbit.org
[14] https://www.hackster.io/72016/smart-fan-control-system-with-micro-bit-c41a14