Supporting situated STEM learning - TouchDevelop Integration of the UCL Engduino over Bluetooth

Simon Baker
Department of Computer Science
University College London
simon.baker.13@ucl.ac.uk

Stoyan Dekov
Department of Computer Science
University College London
stoyan.dekov.13@ucl.ac.uk

Fadi Fakih
Department of Computer Science
University College London
fadi.fakih.13@ucl.ac.uk

Jan Medvesek
Department of Computer Science
University College London
j.medvesek@ucl.ac.uk

Venus Shum
Department of Computer Science
University College London
v.shum@ucl.ac.uk

Dean Mohamedally
Department of Computer Science
University College London
d.mohamedally@cs.ucl.ac.uk

Abstract

Teaching programming is fast becoming a fundamental learning practice in schools for Science, Technology, Engineering and Maths (STEM) subjects. Through interactive learning with tangible devices such as Lego Mindstorms, students and teachers can explore ways to collect data, analyse and illustrate core principles in STEM subjects. The majority of such educational programmable devices require the use of a PC to program them. This short paper reports on a work-in-progress development project that demonstrates a fully interactive, STEM situated and wireless method to enable students to learn programming. It allows students and teachers to create STEM experiments though only a Windows phone and a programmable device solution. This has the intention of focusing their Constructionist learning activities applied to STEM learning when situated with programming capabilities. It comprises of the programming language TouchDevelop with a Bluetooth based library to operate the UCL Engduino, a customised teaching apparatus based on the Arduino platform. We demonstrate a process for integrating teaching devices with TouchDevelop to expand on pedagogical techniques for both student programmers and educators in programming.

Keywords: STEM, learning by doing, Programming, Bluetooth, TouchDevelop, Engduino, Arduino

1. Introduction

TouchDevelop is one example of programming and application creation for anyone to script their smartphones anywhere – you do not need a separate PC [1, 2]. Separately, the UCL Engduino has been designed as Internet of Things embedded device, based on the Arduino design with a number of sensors, triggers and actuators integrated directly onto the board [3].

The Engduino (Fig. 1) is a cheap (~£30) and small teaching apparatus for school students and teachers, designed and developed at the Computer Science Department at University College London. It is based on the Arduino LilyPad design.

SPLASH '14 Companion, Oct 20-24 2014, Portland, OR, USA
ACM 978-1-4503-3208-8/14/10.

The Engduino is pre-equipped with four sensors: Light, Temperature, Accelerometer and Magnetometer. It also has sixteen RGB LEDs for visual output, a button for input, SD card for storage and an infrared transceiver. This design makes it a suitable teaching platform as-is, for basic STEM programming. External modules, sensors, and other devices, including Bluetooth, WiFi, Speakers or pH Sensor etc., can also be connected, through the Arduino wiring connections. The original idea of the Engduino is to be programmed through a USB by using the Arduino IDE, allowing programs to be written and flashed directly onto the board. Once programmed, it can be disconnected and used with the support of a rechargeable battery. For example, it can be connected by a student to collect pH values in soil samples, or heartbeat pulses of a person. The on-board microSD card reader allows students to store data locally to collect and analyse later.

We extend TouchDevelop to allow the user to program, control and gather IO data from external embedded devices, in particular the UCL Engduino over Bluetooth. The use of the UCL Engduino and the Arduino IDE is currently taught in school sessions with step-by-step tutorials of simple programs. There are a number of concerns with this approach involving the use of the Arduino IDE, the necessary hardware connection (USB) between the device and the PC, the complexity of the tasks and more.

The proposed solution makes use of all the cloud, touch and web-sharing benefits of TouchDevelop adding wireless capabilities when programming and controlling the embedded device, with respect to mobile and wireless classroom teaching. This pedagogical approach intends to bring programming to situated classroom and field based learning in STEM problem-based areas.

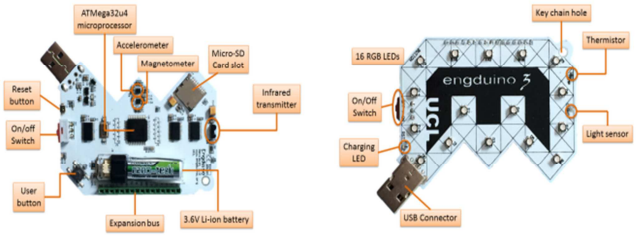

Figure 1. UCL Engduino 3.0 [3]

2. Related Works

Lego Mindstorms: A library has been developed by Christian Jacob that provides direct communication between TouchDevelop and the Lego Mindstorms NXT external device [4]. The library provides a series of commands which are supported by the NXT and are called through TouchDevelop. Each command is encoded into the Lego Mindstorms NXT communication protocol [5] to then be transmitted via Bluetooth, where they are read and executed by the NXT. The communication protocol transmits data in packets of bytes, where each command is represented by a specific byte or array of bytes (field). The protocol consists of fields for command type, operation type and additional information. The packets are processed by the firmware running on the NXT. The communication is bi-directional; the NXT can return packets back to the device running TouchDevelop, containing data recorded from the NXT. To expand the NXT into using Arduino parts for teaching in STEM subjects, a further Breadboard Connector Kit for NXT is required but possible.

TouchDevelop integration with the UCL Engduino (wired version) consists of a library from the Microsoft Research TouchDevelop Team [6,7] which exposes all of the functionality of Engduino 3.0. Programs are created by TouchDevelop and export the generated code to the Arduino IDE requiring further configuration of the Arduino IDE and the Engduino board. This is connected via USB COM port, and re-flashing the code onto the board manually. A second method uses a virtual simulation of the embedded device, which only supports the basic functionalities of LEDs and a button press (mouse click). This approach has the disadvantage of a student having to use the Arduino IDE and limiting the use of TouchDevelop to the PC web app.

3. Problem Formulation

3.1 Examples

A series of open day events for student studying STEM subjects were conducted at UCL, where they were exposed to programming using the UCL Engduino and Arduino IDE. After a brief introductory lecture, they are provided with tutorial materials to help them write their first applications. The students write their applications in C++ in the Arduino IDE, running on a PC. They connect the Engduino to the PC via USB, configure the device in the Arduino IDE, and upload the code to the board to execute their application. They can use the Engduino's built in sensors to record various data, and use the built in LEDs to generate visual feedback. They can view data being recorded by the Engduino, requiring a constant connection to the PC.

3.2 Problems

3.2.1. Limitations with Arduino IDE for Teaching in STEM subjects

The event described above uses the Arduino programming language in order to program the Engduino board by flashing the desired code onto it. The teaching rationale for improving upon this are:

IDE Features lacking for students: There is minimal color-coding of variables / keywords and a learning curve in learning C program structures (for example use of header files). All the source code must be stored in a single file, without an easy way of splitting it into multiple files.

Libraries / naming / need of technical knowledge: Use of low-level C++ libraries, the naming of variables and methods and the extensive need of tutorials can make the Arduino language not well suited for educational purposes. A simple task such as connecting to an LED requires configuration of the pins on the board; whether they are input, output or both. Many simple programs require knowledge of C++, including pointers and other data structures.

Documentation: The Arduino programming language is an implementation of Wiring, using a simplified version of C++ and the Processing IDE. It does not provide programming mechanisms such as IntelliSense, which would allow one to see a list of available functions and variables or use the built-in documentation of the libraries (code comments) directly from the IDE.

Debugging: Students frustration with learning to program is easily seen in the debugging phases of development. Many IDEs and languages have spent intensive resources into making great debuggers with a large selection of features (MS Visual Studio, Eclipse, etc.). The Arduino IDE does not support any type of debugging due to the nature of the programs.

3.2.2 External Devices

Previously TouchDevelop only supported direct communication with the Lego Mindstorms NXT external device. Support for the Engduino amounts to writing the program in TouchDevelop, pasting the translated C++ code into the Arduino IDE and loading it onto the Engduino. This code simply runs in a loop and to change the operation of the Engduino it needs to be reconnected to the Arduino IDE and re-flashed with the new code.

3.2.3 Communication

Laptop/Desktop requirement: Programming for the Engduino currently requires a dedicated computer, which runs the Arduino IDE and uses a traditional keyboard/mouse input. In our approach we describe how the hardware device can be programmed from a smartphone or tablet which runs the TouchDevelop application.

Wired communication: Communicating with the Engduino currently involves a constant wired USB connection to a computer. In our proposed solution the hardware device can be wirelessly communicated with, via a Bluetooth connection with the smartphone or tablet.

Device setup: The initial setup to communicate with the Engduino requires some knowledge of different board types and COM ports; selecting the incorrect board type prevents the program from being successfully written to the board and the COM port must correspond to the USB port in which the device is connected. Using the Arduino IDE would also require hardware drivers installation on multiple computers and different platforms which by itself can be a challenging task. Communication with the device through TouchDevelop simply involves connecting to the device in the tablet's or phone's Bluetooth menu.

Repeated reconnection: In order to change the program running on the board, or retrieve data recorded by the device, it is necessary to reconnect the device to the PC. Following our approach the device's state can be changed and any recorded data can be read without the need for a physical connection.

4. Approach

4.1 TouchDevelop Sensor Library

Our approach was to create a TouchDevelop Sensor library that provides functions to issue commands and receive feedback from the Engduino without any need of the Arduino IDE. Commands include setting LEDs, listening for button input and recording measurements from any of the built in sensors. These functions expose the complete functionality of the Engduino for remote access. They are implemented using our custom Bluetooth Message Format, described in the communication section. The library hides the underlying protocol exchange from the user. The Bluetooth connection is established automatically if the devices are within range of one another and a parser interprets the messages. Commands are converted from user friendly TouchDevelop objects such as "Colours" to their integer representation in the message format. Sensor measurements are encoded from their representation into appropriate TouchDevelop data structures, ready to be processed, e.g. accelerometer values encoded as 3D vectors. This allows us to leverage the visualisation and cloud services of TouchDevelop to analyse, store or combine the measurements with other users.

In order to reduce the delay from message transmission, we try to minimize the number of message packets being sent by grouping commands of the same type into a single packet. For example setting multiple individual LEDs to different colours can be achieved with one message transmission, rather than a separate message for each command. We achieve this by storing the state of the Engduino locally in TouchDevelop and group consecutive commands of the same type before transmission. Without this, delays would become noticeable in programs requiring a fast throughput or large number of messages.

4.2 Preparing the Engduino

We have developed support for direct communication between TouchDevelop and the Engduino. As previously stated, our approach is to have a TouchDevelop library of commands and a program which always runs on the Engduino to accept these commands. The Engduino is flashed only once with Arduino code, and to change its operation, one creates a program in TouchDevelop consisting of library commands which are then sent to the Engduino to execute. TouchDevelop can also receive the results of command execution from the Engduino.

The program parses and extracts the commands. The commands are executed sequentially, if a command requests data to be recorded then a timer is set to record the data at the requested intervals. Finally response packets are generated containing the recorded data corresponding to each command currently under execution; these responses can then be read through TouchDevelop. The program continues in this loop, always ready to accept fresh commands.

4.3 Bluetooth Communication

Bluetooth: Baseband provides a time division duplex link over the Bluetooth radio frequency. L2CAP provides protocol multiplexing, segmentation and reassembly, and quality of service on top of baseband. RFCOMM provides access to a virtual serial data stream over L2CAP. Our custom message format uses Bluetooth to provide a packet structure for transmitting readable integer based commands or data over the serial data stream [8]. Our TouchDevelop library and the embedded software on the Engduino include interpreters for the custom message format.

Custom message format: Communication is facilitated using our custom message format, based on the research work of Jan Medvesek at UCL. The message format provides a packet structure to parse messages at the sender, and allows them to be read and understood by the receiver. Due to the limitations of embedded devices the message format is light-weight and has a limited instruction set, similar to assembly language, however it still supports easy developing and debugging. Working at the lower level especially with higher level languages like TouchDevelop or MATLAB is challenging, therefore we have chosen a reasonable trade-off between implementation, scalability and packet size by using integer representation of the fields. The packet structure is shown in Table 1.

Field	Description
Version	The message format version, e.g. 1
Type	The type of data or command to be executed, e.g. 111 (temperature)
Payload	The data itself or the parameters for the command, e.g. 1000 (record every 1000 milliseconds)

Table 1. Bluetooth packet message format v.1.0 structure.

The RFCOMM protocol provides access to the byte stream. To allow the receiver to synchronise with the beginning of the packet the packet begins with the starting symbol "{", similarly, to allow the receiver to determine the end of the packet, the packet ends with the symbol "}".

The Engduino is flashed with a program which processes received commands, performs the corresponding operations and writes the results into response packets. The L2CAP protocol provides reliability through Cyclic Redundancy Checks (CRC) and retransmissions, meaning implementation of reliable transmission may not be necessary at higher layers.

Packet exchange: Figure 2 shows an example message format exchange, requesting temperature measurements at 1 second intervals. The first integer (1) represents the packet version, the code (111) is the type of request "getTemperature" and the third parameter (1000) is the requested interval in milliseconds.

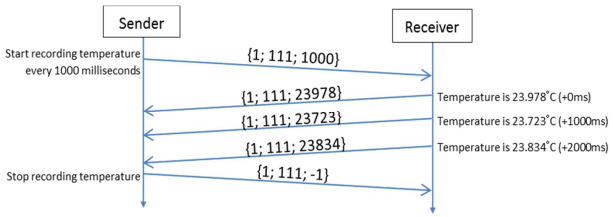

Figure 2. TouchDevelop temperature sensor chart example.

5. Evaluation Method

The evaluation method consists of "Programming 101 with TouchDevelop and Engduino" tutorials, which explain the basics of programming and logic in applied and situated learning roles within STEM based content.

It introduces the external Engduino library in TouchDevelop and consist of basic one to three statement programs, intended to stimulate a scaffolding of constructionist [9,10] approaches, "learning-by-doing" the functionalities of the platform. For example, a one line statement setting the first LED on the board to a given colour (red), showing a function call. This then follows tutorial exercises for setting other LEDs, using different colours, setting multiple lights simultaneously, etc., lights in a set

sequence, demonstrating the role of iteration, and selection to students applying their programming to their STEM tasks.

An example tutorial from the series demonstrates the temperature sensor on the board, introducing students to variables. These variables are shown in the tutorial how to display data graphically, combining the library we have created with the built-in functionalities and benefits of the TouchDevelop platform. (Fig.3)

A series of STEM based on-site [11] classroom scenarios of usability tests with our solution and tutorials is scheduled for term 1 of the new school year (Oct 2014-Jan 2015). Questionnaires and interviews have been prepared on a number of STEM topics for each session as well as in conjunction with direct and indirect observation studies throughout each session. The purpose of these elicitation tests will report data on the students regarding the experience they had with the platform at different levels of difficulty. We shall then report our results comparitively with the previous approach using the Arduino IDE and other educational programming languages such as LMC, Logo, etc. Comparison criteria include: background students scores, completion time, level of satisfaction, perceived difficulty, desire to create more applications, numbers of operations performed and situated location of study (e.g. at a desk, in a group, at an experiment site, field trip etc).

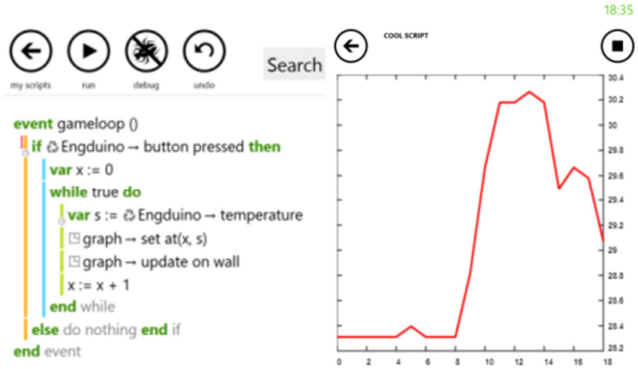

Figure 3. TouchDevelop-Engduino temperature graphing example.

6. Future Work

Message Format v2: A second message format version is currently being designed. Table 2 shows proposed additions towards creating further interoperable and programmable teaching apparatus.

Field	Description
Version	The message format version, e.g. 2
MAC	MAC Address of the sender.
Type	The type of data or command to be executed, e.g. 111 (temperature)
Time Stamp	The relative time of packet construction
Payload	The data itself or the parameters for the command, e.g. 1000 (record every 1000 milliseconds)

Table 2. Bluetooth Message Format v.2.0 structure

A MAC Address field will be added to support multiple communication channels simultaneously. The time stamp field will be used to get measurements of when data has been recorded in time, for the purposes of data analysis and visualisation.

TouchDevelop Cloud Grouping: Future work will include the built-in extension of TouchDevelop allowing the creation of groups and tutorials [12]. A teacher can create a STEM event by assembling a group of users, for example a teacher creating a group of all the students in a class, hackathons or other organised activities. They can further monitor the progress of all the users in the group, share scripts and tutorials to everyone simultaneously making the event more interactive. It also supports comments in the form of messages to all participants which makes it a great opportunity for cloud-enabled app development, including on-line courses, remote team work projects and home learning.

7. Conclusion

We believe the work in progress platform has potential in supporting STEM education with students and teachers. It can be used as multipurpose teaching tool for applying different aspects of programming, from data structures in collecting STEM data, to graphing results with graphics via TouchDevelop features. The ease of learning to program to collect data and visualise results with a phone and a touch-based programming interface, coupled with being mobile for running experiments is a new tool for students and teachers. It focuses less on the PC experience of programming and more on the use of programming as a tool for STEM purposes, which many hope will drive further educational motivation. Future works and case studies intend to demonstrate their expression of STEM based creativity with it.

References

[1] TouchDevelop, https://www.touchdevelop.com/

[2] Tillmann, Nikolai, Michal Moskal, Jonathan de Halleux, Manuel Fahndrich, Judith Bishop, Arjmand Samuel, and Tao Xie. "The future of teaching programming is on mobile devices." In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pp. 156-161. ACM, 2012.

[3] Engduino, http://www.engduino.org/

[4] Christian Jacob. "NXT TouchDevelop Library". https://www.touchdevelop.com/nsdsa (accessed August 2014)

[5] The Lego Group. 2006 "Lego MINDSTORMS NXT Communication Protocol."

[6] Michal Moskal et. al. 2013. *Engduino Tutorial*. Retrieved from https://www.touchdevelop.com/rjtz/info

[7] Tillmann, Nikolai, Michal Moskal, Jonathan de Halleux, and Manuel Fahndrich. "TouchDevelop: programming cloud-connected mobile devices via touchscreen." In *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, pp. 49-60. ACM, 2011.

[8] "Introduction to Bluetooth, 2nd Edition". Lawrence Harte. Pages 23-28. Althos, 2008.

[9] Papert, S. & Harel, I. (1991). Situating Constructionism. Constructionism, Ablex Publishing Corporation: 193-206

[10] Papert (1980). Mindstorms: children, computers, and powerful ideas Basic Books, Inc. New York, NY, USA 1980

[11] Lave, J & Wenger, E. (1991.) Situated Learning. Legitimate peripheral participation, Cambridge: University of Cambridge Press

[12] "TouchDevelop Groups", https://www.touchdevelop.com/docs/groups