René Röpke und Ulrik Schroeder (Hrsg.): 21. Fachtagung Bildungstechnologien (DELFI),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 269

Flowboard: Visual Flow-Based Embedded Programming for Young Learners

Anke Brocker[1] and Simon Voelker[1]

Abstract: Through beginner-friendly environments like the Arduino IDE, embedded programming has become an essential part of STEM education. Learning embedded programming demands coding knowledge as well as basic electronics skills. To explore if a different programming paradigm can help with learning, we developed Flowboard, which uses Flow-Based Programming (FBP) rather than the usual imperative programming paradigm. This way, users code using processing nodes arranged in a graph instead of command sequences. Flowboard consists of a visual flow-based editor on an iPad, an Arduino board in the hardware frame and two breadboards next to the iPad, letting learners connect their visual graphs seamlessly to the electronics. Graph edits are implemented directly, making Flowboard a live coding environment.

Keywords: Embedded Development Environments, Young Learners, Learning Tools

1 Introduction

Embedded development environments like the Arduino IDE enable makers and novices to develop interactive artifacts [PGJ00]. However, learning embedded programming is challenging as it requires an understanding of (a) basic electronics, (b) coding, and (c) the connections between hardware and software [Mc01]. The traditional *imperative programming* paradigm is widespread. Users mostly need to type source code in text, which may lead to syntax errors. *Block-based* environments like Scratch replace textual source code with a graphical editor to assemble code from visual programming blocks. That avoids syntax errors but is still the imperative programming paradigm. We wanted to understand if a different programming paradigm, called *flow-based programming (FBP)*, can help learners even more. In FBP, data flows through a network of nodes that process the data. This paradigm closely resembles electronic signal processing circuits. Unlike in imperative programming, parallel processes in one program are straightforward [WMR02]. FBP development environments such as Microflo or XOD have been available for a few years. These systems are missing two aspects that motivated us to design and build our own hardware and software: 1) *Liveness:* Program graphs can process incoming data and reflect changes directly. These live programs, like analog circuits, can respond immediately to incoming electronic signals, without an Edit-Compile-Run cycle. 2) *Seamlessness:* The points where electrical signals flow into and out of the program graph have a direct correspondence both in the visual graph and as hardware I/O pins.

[1] RWTH Aachen University, brocker@cs.rwth-aachen.de, voelker@cs.rwth-aachen.de

doi: 10.18420/delfi2023-50

2 System Design

The user creates her program graph using a visual, flow-based multitouch editor (cf. https://hci.rwth-aachen.de/flowboard) running an iOS app on a 12.9" iPad Pro. Touch-based interfaces also support more natural interactions that can support learning [Ho04]. Flowboard contains an Arduino Uno board and a custom printed circuit board that also holds the “switchboard": a second microcontroller and 18 electronic switches. The iPad editor talks to the Arduino and the switchboard controller via Bluetooth. The Arduino is running our modified version of the Firmata protocol. Firmata allows the iPad editor to set and read the Arduino pins through serial commands sent via Bluetooth. With a real-time protocol like Firmata Flowboard is a live system as the iPad editor interprets the graph continuously, sending Firmata commands to the Arduino to achieve the appropriate behavior. We provide all files open source. The user has access to all Arduino's I/O pins twice, once on each side of the iPad. Pins are always active and detect plugged-in components automatically. Below the screen, a hardware toggle switch allows disconnecting power from the breadboards to reduce the risk of short circuits while building them. The Flowboard case has three layers, the bottom layer contains the custom circuit board, switchboard controller, and cables. The middle layer holds the breadboards and the iPad. The top layer contains the breadboards, the external pin row connectors, and a power switch. The user drags nodes onto the canvas from the node menu and connects them by drawing virtual wires between them. Both sides of the editor show virtual representations of the input resp.~output pins aligned with the hardware pins on each side of the iPad. Active pins show a green LED and are not greyed out on the screen. The node menu includes nodes for basic mathematical and logical functions as well as nodes to work with more complex electronic components, such as servo motors. We believe that our *Flowboard* prototype using FBP, liveness and seamlessness may increase students' understanding of the interaction between embedded hardware and software code. We hope to facilitate the mental model of students in terms of what programming an electronic component involves and would like to study what students are able to translate of their gained knowledge to non-graphical IDEs. More about the project can be found on https://hci.rwth-aachen.de/flowboard.

Bibliography

[PGJ00] Papavlasopoulou, S.; Giannakos, M.; Jaccheri, L. (2016). Empirical Studies on the Maker Movement, a Promising Approach to Learning: A Literature Review. Entertainment Computing. 18. doi: 10.1016/j.entcom.2016.09.002.

[Mc01] McGrath, W. et al: 2018. WiFröst: Bridging the Information Gap for Debugging of Networked Embedded Systems. In Proc. ACM UIST '18. 447–455.

[WMR02] Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. 2004. Advances in dataflow programming languages. ACM Comput. Surv. 36, 1 (March 2004), 1–34.

[Ho04] Hornecker, E. et al.: 2008. Collaboration and interference: Awareness with mice or touch input. In Proc. CSCW '08. 167–176.