BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2024
http://excel.fit.vutbr.cz

Visual Programming of IoT devices

Lukáš Podvojský*

Abstract

There are a lot of different types of IoT devices that use various communication protocols. The lack of standardization for these devices forces companies to create customized solutions that solve specific problems. Users are then presented with a predefined functionality that can be only slightly altered. One of the solutions for this lack of ability for end users to customize device behavior is to give them more power through the concept that is called visual programming. Users can use visual programming languages and tools to create programs in an easier and more user-friendly way. Visual programming is supposed to be easy to use. However, many programming concepts are still present. Therefore, it is expected that users are willing to learn those basic concepts in exchange for more freedom and power when defining custom logic. This paper will present a new library that implements a visual programming language and a visual editor for creating programs intended for IoT devices.

*xpodvo00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Many users desire more control when using IoT devices. Also, companies whose domain is IoT need a way to allow users to define their own logic. Users of IoT devices can be programmers, hobbyists, domain experts, or just common users. It is difficult to target this broad spectrum with a tool that would accomplish ease of use but also give users enough flexibility to customize the behavior of their devices according to their unique needs or preferences. A proper tool should be intuitive and easy to use but not limit users in terms of their capabilities. The goal is to provide a simple–to–use tool for end users to create and share custom programs for IoT devices and a way for companies to integrate this tool into their existing solutions.

2. State of the Art

Many tools use visual programming concepts in different forms. These tools are called visual programming languages (VPLs), and in most cases, they also include visual editors, where users can create their programs. VPLs can be classified into multiple categories. Kuhail et al. [1] mentions several categories of VPLs: *block–based*, *icon–based*, *form–based*, and *diagram–based*. Remember that VPLs are not strictly classified and different categories can overlap.

The most popular VPL is Blockly[1], developed by Google and served as a base for other VPLs like Scratch[2], which was built on top of it. Figure 1 shows a simple example of a program created using the Blockly visual editor. Another popular VPL, especially in the world of IoT devices, is Node–RED[3]. As you can see in Figure 2, Node–RED is a diagram-based language that provides users with nodes and wires to connect them to create a program flow.

Current VPLs often use a canvas where users drag and drop visual elements to construct a program. This approach can sometimes introduce more complexity to the users because they are more focused on where the elements should go rather than on a core logical problem they are trying to solve. Also, when a program gets more complex, it is difficult to navigate it, and users are forced to scroll or zoom in and out multiple times to get to the part of the program they want to modify. Moreover, support for mobile devices or smaller screens in these tools is fairly limited, considering their popularity among users. Nonetheless, these tools provide a great and unique way for users to create their programs and learn more about programming concepts.

[1] https://developers.google.com/blockly
[2] https://scratch.mit.edu/
[3] https://nodered.org/

3. Proposed Solution

VPL for Things is a typescript library that allows developers to implement visual programming language in their web applications. The library includes a responsive visual editor for creating programs intended for IoT devices. This library is only the "front-end" part of the solution and provides a custom format in the form of JSON files for sharing created programs with a backend solution that can then interpret them accordingly. This allows developers to have broad customization capabilities on the backend part.

Library architecture can be seen in Figure 4. The library consists of multiple graphical components that create a user interface and modules for working with a custom VPL data model. The data model is divided into two main parts: *language* and *program*.

3.1 Language and Program

The language data model defines custom language statements that the user can use and represents the metamodel of the program. Every statement has its own visual and other properties, including syntactic rules, as seen in Figure 5. The program data model defines the structure of a program created by the user.

The language statements are divided into *basic statements* (if, else, while) and *device statements* (Camera.takePicture). Because each IoT device has its functions and attributes, the language is dynamically created based on which devices are used in the program. Both types of statements can be modified or extended by the developer. Language statements can be further classified into compound statements that allow nesting of other statements and statements that take input arguments of various types. Language also includes device and user variables that can be used in statement arguments or when constructing an expression.

3.2 Visual Editor

Visual editor has two views that users can switch between (see Figure 6). A graphical view shows a visual representation of the program and is considered the main working area. A textual view is a raw program output in a JSON format intended for more experienced users to take advantage of for rapid program prototyping. Both views are synchronized, and changes in one are reflected in the other.

Individual program statements are composed into a block where they can be added, removed, or moved into a different position. Statements are grouped by their functionality and differentiated by color so that users associate each color with different functionality. Statement arguments have specified types, so the user is automatically presented with the correct input field type. For example, a *Repeat* statement takes a number of repetitions as an argument, so the user is presented with numeric input. The same thing applies to different types, including a selection from predefined values.

Users are given the option to create their own variables (see Figure 7) and procedures. This helps with better program maintainability and clarity. Users can enter the initial value immediately when creating user variables rather than setting all the variables by a dedicated statement. User procedures are important for abstracting more complicated logic or reusing a block of statements in multiple places in the program (see Figure 9).

4. Conclusions

The created library achieves the goal of providing users with a simple tool to define custom logic for IoT devices and simultaneously allows developers to integrate this tool easily. However, based on user feedback, there is room for improvement in user-friendliness and user help. This tool is more suited for users with at least essential programming knowledge who want more control when creating a program rather than absolute beginners. Further research on visual programming and end-user development is needed to better understand how generic or specific VPLs should be and how much control the end user should have.

Acknowledgements

I would like to thank my supervisor Ing. Jiří Hynek, Ph.D. and Ing. Petr John for their help.

References

[1] Mohammad Amin Kuhail, Shahbano Farooq, Rawad Hammad, and Mohammed Bahja. Characterizing visual programming approaches for end-user developers: A systematic review. *IEEE Access*, 9:14181–14202, 2021.