Visual Programming of IoT Devices

Author: Lukáš Podvojský
Supervisor: Ing. Jiří Hynek, Ph.D.

Brno, 2024

TRY DEMO

Motivation

There is a lot of smart devices that we encounter in our day-to-day lives. From a **smart doorbell**, that lets you see who is in front of your door, even if you are at work to a whole **traffic light system** that controls all the traffic in your city. Their functionality is usually predefined for us, but what if we would like a little bit more control? What if we would like to **define our custom logic** across multiple connected devices in a more **user-friendly way**?

Visual programming (VP) targets **common users and hobbyists**, that have basic understanding in field of programming and empowers them with tools that help bridge the programming knowledge gap while still allowing for the definition of custom logic. VP utilizes **graphical elements** instead of classical programming language textual constructs for easier development.

Existing Tools

Blockly – block-based language, that utilizes **drag-and-drop** method to move and connect **blocks** on canvas. Blockly then generates a final program in different programming languages like Javascript, Python or PHP.

Node-RED – flow-based language where **nodes** represent functions with input/output messages and **wires** that connects them to pass the messages.

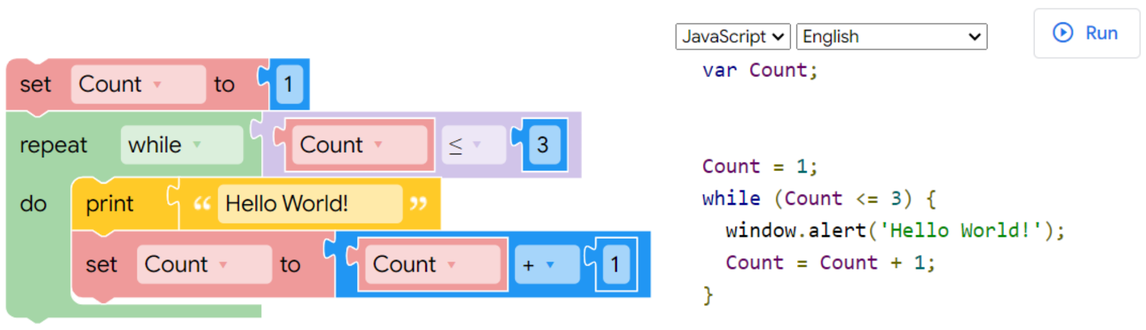

Figure 1: Example of a program created using Blockly, that will print „Hello World“ message three times.

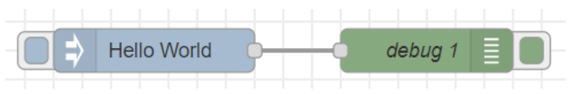

Figure 2: Node-REDs *flow* (set of connected nodes) that prints „Hello World“ in a debug console.

VPL for Things

VPL for Things is a typescript library, that allows you to implement **visual programming language** into your web application. The library includes responsive **visual editor** for creating programs intended for IoT devices. Programs are stored as JSON files in a specific format, that can be shared and interpreted by the backend.

Main features

- ✓ Framework agnostic
- ✓ Easy to integrate
- ✓ Expandable
- ✓ Responsive

```
else: {
    type: 'compound',
    group: 'logic',
    label: 'Else',
    icon: 'diagram2',
    foregroundColor: '#ffffff',
    backgroundColor: '#3b82f6',
    predecessors: ['if', 'elseif'],
}
```

Figure 5: Definition of a standard language statement *else*.

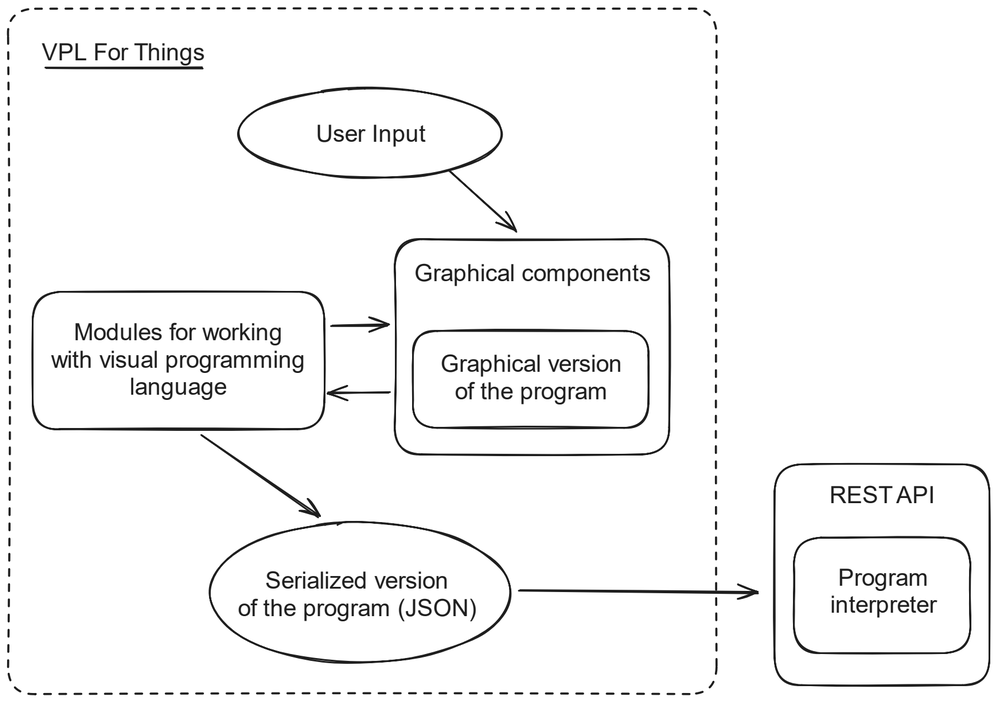

Figure 4: Library architecture.

Visual Editor

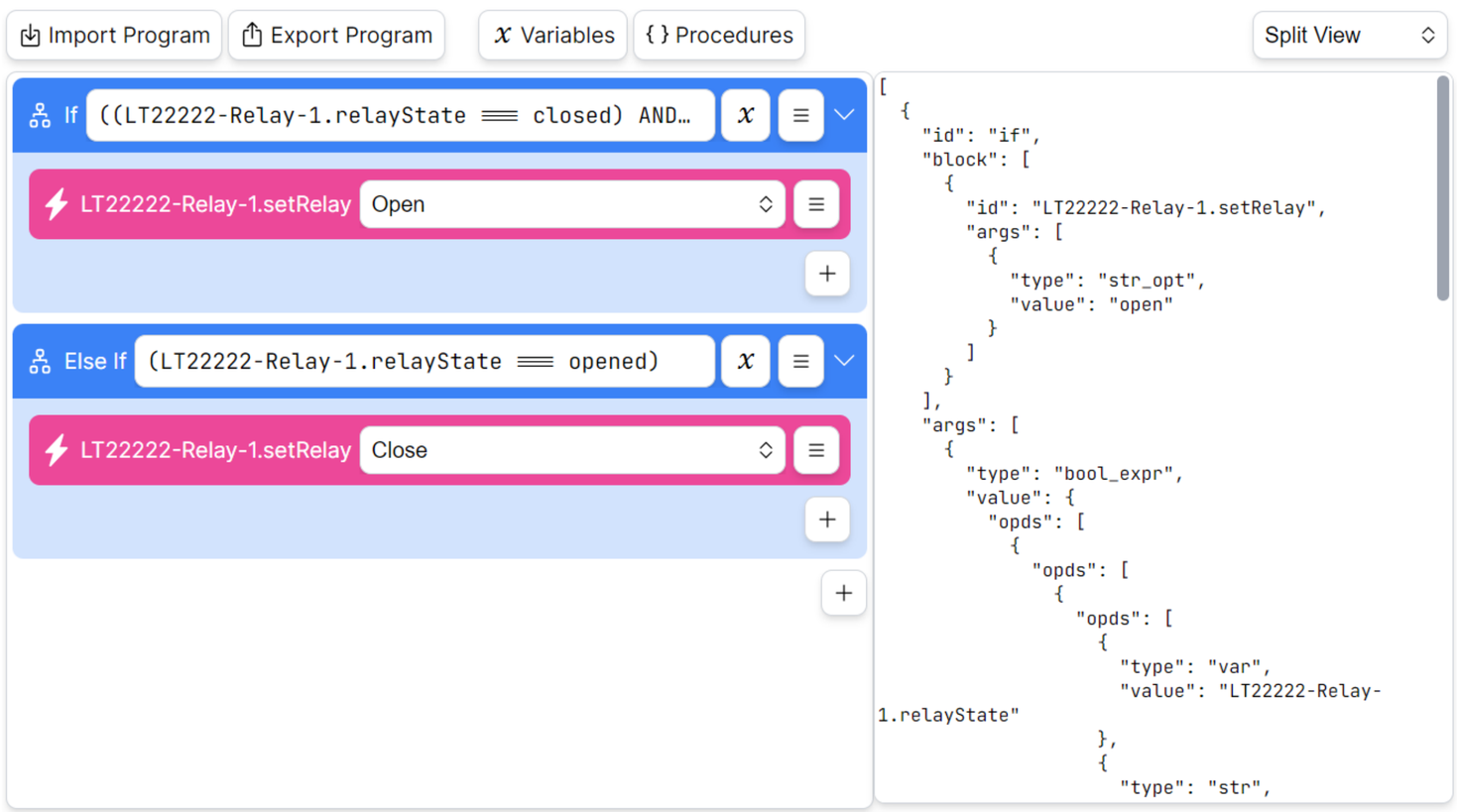

Figure 6: Visual editor with a graphical view on the left side and textual view on the right side.

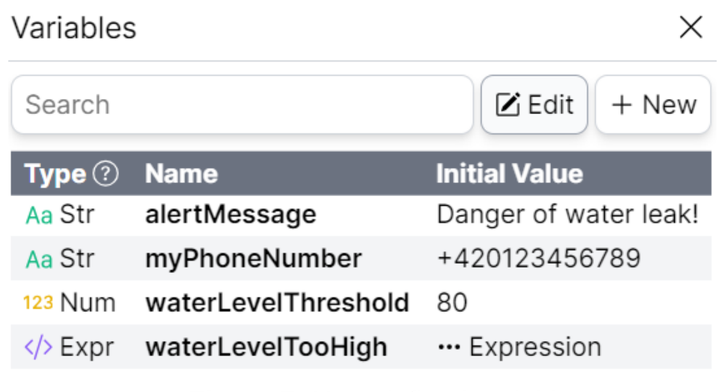

Figure 7: User defined variables.

Figure 8: Expression creation.

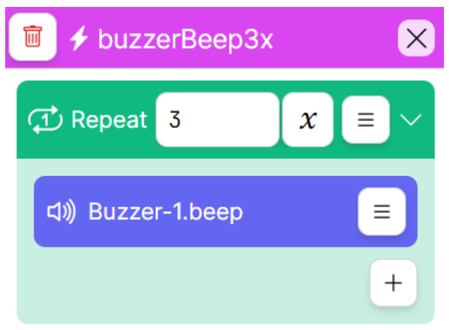

Figure 9: Simple user procedure, that makes a buzzer beep 3 times.