Ardestan: A Visual Programming Language for Arduino

Hiroki Nishino, Ph.D.
Chang Gung University, Taiwan

Abstract

We are currently developing Ardestan, a new Visual Programming Language (VPL) for Arduino. The language is designed for art and design students as novice programmers in mind.

While existing novice programming environments for Arduino may ease prototyping activity for simple tasks, novice programmers often face a difficulty in implementing multitasking and event scheduling. As these features are often essential for interactive artwork/product prototypes, as the lack of them can significantly damage the quality of interactive prototypes.

Fortunately, as multitasking and event scheduling are essential features for interactive music applications, some computer music VPLs are designed so that they can significantly ease the implementation of multitasking and event scheduling and there exist extension software modules to control an Arduino device from the VPL. However, such a system design doesn't allow Arduino system to run standalone without the a personal computer to control its behavior. This is not quite desirable, not because it is not cost effective even for a prototype especially when many devices must be utilized, but also because standalone systems are far more preferable for certain artistic expressions or design ideas (for example, fashion technology or IoT products).

With consideration of such needs, we are currently developing *Ardestan*, a visual programming language for Arduino. As its language design is mostly borrowed from the node-based programming concept of Pure Data, a VPL for interactive music, the language significantly facilitate the implementation of multitasking and event scheduling. *Ardestan* also translates its visual programs into C++ source code for standalone Arduino systems. Such a language can be practically beneficial to support prototyping activity by art and design students as novice programmers in undergraduate interaction design courses.

Related Work

There already exists programming environments that target novice programmers for Arduino. For example, BlocklyDuino [1] is a block-based coding environments that translates a visual program to an equivalent C++ code.

Pduino [6] is an extension modules for Pure Data [4], node-based VPLs for interactive music to control an Arduino device. Yet, while such a programming environment facilitates multitasking and event scheduling, a standalone Arduino system can't be implemented on Pduino.

Visuino [7] and XOD [8] are similar node-based VPLs, yet they can generate C++ source code for Arduino. However, the features for abstraction (subpatching) and event scheduling seems still weak compared to interactive music VPLs.

Description of Our Work

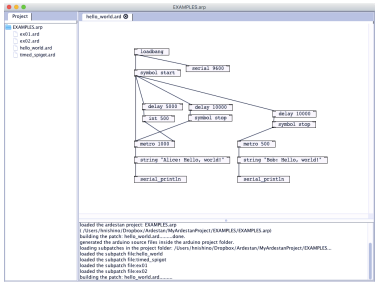


Figure 1: The Ardestan IDE

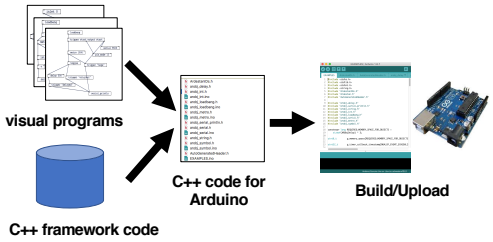
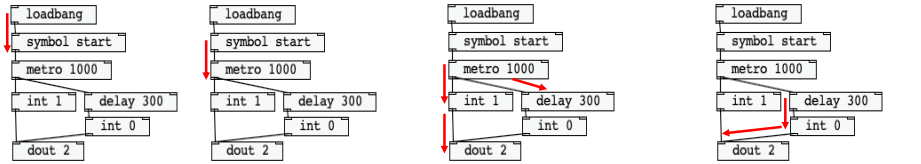


Figure 2: The Build Process of Ardestan visual programs

Figure 1 shows the current version of the Ardestan IDE. It is written in Java to support multiple platforms (Mac/Linux/Win). The code written in this IDE will be first translated into C++ code for Arduino using the Ardestan C++ framework code. The generated C++ code can be compiled and uploaded to the Arduino device (we are currently developing a new version of the Ardestan IDE that integrates the Arduino commandline tools within so that the build/upload process can be performed entirely in the IDE).



When the system start up, The *loadbang* object emits a *bang* message.

When the *symbol* object receives *bang*, it send out a *start* symbol, resulting the *metro* object to starts emitting a *bang* message every 1000 msec.

The *bang* message is received by the *int* object and the *delay* object. The *int* object output an integer value :1, resulting the *dout* object to set the digital out #2 high.

After 300 msec, the *delay* object emits the delayed *bang* message received previously. The message is received by the *int* object, which output an integer value:0. The *dout* object receives it and then set the digital output #2 low.

Figure 3: An Ardestan example to blink one LED (700 msec on/300 msec off)

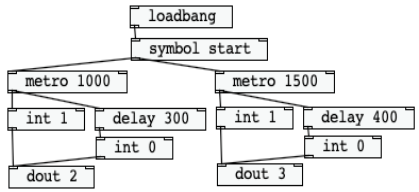


Figure 4: An Ardestan example to blink two LEDs (700 msec on/300 msec off and 1100msec on/400 msec off)

Figure 3 is a simple Ardestan example to blink one LED. As shown, Ardestan is quite similar to node-based VPLs for computer music such as Pure Data and Max. As such a language design can reduce the complexity in the implementation of multitasking and event scheduling, the code to blink two LEDs can be simply created by copying the code to blink one LED and change its parameters as shown in Figure 4.

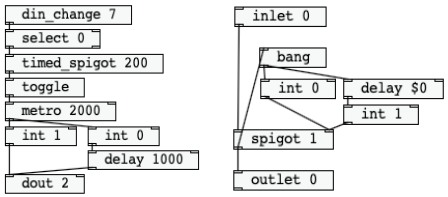


Figure 5: A subpatching example in Ardestan. The patch on the left is using the *timed_spigot* subpatch on the right

Similarly as Pure Data and Max, Ardestan supports the abstraction by subpatching. Figure 5 shows an example of subpatching. The left patch is saved with the filename: 'timed_spigot.ard,' and is utilized the subpatch giving an argument: 200. Inlets and outlets can be given by using *inlet* and *object* objects. By using symbols for parameters such as *$0, $1, $2…,* arguments can be received by the subpatch. Subpatches can be nested.

Discussion

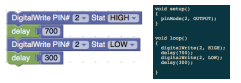


Figure 6: An example to blink one LED In BlocklyDuino and in C++.

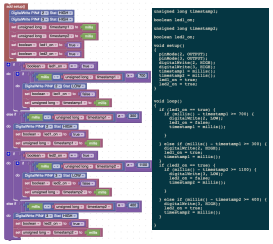


Figure 7: An example to blink two LEDs at different periods in BlocklyDuino and in C++.

As mentioned earlier, while existing novice programming environments for Arduino can ease difficulty in programming for simple tasks, novice programmers can face a difficulty when the implementation must involve multitasking and event scheduling.

Figure 6 is an example to blink one LED in BlocklyDuino, which performs an equivalent task as in the Figure 3 example in Ardestan (the left C++ code is generated by BlocklyDuino). As shown, the task can be simply implemented by using the *delay* function, which let program sleep for the given duration. However, the code can be a lot more complicated for a task to blink two LEDs at different periods. As shown, the code must involve state variables (*led1_on* and *led2_on*) and timestamps (*timestamp1* and *timestapm2*). Such an introduction of new concepts and programming patterns can be a significant obstacle in both program comprehension and implementation for novice programmers .

On the contrary, the Figure 4 example in Ardestan, as the programming language is designed to facilitate the implementation of multitasking and event scheduling, the code doesn't introduce new programming concepts and it simply suffices to copy the code to blink on LED and to change the parameters for the period.

Conclusion/Future Work

As discussed in the previous sections, the implementation of multitasking and event scheduling can be a large obstacle for novice programmers, while the lack of these two features can significantly damage the quality of interactive artwork/product prototypes. The language design of VPLs for interactive music can facilitate this difficulty to a large degree, in such existing VP:s for interactive music, standalone Arduino systems can't be developed, while a standalone system is far more preferable for a certain artistic concept or design idea.

With consideration of such needs, we developed Ardestan, a new VPL for Arduino, borrowing the language design from interactive music VPLs while making it possible to generate the code for a standalone Arduino system. Such a VPL is practically beneficial to support prototyping activity by art and design students in undergraduate interaction design courses.

While it is still in the early stage of the development, Ardestan already provides basic objects such as digital I/O, analog I/O, serial output, arithmetic/relational operators etc., yet more objects are currently developed to further facilitate prototyping activity. We are also planning to integrate more features such as a debugger and GUI objects.

References

[1] Fred Lin. 2015. *GitHub – BlocklyDuino/BlocklyDuino*. Retrieved Mar 26 from https://github.com/BlocklyDuino/BlocklyDuino.
[2] Marius Schebella. 2007. Pduino and other AArduino interface for Pd. In *Proc. of Pd. Convention*.
[3] Miller Puckette. 1997. Pure Data. *Proc. Of Int'l Computer Music Conference*.
[4] Mitov Software. 2017. Visuino – Visual Development for Arduino. Retrieved Mar 26 2019 from https://www.visuino.com
[5] XOD Inc. 2017. *XOD*. Retrieved Mar 26 2019 from https://xod.io