Remotino: Supporting End-User Developers in Prototyping Embedded Devices

Julian Dax
University of Siegen
57072 Siegen, Germany
julian.dax@uni-siegen.de

Thomas Ludwig
University of Siegen
57072 Siegen, Germany
thomas.ludwig@uni-siegen.de

Volkmar Pipek
University of Siegen
57072 Siegen, Germany
volkmar.pipek@uni-siegen.de

AVI, June 07–10, 2016, Bari, Italy

Abstract
In recent years, platforms such as Arduino have made it easier for makers and hobbyists to create "smart objects" and connected devices. However, there are some unique barriers when developing embedded systems that challenge non-professionals. We derive these barriers from existing literature on end-user-development (EUD) in general and in particular EUD for embedded devices. In this paper, we present the first prototype of a tool that supports end users to overcome some of these barriers by addressing problems of missing information and lack of understanding as well as providing visibility and help in decomposition. Our prototype comprises enhanced functionality for the debugging and prototyping process of hardware by showing the current and past state of the embedded device, and allowing end users to remotely control it, when they want to focus on circuit design.

Author Keywords
Debugging; Prototyping; End-User-Development; Appropriation; Internet of Things.

ACM Classification Keywords
D 2.5 Testing and Debugging: Testing tools

Introduction

With the proliferation of Internet of Things (IoT), it has more and more impact on the everyday life of average citizen. This evolution implies that configuration, understanding and programming of IoT technology is getting more and more relevant for a diverse group of people who are not traditional programmers. However, the IoT ecosystem is complex and poses new challenges to individualization of soft- and hardware [3] as well as understanding the hardware in relation to its context [5]. The maker community has developed various hard- and software-systems in recent years which make working with electronics and microcontrollers easier for non-professional programmers than before (e.g. Arduino, https://www.arduino.cc/). While IoT and embedded systems are not synonymous, they are an important piece of the IoT ecosystem.

EUD for Embedded Systems

End-user development (EUD) is defined by the aim of developing "methods, techniques, and tools that allow users of software systems, acting as non-professional software developers, to create, modify or extend a software artifact at some point" [4]. EUD asks how end-users can be provided with support to incorporate and adapt software artifacts to their work practice. With regard to EUD, there is an ongoing discourse focusing on studying appropriation as well as how to design support for these kinds of activities. A common insight is that it is important for end users to learn how they operate their machines [5].

Developing embedded systems comes with its own special challenges. In this paper, we focus on problems with **decomposition** and problems with **visibility**.

Decomposition – one of the core goals of software engineering – is the the idea that a problem should be split up in "mind-sized bites" [6] in order to be solvable. The decomposition problem in embedded systems development is due to the fact that the "powerful separation of computation (software) from physicality (platform and environment), which has been one of the central ideas enabling the science of computing, does not work for embedded systems" [2].

This separation problem is what makes decomposition hard to archive. When debugging embedded systems, programmers oftentimes face the challenge, that they do not know if the bug they are trying to fix is soft- or hardware related [5]. To answer this question, programmers need to decompose the integrated, embedded system into the hard- and the software components in their head. They need to make sure that either the soft- or the hardware works correctly.

The second major reason why developing embedded systems is so challenging is the lack of **visibility** and **observability.** It is a well-known problem in embedded systems development, that the internal state of the hard and -software system is largely invisible to the developer [7]. Booth and Stumpf [1] studied non-professional Arduino programmers and called this problem the "information barrier". Programmers do not know which state the program and the electric circuit is in at a given time and they have only very limited ways to find it out (debuggers are not available in the Arduino platform). Observing, how the state of the embedded system changes over time is especially difficult. Our research question was therefore how to support end users in understanding and debugging embedded systems.

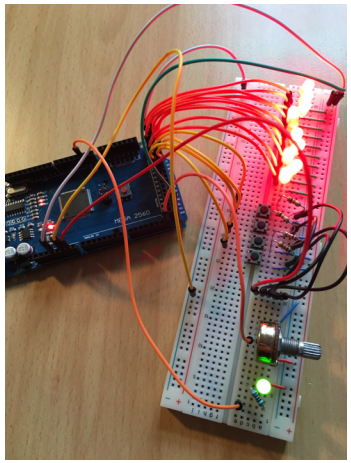

Figure 1: The test setup we used for our Application. It consists of an Arduino Mega, 11 LEDs, a potentiometer and 4 push buttons. The LEDs are used to test digital and analog output. Analog and digital input can be tested with the potentiometer and the push buttons respectively.

Research Approach

To pave the way for answering the research question, our approach tries to address the challenges of archiving decomposition and visibility only in relation to debugging and prototyping of the electrical circuit. For that, we developed a GUI application called *Remotino*. Remotino allows end users to remote control the digital and analog pins on the Arduino form a computer plugged into the Arduino board and to view the inputs on these pins. This way, users can see how the electrical circuit interacts with the Arduino microcontroller by viewing the input and they can test out their circuit by sending output. This helps with decomposition, as the user can focus purely on the electrical circuit first and does not need the ask himself the question, if the problem is hard- or software related. It also helps with visibility and observability in the prototyping phase.

Implementation

To implement Remotino, we chose the Johnny-Five JavaScript library (http://johnny-five.io/) and the Electron application framework (http://electron.atom.io/). The Johnny-Five JavaScript consists of a server, which has to be installed on the Arduino and a client library, which sends commands to the server using the Firmata protocol (https://github.com/firmata/protocol). To provide a modern, cross platform GUI we chose the Electron framework, which is based on the Chrome Browser and allows the creation of native applications using web technologies.

User Interface

In the user interface, we chose to represent each pin of the Arduino (besides the pins for the serial ports) as a separate row (see figure 2). As some Arduinos have more than 50 pins, users can filter the pins and only show pins which are digital only, allow analogue output, allow analog input (analog pins can always also be put in digital mode) or are enabled.

In order to send a value to a pin or to receive values from it, the pin needs to be "enabled" with using a checkbox. The pins can then be switched in different operation modes (digital in, digital out, analogue in, analogue out) depending on their capabilities. Remotino knows which pins support which modes and which microcontroller is attached to it as this information is made available by the Johnny-Five library. For each enabled pin, the current value is shown and the values of the last 10 seconds are plotted.

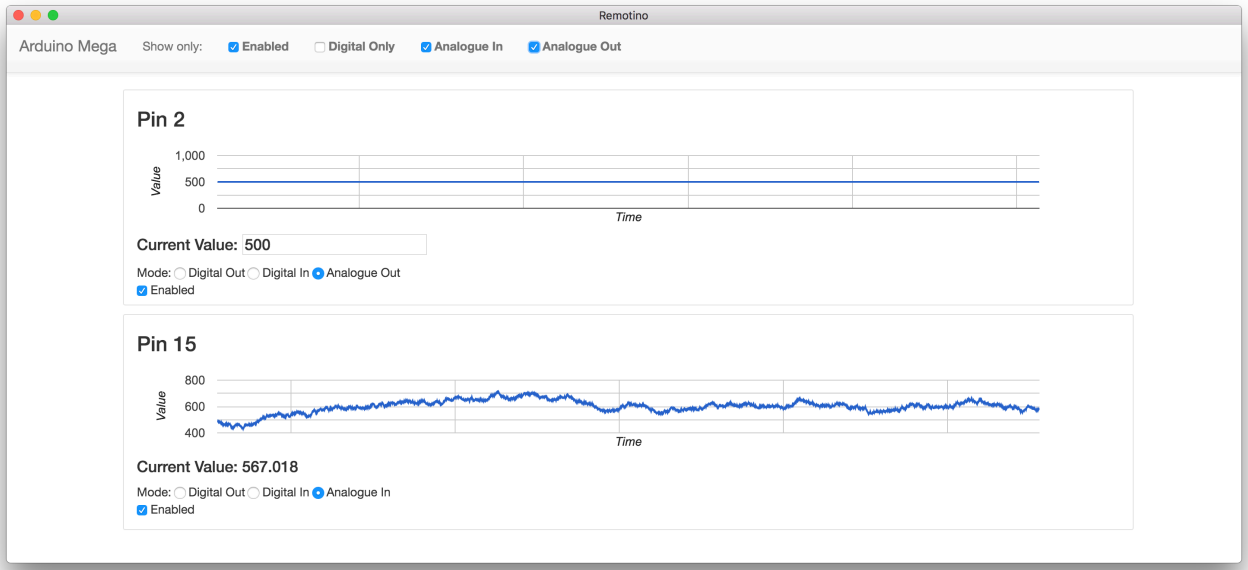

Figure 2: A screenshot of Remotino showing enabled analog in- and output pins

Future Work

In future versions of our tool, we want to include the possibility to use pins that are not simple digital or analog pins. A first step in this is the support for serial ports. We are also working on an integrated event plot, which shows an overview of all enabled pins and visualizes when significant changes on these pins happen. Moreover, macro-recording and playback functionality is planned.

For further requirements analysis and evaluation of the usability and applicability, Remotino will be tested in the setting of a university course on embedded programming for students that have no high programming experience and that mainly focus rather on the design of the hardware artifacts than the programming itself. Based in this long-term evaluation we will be able to examine the appropriation of embedded systems based on Remotino.

References

1. Booth, T. and Stumpf, S. End-user experiences of visual and textual programming environments for Arduino. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), (2013), 25–39.

2. Henzinger, T. and Sifakis, J. The embedded systems design challenge. Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science, (2006).
3. Kubitza, T. and Schmidt, A. Towards a Toolkit for the Rapid Creation of Smart Environments. In P. Díaz, V. Pipek, C. Ardito, C. Jensen, I. Aedo and A. Boden, eds., End-User Development SE - 21. Springer International Publishing, 2015, 230–235.
4. Lieberman, H., Paternò, F., Klann, M., and Wulf, V. End-user development: An emerging paradigm. End User Development SE - 1 9, (2006), 1–8.
5. Ludwig, T., Stickel, O., Boden, A., and Pipek, V. Towards Sociable Technologies: An Empirical Study on Designing Appropriation Infrastructures for 3D Printing. Designing Interactive Systems, (2014), 835–844.
6. Papert, S. Mindstorms: Children, computers and powerful ideas. New Ideas in Psychology 1, (1983), 87.
7. Vermeulen, B. Functional debug techniques for embedded systems. IEEE Design and Test of Computers 25, 3 (2008), 208–215.