Gadgetron Project Proposal

Hannah Chu, Priyanka Kulshreshtha, Michael Gonzalez and Paula Quach

October 26, 2015

1 Research Context and Problem Statement

Our research focuses on automating the process of building gadgets. This process is performed in various steps through a tool-chain. This project is called Gadgetron. Gadgetron's focus at the moment is on the automation of design by allowing users a visual platform to design their desired gadget. It then generates the appropriate files needed to be able to print out the circuit board, a process easily performed by sending the files off to any PCB company. Gadgetron has greatly reduced the limitations, making it possible to design and build a gadget without having to know much about the process especially electrically. The automation of circuit-building both for gadgets and other electronics has been vastly explored in research in regards to rapid prototyping. However, while Gadgetron is part of that number, it will go further than just automating and accelerating the process of building the hardware.

1.1 Problem Statement

The learning curve for building gadgets can be very high. Sometimes a child, or maybe even an adult, gets an idea whether it be a tiny hobby idea or even the first step for a future Nobel Peace Prize. However, in order to make their dreams a reality, they must learn circuits, capacitors, and resistors. Not only that, but the maker must learn to code in order to control their device to do what they envisioned. Of the research that has been done, the focus has usually been on one side (hardware) or the other (software). For instance, the creation of electronics has been improved through research on rapid prototyping like circuits that shrink for flexibility[3] and printed circuits[1]. On the other hand, improving the coding learning curve has been explored in visual programming languages like Snap or UUistle which focused specifically on computer science education[4].

Our tool wants to deal with those needs so it must become more universal and cater to a larger audience. With our project added to Gadgetron, it will become easy to use for any age group as it will require very little prior knowledge. We can introduce the tool, and therefore the process of creating gadgets, at an early age which will also motivate students to pursue it in the future. There have been trials using LEGO MindStorm to teach groups of junior high kids about robotics[2]. With the levels of abstraction that Gadgetron provides, students can start learning about this stuff as early as in elementary school. It can add to the existing curriculum at lower level schools and include a discipline to learn about other than the usual math, science, history, etc.

Gadgetron is already a novel project in that it attempts to automate such a complex process. Combined with the Code Generation Framework, Gadgetron will become one of the easiest ways to create one's very one gadget. It will aid in teaching kids about engineering from an early age as well as fully automate the gadget-building process for anyone who would like to build something for themselves.

2 Proposed Solution

Gadgetron so far has only covered the hardware part of automation. Our plan is to expand on the Code Generation Framework that was started by a graduate student Alexander Caughron. Basically, we aim to extend Gadgetron to also automate the process of writing code for gadgets it builds as well. To do this, we plan to create a visual-based language that will enable users to outline what they want their gadget to do. The program will then generate the necessary code that is compatible with the gadget made using Gadgetron's web tool.

We would like to take this automation one step further and provide a way to automatically generate the appropriate code for Gadgetron users to not only build the gadget but also easily control what they build. That way, they would not have to learn an entire programming language in order to make what they want do what they want: they can just use Gadgetron to automate everything to their desire. We believe that this a necessary step to achieve Gadgetron's overall goal. Adding this piece will reduce the constraints even further and enable people with very little technical knowledge to make a working gadget. All a user would need to do is assemble it and even that may be easily taken care of with some out of scope plans to have Gadgetron also generate steps for the user to follow to attach their components to their gadget. Our focus, however, will be on the visual code-generation aspect of Gadgetron.

Perhaps the most successful model of this type is the LEGO Mindstorm model. It allows users to program their robot by using the accompanying Intelligence Brick. Although the kit claims to give users full freedom on how to design their robot, they have a constraint in the form of the I-Brick [3]. No matter what designs they choose, that block needs to be included. Gadgetron does not have any such constraint and truly gives the user the unique experience of designing a robot according to their preferences.

There has also been a few attempts at creating something similar to our idea. One such system was the visual robot programming for generalizable mobile manipulation tasks project at the University of Washington. Their goal was to make a product, called RoboFlow, that could replace traditional coding when it came to programming robots[2]. The difference is usability. The format of Gadgetron' s Code Generation Framework is dedicated towards providing the easiest way to program robots without having to know much about the process itself. This is different from RoboFlow, in which you do need to have a basic understanding of hardware engineering. Again, Gadgetron would get rid of this limitation, making the tool more universal.

3 Evaluation and Implementation Plan

How will you know if your research was successful? And how will you organize your work to get it done in the time you have?

3.1 Evaluation Plan

Our plan is probably going to stay in the early stages of development. Should we actually complete our solution within the time frame, we have several options open to us in order to test our work. Maybe even further down the line, we could develop our own metrics to evaluate our success, but for now, our methods will have to cover two modes of evaluation:

First, and most simple, is just in terms of correctness in that our solution would actually be viable. As we plan to use our visual language to correctly control gadgets built off of Gadgetron, we can ensure our success by simply testing it ourselves and with other experienced members of Gadgetron. Modular testing of each component separately and then in groups like an average gadget can ensure that the controls work as planned.

Next, and more difficult, are our plans for evaluating whether our solution is actually successful in our goals. We must test how easy the learning curve is for our language and how well it can stand up to our current C/Arduino programming. Thus, we have an idea to gather test groups, perhaps through Amazon's Mechanical Turk where 1 control group will code our gadgets through C/Arduino and another group will use our language to perform the same tasks. We will mostly measure ourselves through the time it takes to complete the task though we might also use surveys to find the hardest issues that come from the programming. We will examine how much faster our visual language helps users acclimate to using Gadgetron and maybe also compare levels of knowledge required to use our work.

3.2 Timeline

Our plan first will start off with getting a firm understanding of similar works: Snap, Scratch, and maybe even more. After that, we will start either by modifying their frameworks or converting Alexander Caughron's for our own purposes. Once we have a solid base, we will then have to integrate that framework into Gadgetron. Once integrated, we will have to start building from our base framework to include all the components available to Gadgetron so the user can actually program all of that. We do not plan on finishing this early, so our evaluation plan will probably have to take place during the Spring. However, throughout our work on adding component controls, we will probably be consistently testing the correctness of our work through programming gadgets.

References

[1] Yoshihiro Kawahara, Steve Hodges, Benjamin S. Cook, Cheng Zhang, and Gregory D. Abowd. Instant inkjet circuits: Lab-based inkjet printing to support rapid prototyping of ubicomp devices. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 363–372, New York, NY, USA, 2013. ACM.

[2] Seung Han Kim and Jae Wook Jeon. Programming lego mindstorms nxt with visual programming. In *Control, Automation and Systems, 2007. ICCAS '07. International Conference on*, pages 2468–2472, Oct 2007.

[3] Joanne Lo and Eric Paulos. Shrinkycircuits: Sketching, shrinking, and formgiving for electronic circuits. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 291–299, New York, NY, USA, 2014. ACM.

[4] Teemu Sirkiä and Juha Sorva. Exploring programming misconceptions: An analysis of student mistakes in visual program simulation exercises. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, Koli Calling '12, pages 19–28, New York, NY, USA, 2012. ACM.