Pin Status

An Arduino debugging library for high school e-textile courses

Michael Schneider
Computer Science
University of Colorado
Boulder CO USA
Michael.J.Schneider@colorado.edu

ABSTRACT

When learning to code a student must learn both to create a program and then how to debug said program. Novices often start with print statements to help trace code execution and isolate logical errors. Eventually, they adopt advance debugger tools such as breakpoints, "stepping" through code execution, and "watching" variables as their values are updated. Unfortunately for students working with Arduino devices, there are no debugger tools built into the Arduino IDE. Instead, a student would have to move onto a professional IDE like Atmel Studio or acquire a hardware debugger. Except, these options have a steep learning curve and are not intended for a student who has just started to learn how to write code. We are developing an Arduino software library, called *Pin Status*, to assist novice programmers debug common logic errors and provides features specific to the e-textile microcontroller, Adafruit Circuit Playground Classic. This work has been funded by NSF STEM+C, award #1742081.

CCS CONCEPTS

• Social and professional topics ~ Computing education

KEYWORDS

Arduino, e-textiles, debugging tool, novice programmer

ACM Reference format:

Michael Schneider. 2020. Pin Status: An Arduino debugging library for high school e-textiles. In *Proceedings of ACM SIGCSE conference (SIGCSE'20). ACM, New York, NY, USA, 2 pages.*

1 Problem and Motivation

The Pin Status library is part of a suite of tools designed for our Debugging By Design project [4,5,8] which focuses on creating debugging tools for high school students who are learning to program with e-textiles . Current tools for debugging embedded systems are intended for experienced programmers and engineers, with advanced IDEs that provide breakpoints and "watch variables" or hardware debuggers which are physical devices that can be connected to a development board or directly to the microchip.

Because an IDE like Atmel or Visual Studio would be intimidating and difficult for a novice programmer to learn, high school students start with the Arduino IDE, which provides a basic text editor, library manager, serial monitor, and can compile and upload code onto the Arduino microcontroller. This basic IDE works well to start with but lacks necessary debugging tools for students as they expand their knowledge. There needs to be some scaffolding that helps beginners to gain basic debugging skills so that they can fix their starter code and then prepares them to move on to more advance tools when they are ready. The *Pin Status* library provides the initial scaffolding with methods that emulate debugger tools, breakpoints and variable "watching", found in more advance IDEs, while keeping the students in the Arduino IDE.

2 Background and Related Work

The lack of beginner friendly debugger tools for Arduino is a known concern within the Maker community [3,4,5,6,7,8]. Tools have been created that provide debugging features (breakpoints, variable "watch") via plug-ins to Visual Studio or Atmel Studio [1,2,6]. These plug-ins allow the student to use the IDE's debugger with the Arduino core library they are familiar with and do not require a hardware debugger (on supported Arduino boards). This pairing of the familiar Arduino libraries with professional IDE debugger tools, makes these plug-ins a great option for more advance undergraduates. But these tools still require a student to learn how to work with a complex IDE like Visual Studio, which can be intimidating for first time programmers.

Recognizing this need for novices to work within a simple IDE, [7] created a customized version of the Arduino IDE and its drivers to provide breakpoints and allow students to step through their code but did not allow for students to view their variables. Also, their system, as of 2019, is not publicly available and may not still be in development.

3 Approach and Uniqueness

The goal of the Pin Status library is to assist students in finding errors in their e-textile projects. More specifically, it has been designed to address common software errors found in beginner code [4]. This section will provide examples of these bugs and detail how the Pin Status library has addressed them.

3.1 Problems with Variables

Kafai, et.al. define two key problems students face when dealing with variables, incorrect variable initialization and variables not matching the circuit [4]. These two errors relate to a common Arduino practice of using variables to keep track of what components are connected to each Pin. For example, a student might connect a potentiometer to Pin 12 and an LED to Pin 3. Their code would look something like:

```
int pot_pin = 12;//Potentiometer
int led_pin = 3;//LED
```

The first line code is an example of incorrect variable initialization. Although a potentiometer might be connected to Pin 12 (See Diagram 1 for image of Circuit Playground), a potentiometer provides an analog signal and requires the code to use the Pin's analog designation (A11). Referencing the correct Pin number for analog, digital, or PWM can be difficult because they share the same physical location on the Circuit Playground! In order to read the potentiometer's value correctly, the code would need to be updated to use A11.

```
int pot_pin = A11;//Potentiometer
```

The second line of code could be an example of variables not matching the circuit. Either because the value 3 was typo or perhaps the student accidentally wired the LED to another Pin. Nevertheless, the value of led_pin no longer points to the correct Pin number. To handle both errors, I designed a selection of methods that can provide detailed Serial output to describe each Pin's power status (On, Off, Receiving Power). Also, the debugging methods will set a corresponding Circuit Playground Neopixel LED (White=On, Off, Green=Receiving Power). Figure 1 details how the Pins are mapped to the on board Neopixel LEDs. With this detailed information, the student will be able to see how power is currently flowing into and out of their microcontroller at any given point in the program and help determine the source of an error (software logic error or hardware failure). The debugging methods created can be divided into two groups, Passive Debugging and Active Debugging.

3.1.1 Passive Debugging

Passive debugging is a way to automatically provide the power status of a pin each time its state changes or is read. A set of C macros replace method calls to the core Arduino library in a student's code with a corresponding *Pin Status* method call. The Arduino core methods are digitalWrite(), digitalRead(), analogWrite(), analogRead(). The *Pin Status* equivalents are digitalWriteD() … analogReadD(). The *Pin Status* methods will perform three actions: validate method parameters and provide appropriate error messages, display the updated Pin state, and update the Pin's Neopixel LED.

3.1.2 Active Debugging

Active debugging requires the student to explicitly determine where to place a debugging method call. For example, if a student needed to check which Pins were receiving power, they could iterate over each Pin reading and then displaying the Pin's state. But with the *Pin Status* library they can reduce this tedious task to a single method call. The method debug() was created to describe each Pin's digital state with Serial output and update each Pin's Neopixel LED. The debug() method was overloaded to instead check a single pin's state and choose between reading the Pin's Digital, Analog (if supported), or PWM (if supported) state.

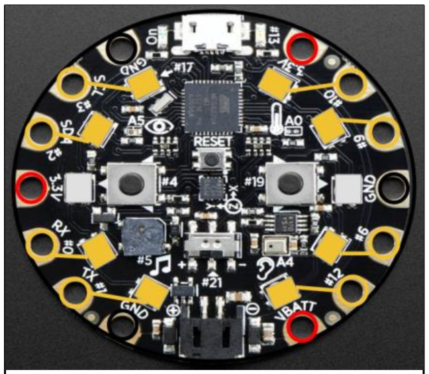
Figure 1: Adafruit Circuit Playground Classic's Pin to Neopixel LED mapping. The Neopixel LEDs map to the nearest Pin, with the center Neopixels being unused.

3.1.3 Breakpoint Proxy

As stated earlier, the Arduino IDE does not provide a way for students to add breakpoints to their code and pause the program's execution. The *Pin Status* library contains a pause() method which will halt the code's execution until the student hits enter in the Serial monitor, i.e. "Press enter to continue".

4 Results and Contributions

The Pin Status library is still under development, but we have begun testing with undergraduate students in an informal demo session. Students appreciated the detailed printed information the debugging methods provided but weren't sure whether the Neopixel LEDs actually helped. During the demo session, students had difficulty with the pause feature. For example, when testing a button, the program would pause before reading the button's state/value. Students would press the button and then un-pause the program. Because the program had been paused, the button press was never read. Instead the students would have needed to hold the button down while they un-paused the program in order to register the button press. This shows a need to further understand the usability constraints for providing a breakpoint scheme for novice programmers dealing with live sensor input.

Overall students found the Pin Status library helpful, but a more formal study is needed to (1) better understand how students utilize the library to determine how effective its current features are, (2) what other features might be included to help students learn debugging techniques, and (3) analyze how students transition, if they do transition, from using the Passive Debugging to the Active Debugging methods.

ACKNOWLEDGMENTS

Craft Tech Debugging Team: Arielle Blum, Chris Hill, and Rona Sadan. Principal Investigators: Ann Eisenberg, and Mark Gross. NSF STEM + C funding, award #1742081.

REFERENCES

[1] Anon. (Oct. 2019). Arduino IDE for Microsoft Visual Studio. [Online]. Available: http://www.visualmicro.com

[2] Anon. (Oct. 2019). Cross-Platform debugger for Microsoft Visual Studio. [Online]. Available: https://visualgdb.com/

[3] Deborah A. Fields, Kristin A. Searle, and Yasmin B. Kafai. 2016. Deconstruction Kits for Learning: Students' Collaborative Debugging of Electronic Textile Designs. In Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education (FabLearn '16). ACM, New York, NY, USA,82-85.DOI:https://doi-org.colorado.idm.oclc.org/10.1145/3003397.3003410

[4] Fields, D. A., Nakajima, T., Amely, J., Fields, D., Landa, J., Amaya, P., & Ottina, J. (2018). Stitching the Loop: A Resource Guide for using Electronic Textiles in Exploring Computer Science. Exploring Computer Science. Available at http://exploringcs.org.

[5] Fields, D. A., Kafai, Y. B., Searle, K. A., and Min, H. S. 2012b. Debuggems to assess student learning in e-textiles. In Proceedings of ACM Special Interest Group on Computer Science Education (SIGCSE).

[6] Dolinay, Jan, Petr Dostálek, and Vladimír Vašek. "Arduino Debugger." IEEE Embedded Systems Letters 8.4 (2016): 85-88.

[7] Torroja, Yago, et al. "A serial port based debugging tool to improve learning with arduino." 2015 Conference on Design of Circuits and Integrated Systems (DCIS). IEEE, 2015.

[8] Yasmin B. Kafai, Eunkyoung Lee, Kristin Searle, Deborah Fields, Eliot Kaplan, and Debora Lui. 2014. A Crafts-Oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles. Trans. Comput. Educ. 14, 1, Article 1 (March 2014), 20 pages. DOI=http://dx.doi.org.colorado.idm.oclc.org/10.1145/2576874