The Toastboard: Ubiquitous Instrumentation and Automated Checking of Breadboarded Circuits

Daniel Drew[†], **Julie L. Newcomb**[‡], **William McGrath**[⋆], **Filip Maksimovic**[†], **David Mellis**[†], **Bjoern Hartmann**[†]
†: UC Berkeley EECS, ‡: University of Washington PLSE, ⋆: Stanford University HCI Group
ddrew73,fil,mellis,bjoern@berkeley.edu, newcombj@cs.washington.edu, wmcgrath@stanford.edu

ABSTRACT
The recent proliferation of easy to use electronic components and toolkits has introduced a large number of novices to designing and building electronic projects. Nevertheless, debugging circuits remains a difficult and time-consuming task. This paper presents a novel debugging tool for electronic design projects, the Toastboard, that aims to reduce debugging time by improving upon the standard paradigm of point-wise circuit measurements. Ubiquitous instrumentation allows for immediate visualization of an entire breadboard's state, meaning users can diagnose problems based on a wealth of data instead of having to form a single hypothesis and plan before taking a measurement. Basic connectivity information is displayed visually on the circuit itself and quantitative data is displayed on the accompanying web interface. Software-based testing functions further lower the expertise threshold for efficient debugging by diagnosing classes of circuit errors automatically. In an informal study, participants found the detailed, pervasive, and context-rich data from our tool helpful and potentially time-saving.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords
Physical computing; debugging interfaces; circuits

INTRODUCTION
In recent years the confluence of easily accessible microcontroller platforms (e.g., Arduino [19]) and component vendors that focus on making electronic components usable by hobbyists (e.g., SparkFun [25], Adafruit [1]) have allowed countless people to get started with programming and electronic design. At the same time, educational programs using these systems have grown in both size and scope. However, projects that require prototyping hand-built circuits are often confusing, frustrating, and time-consuming to debug. Errors can stem from a variety of sources all requiring different tools and techniques to diagnose. Users who have not yet built up debugging experience and intuition often find it difficult to efficiently locate and correct hardware bugs [7, 13, 18, 27].

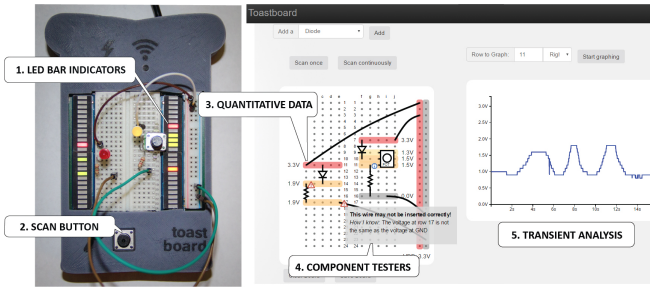

Figure 1. The Toastboard device and accompanying software. (1) LED bars indicate power, ground, or other voltage. (2) A push button triggers a scan. (3) Quantitative voltage data is displayed in the accompanying software. (4) Components are associated with testers that run on every scan. (5) The voltage at a selected row can be viewed over time as a graph.

The solderless plugboard, or breadboard, became available in the 1970s [24]. This passive device for making electrical connections is so prevalent that it is almost synonymous with the act of circuit prototyping. Despite their popularity, breadboarded circuits are prone to a host of common issues such as incorrect component placement, faulty connections, and power management problems. The visual complexity of breadboarded circuits makes it even harder to probe and debug circuits.

We hypothesize that many difficulties in circuit debugging are due to the fact that *much of the state of such systems is both hidden by default and tedious and difficult to examine.* Traditionally, multimeters, oscilloscopes and logic analyzers are used to measure and debug electronic circuits, but they can only probe the circuit at a single specified point (or a small number of points in the case of logic analyzers). Users first have to come up with a hypothesis as to why their circuit is not working, and then choose a specific measurement to test their idea. This process may have to be repeated many times, as inexperienced users often do not know how to form likely hypotheses [11].A recent study of end-users building physical computing projects suggests that new tools should provide in-context advice during the circuit construction process [4].

UIST 2016, October 16-19, 2016, Tokyo, Japan
ACM 978-1-4503-4189-9/16/10.
http://dx.doi.org/10.1145/2984511.2984566

Motivated by this call, as well as Bret Victor's concept of "Seeing Spaces" [29] (which support understanding complex systems through visualization of system state and behavior), we explore how showing information about circuit state proactively can reduce the barrier to evaluate hypotheses and in turn aid electronics debugging.

The Toastboard (Figure 1) is a hardware and software system that makes three core contributions: (1) a ubiquitously instrumented breadboard that enables low-effort, pervasive data collection, (2) a continuously updated, web-based visualization of the state of the breadboard that provides easy access to critical information; and (3) an automated component testing framework that can diagnose common errors and provide useful information automatically. The Toastboard captures and analyzes an entire circuit's voltage characteristics at a specific point in time, or continuously at approximately 100Hz. Voltage signals varying with a higher frequency (around 1KHz) can also be tracked and graphed in some scenarios where an oscilloscope would typically be used. Results are displayed both quantitatively in software and qualitatively with LEDs on the breadboard denoting where electrical connections have been made, as shown in Figure 1. Simple wiring or component placement mistakes can be diagnosed visually.

Further, if a user provides basic information about components on the board to the GUI, Toastboard can calculate some of the circuit's electrical characteristics such as current flow. Component test functions use this information to alert the user to potential circuit errors and guide them towards a solution. For example, if a wire has been drawn in software between two rows but they are measured to not have equal voltages, the user will be alerted to a possible bad connection. Unlike circuit simulators (e.g., SPICE [22], CircuitLab [8]), our IDE uses the actual electric signals from the hardware to test for proper implementation; our hardware design prevents these measurements from affecting the behavior of most circuits that do not involve precise analog timing.

In an informal user study with seven participants the availability of immediate visual feedback was unanimously noted as a useful feature for debugging. Further, the majority of participants commented that the component test functions allowed them to rectify circuit errors without having to chase any false hypotheses. The device did not appear to interfere with circuit building tasks. Multiple participants noted that using it was like "using any other breadboard." Anecdotally, our system compared favorably to common tools such as digital multimeters. Participants greatly preferred the ubiquitous instrumentation to taking painstaking point measurements, which can be physically challenging for complex circuits.

FORMATIVE WORK: IDENTIFYING DEBUGGING NEEDS

To gain an understanding of common circuit debugging needs, we interviewed five practitioners: a novice circuit designer, two electrical engineering laboratory teaching assistants, and two "maker-space" instructors (one with EE background, one with Product Design training). We also supplemented interview data with our own observations teaching physical computing to hundreds of students. The independent interviews were loosely structured and participants were asked to talk freely about about their personal debugging strategies, issues they see students commonly encounter, and methods to reduce debugging time for novices.

The strategies employed by the advanced practitioners appeared to rely heavily on intuition gained through experience. Two suggested the importance of following a set pattern until such intuition is developed – e.g., first check for power, then check connectivity, then check if microcontroller code is actually running.

We identified the following categories of common errors: **Power Management:** power or ground connections not made, improper voltages or currents supplied, power supply not on. **Improperly Placed or Connected Components:** shorted pins, improper component polarity. **Bad Connections:** wires improperly seated in breadboard; bad wires; disconnections while moving or measuring circuit. **Defective Components:** blown capacitors or component pins. Locating errors is additionally complicated by **Visual Complexity:** a "rat's nest" of wires and long jumpers that is difficult to interpret. Designers of circuits that included microcontrollers also frequently encountered additional errors at the intersection of circuits and code: **Incorrect Pin Usage:** setting digital inputs and outputs; internal pullup/pulldown resistors; invalid pin assignments, and **Unexpected Sensor Readings:** temperature variation; supply voltage fluctuation; series resistance of connections.

We distilled the following essential functionality for our debugging tool from both these broad categories and our review of related work:

- Sense the presence of power/ground connections
- Sense when components have been connected
- Measure many circuit nodes at once (to save time and provide contextual information)
- Visualize relevant information about circuit behavior
- Lower the debugging knowledge barrier with automatic analysis

RELATED WORK

The most closely related prior work is on tools for circuit prototyping and learning about electronic circuits. We also survey relevant user interface techniques developed for end-user debugging of software systems.

Physical Circuit Prototyping

Booth et al. [4] studied the performance of 20 amateur makers in building a small project which involved wiring together components on a breadboard and writing an Arduino program. Participants encountered many challenges: 14 were not successful in finishing the project. Participants made many mistakes while writing code, but the majority of task failures involved circuit-related problems, such as using incorrect resistors, miswiring pins, and so on. Participants had difficulty localizing the faults in their projects; for example, they frequently blamed circuit-related problems on their code. The study notes the lack of debuggers for hardware, a role that

Toastboard and other physical circuit prototyping tools seek to fill.

The Visible Breadboard [23] shows voltages at individual holes with LEDs. Instead of standard breadboard connectivity, users "draw" electrical connections using touch input. Voltage information is also displayed in software, however, the software does not provide specific debugging support. Its limited number of holes (36) and its non-standard pitch (3.2 cm), make it difficult to work with standard components or create circuits of even moderate complexity.

The commercially available Digilent Electronics Explorer breadboard [10] offers an oscilloscope, pattern generators, and logic analyzer connected to a GUI for signal analysis. However, it is not ubiquitously instrumented and users must choose test points and manually wire them to the board's test sockets. Its GUI does not offer automatic analyses like Toastboard's component testers.

VISIR implements a relay-based approach that allows students to build a virtual circuit in software, then automatically rewire a remote physical circuit board to match this specification and perform measurements [26]. It aims to increase access to and utilization of lab equipment, but does not focus on improving the debugging process of a single co-located student.

AnyBoard [28] simplifies the prototyping of large discrete logic circuits by modelling them with an FPGA. In contrast, Toastboard focuses on teaching and assisting novice users in assembling and debugging simple analog, digital, or microcontroller-based projects.

Virtual Circuit Constructions and Simulation

Several tools facilitate drawing and simulating electrical circuits. Fritzing [14] allows users to lay out their circuit on a virtual breadboard. Users can toggle between this representation and a schematic. Several other visual tools, e.g. Circuitlab [8], Autodesk Circuits [3], also include circuit simulation.

In contrast to this work, Toastboard provides a view of a user's actual operating circuit by continuously combining actual measurements with user-supplied component information. This approach allows novices to gain intuition about practical circuit construction, and provides time-saving information to users of all skill levels without disrupting their usual workflow.

Education Tools with Physical Interfaces

Several projects have sought to improve electronics education by integrating physical hardware components with context-aware digital information [9, 6, 2]. These toolkits are typically focused on "augmented workspaces" which track custom-made components and provide relevant information in situ. One downside of these toolkits is that they abstract the electronic components into higher level building blocks, limiting the possible combinations and interactions [21, 20]. For example, the Flow of Electrons system [9] displays simulated behavior alongside actual components like LEDs, which are not themselves electrically active. Our alternative approach is to augment the prototyping substrate itself without altering the circuit components, letting novices learn by building real and functional circuits.

General Debugging Techniques

There is a rich body of work in HCI dedicated to supporting end-user software construction [16, 5] and debugging [15]. Our key takeaways from this literature are that a good debugging tool should give its users greater insight into the otherwise hidden behavior of their code and should educate and guide users to solutions, rather than solving problems automatically. We believe that our component testers are a natural extension of these ideas into the hardware debugging realm; they let users know what the problem is, why it happened, and give suggestions on how to fix it.

Programming interfaces that display live information, or "always-on" visualization [17], have shown debugging time can be decreased by proactively displaying information about code that is normally hidden. Our system offers a hardware equivalent of this philosophy, where continuous voltage checks can proactively identify error conditions in the physical circuit.

The HelpMeOut system [12] explored presenting crowdsourced software debugging solutions inside the IDE. Our checker functions similarly automatically present suggested fixes to common problems. Crowdsourced collection of checkers and associated explanations could be integrated into the Toastboard IDE in the future.

Ultimately, the clear connection between software and hardware debugging tools is that when users are equipped with more and richer information, they can both form better hypotheses founded on solid assumptions and can more quickly evaluate them.

A QUICK BITE OF THE TOASTBOARD

We introduce a running example used to show different aspects of the Toastboard's debugging capabilities, shown in Figure 2.

Alice is a novice electrical engineering student taking a class on interactive device design. She is designing a circuit that takes a measurement using an ultrasonic sensor when a button is pressed. For visual feedback, an LED should light up when the measurement is being taken. She plans to use an Arduino microcontroller to record these measurements and send a notification if the reading is below a certain threshold.

She has used her hand-drawn schematic (Figure 2.1) to construct the circuit on the Toastboard, but something must be wrong; the LED is already lit before the button is pressed (Figure 2.2).

In order to debug her circuit, she places components on Toastboard's virtual breadboard in locations matching her physical circuit. The Toastboard scans her circuit automatically and she looks to her software client, where the button she placed now displays a notification (Figure 2.3). Hovering over the button in the client shows details about the issue it has identified (the button's pins appear to be shorted), including suggestions on how to fix it (if the button isn't pressed, try rotating it by 180 degrees). During the scan, Toastboard compared voltages on her physical circuit to expected voltages of the virtual components. The software has an internal model of how the push button should function; when the voltages across it are the same, a checker indicates that there might be a problem.

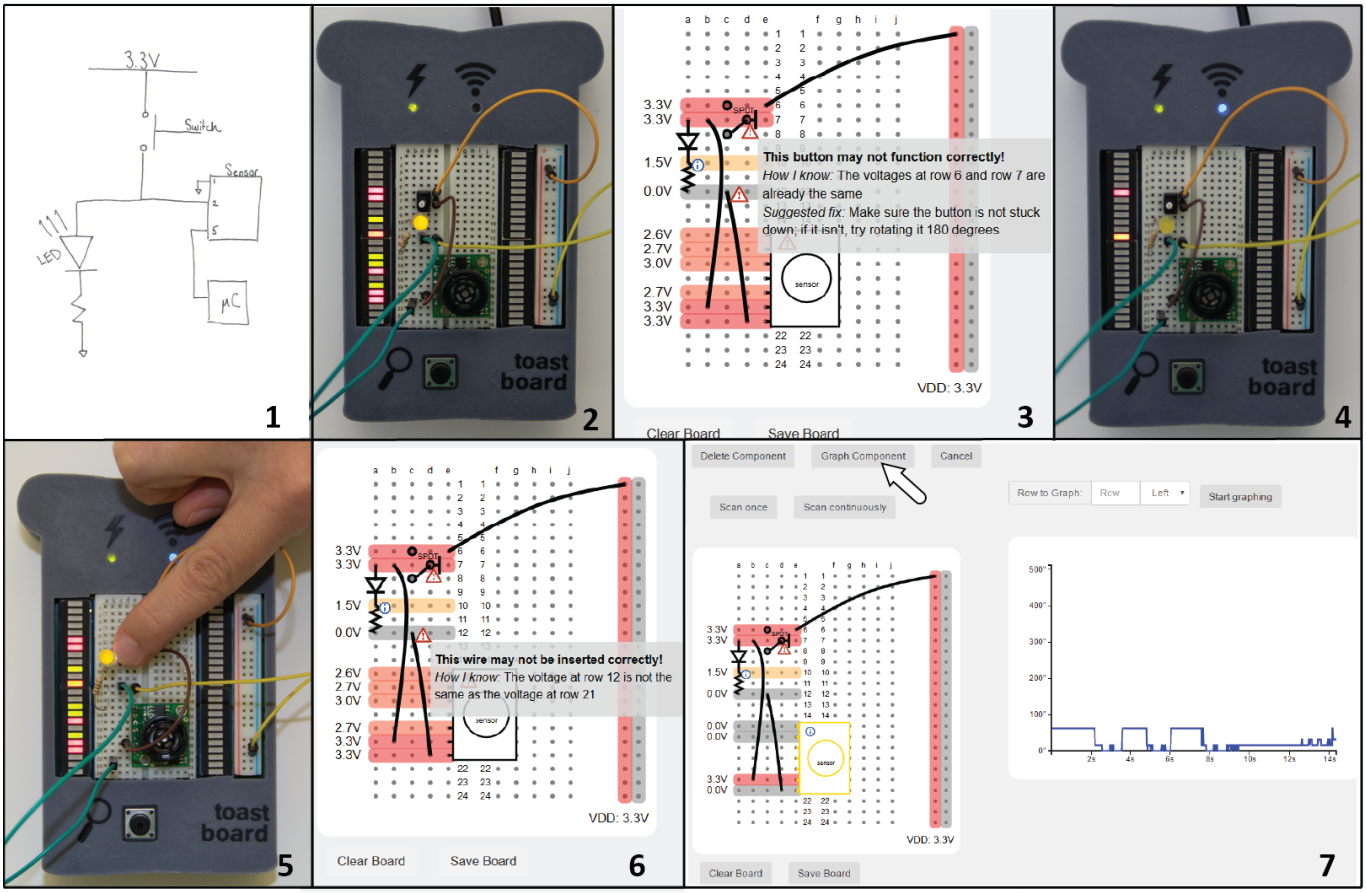

Figure 2. 1) Alice's hand drawn schematic for her circuit. 2) The LED is shining despite the button not being pressed. 3) The client suggests a fix . 4) The LED is now off, as intended. 5) After pushing the button Alice can see the supplied voltage propagate through the system; however, the sensor does not function. 6) An error on the client reveals that a wire isn't connected to the internal ground pin of the sensor. 7) By highlighting the sensor in the software and pressing "Graph Component," the oscilloscope is automatically configured to read from the correct pin of the sensor. The graph value is also automatically converted to inches using the datasheet's scale factor.

After reversing the button, the LED works as intended (Figure 2.4, 2.5). When she presses the button to activate the sensor, however, it doesn't give her a sensible output signal. Alice activates the continuous scanning mode and can now watch the electrical signal propagate when the button is pressed. With the button depressed, an indicator remains over the wire connecting to the ground pin of her sensor (Figure 2.6). Checking the physical wire, she finds that it is indeed loose. When she pushes it in correctly, she gets immediate visual feedback from the LED bars alongside the breadboard.

Having corrected this last error Alice now needs to find the sensor reading range in order to set her Arduino's threshold value. By highlighting the sensor with her cursor, the oscilloscope channel in the software client is automatically set to record from the correct row based on its internal model of the component (Figure 2.7). It also sets the y-axis to the correct units based on the datasheet scale factor included in the component model.

IMPLEMENTATION

The primary design goals for the Toastboard are to provide instrumentation without influencing normal circuit operation and multiple levels of useful feedback to the user.

We connect every row of a standard breadboard to an Analog to Digital Converter (ADC) in order to measure the voltages at each row. In order to sample this large number of inputs we use a chained set of multiplexers controlled by a CC3200 Launchpad microcontroller. This information is visualized using LED bars: the LEDs light up red for power, orange for ground, and green for all other voltages. The software provides additional information to the user based on a combination of user-placed virtual components and this voltage data. A simplified system block diagram is shown in Figure 3.

We took a standard breadboard, removed the backing, and adhered it to a custom PCB using Z-axis electrical transfer tape in order to obtain a unique electrical connection to every row. Because of the large number of rows being sampled (48), we use a two-stage cascade of multiplexers to selectively connect the rows to an ADC on our on-board microcontroller. The microcontroller samples the rows round-robin, either on de-

mand or continuously at approximately 100Hz. Alternatively, a single row can be sampled at approximately 1000Hz.

Multiplexer Cascades

In order to sample the 48 rows of the breadboard we constructed three two-stage cascades of 4:1 multiplexers (shown in Figure 4). This allows us to digitally control which row on the breadboard is measured while providing a high impedance when inactive. As shown in Figure 5, each multiplexer has an associated capacitance. By creating three identical multiplexer cascades to take advantage of the three available ADC channels of the CC3200, we were able to pipeline the multiplexer select signals and reduce both the sampling delay contribution and charge feedthrough effect of these capacitors. Using simple pass-transistor multiplexers (TI SN74HC4066) reduces the delay overhead from decoding control signals at the cost of using more GPIO pins on the CC3200.

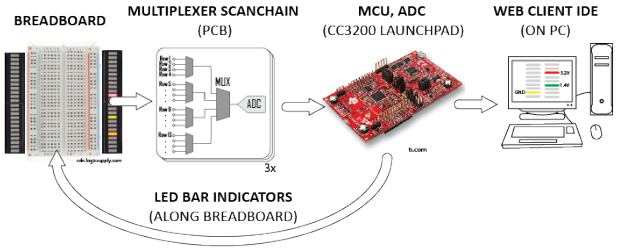

Figure 3. Simplified block diagram of the Toastboard. Direct electrical connections are made from the back of the breadboard to the PCB. These signals are routed to a multiplexer cascade for sequential sampling. There are a total of three multiplexer cascades on the PCB, one for each ADC channel on the CC3200, each servicing 16 breadboard rows. After performing some processing and analysis in the MCU, data is sent to the web client and the LED indicator lights are programmed.

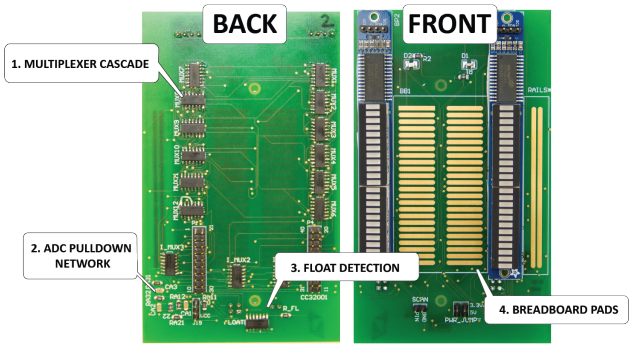

Figure 4. The Toastboard PCB. The entire PCB acts as a shield for the CC3200 Launchpad and includes the multiplexers, LED bars, power rails, and a mounting location with row contacts for the electrical transfer tape and breadboard.

Design Tradeoffs for the Instrumentation Circuitry

The Toastboard's instrumentation electronics are designed specifically so that any circuit prototyped on the Toastboard would operate with the same (or nearly the same) performance as if the user had built it on an ordinary breadboard. This was an important technical challenge: measuring a voltage without the measurement electronics drawing or providing excess current or charge to the user's circuit.

In essence, any resistance that we add to the breadboard rows should be very high (to limit current flow), and any capacitance added should be as small as possible (to limit timing alterations to sensitive circuits).

Figure 5 shows a simplified schematic of the full instrumentation circuit. The analog-to-digital converter (ADC) built into the CC3200 is rated to an upper voltage limit of 1.4V, while the power rail on the breadboard is tied to a regulated 3.3V supply from the CC3200. This necessitates the use of a voltage divider network between the multiplexer cascade and the ADC. This voltage divider network should be made to have as high an impedance as possible to minimize loading of the breadboard rows. The ADC also has its own effective sample-and-hold capacitance of 12pF. This means that when the sample is taken, charge will redistribute from the capacitor C_{samp} to the capacitor C_{ADC}. In order to prevent this charge sharing process from altering the measured voltage, it is necessary to make C_{samp} much greater than C_{ADC}. The charge buildup on C_{ADC} during a sample period will be governed by the time constant $\tau = RC$. By increasing the resistance values (R_1 and R_2) in order to minimize loading and by increasing the capacitance (C_{samp}) in order to minimize charge sharing, we effectively increase the latency for an accurate sample and therefore increase the total scan time. Concretely, we chose resistance values of $15k\Omega$ and $50k\Omega$ with a $1nF$ capacitor.

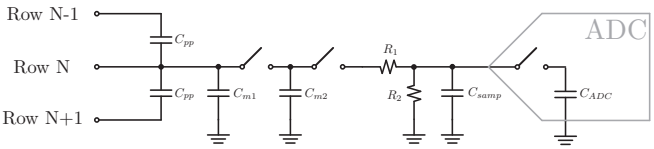

Figure 5. Simplified circuit-level schematic of the Toastboard scan chain for a single breadboard row. The two multiplexers are shown as open switches.

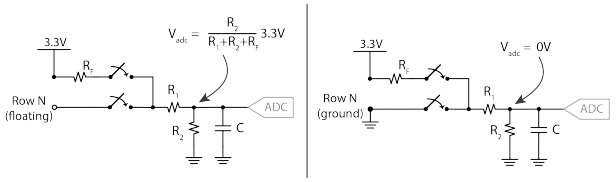

Figure 6. Simplified circuit to demonstrate the float detection procedure. Left: When the sampled row is floating, connecting the float detection output pin forms a new resistive divider yielding a known voltage at the ADC input node. Right: When the sampled row is grounded, connecting the float detection output pin does not change the voltage at the ADC input node.

Floating Node Detection

Breadboard rows that are not part of the circuit (floating rows) should be shown as inactive so that we can notify the user if a row they think they have connected to ground is actually disconnected and prevent the display of distracting and potentially confusing data.

In essence, this requires reliable detection of rows that are not being driven "strongly" (with appreciable current) by a voltage source. Because there is no direct control on these floating nodes, their voltage will vary in the presence of external influences including: 60Hz periodic oscillation, thermal

Component	Error Condition
Resistor	The two ends of the resistor have the same voltage, or at least one row is floating.
Diode	The voltage drop across the two pins is greater than 2.0V, or at least one row is floating.
Wire	The two ends have different voltage, or if at least one of the rows if floating.
Button	The two halves of the button are shorted.
LMC6482 Op Amp	The chip's positive supply is floating or negative supply is not grounded.
Ultrasonic Rangefinder	Either power or ground are not supplied.

Table 1. Supported components and their testers. A message is displayed when the error condition is fulfilled.

noise, capacitive coupling to nearby breadboard rows, and the voltage measurement itself.

To detect whether a row is floating or driven, we take advantage of the fact that the pulldown network will cause floating nodes to be read the same as a ground connection during a normal scan of the board. We first detect this subset of rows that could be either grounded or floating, and exclude all other rows from subsequent analysis. Prefiltering the rows we check in this manner prevents interference with user circuits.

To disambiguate floating from grounded rows, a second scan connects a digital output pin of the CC3200 to this subset of rows through a dedicated float detection multiplexer and reads the resulting voltage. If the row is floating, it will be driven to a known value determined by the resistive divider of the new path and the ADC pulldown network; if it is grounded, the reading will still be at ground (see Figure 6). Floating rows are flagged to prevent them from displaying either LED notifications or web client information.

Components and Testers

In order to assist users when debugging circuits we propose and have implemented an extensible approach for component tester modules. These modules are designed to be placed on the Toastboard GUI as a user builds their circuit on the physical breadboard or be optionally pre-loaded in an educational setting. The virtual components automatically perform tests on the circuit with each scan. Each component takes its type (wire, resistor, etc), row connections, and the voltage values measured by the board to evaluate a set of tests, intended to check for common wiring and power mistakes that might occur while the circuit is being built, as well as to teach and guide novices as they work. We currently support the components listed in Table 1, but the framework is easily extensible by defining new components and tests as Javascript functions. As an example, the code defining the Button tester is shown below ("this" is replaced with "t" and markup is stripped here for brevity).

```
Button.prototype.test = function() {
  if (t.breadboard.getVoltage(t.startRow,t.startPinNum)
     == t.breadboard.getVoltage(t.endRow,t.endPinNum)){

    return "This button may not function correctly!
      How I know: The voltages at its pins are already
     the same. Suggested fix: Make sure the button is
     not stuck down; if it isn't, try rotating it 90
     degrees"; }}
```

All tests on the configured components are run every time the Toastboard takes new measurements; failing tests appear in the client after every scan, or in real time in continuous scan mode. Once the user places the button component on the virtual breadboard, it has access to the location of its two tracked pins in the diagram via its attributes `startRow`,`startPinNum`,`endRow`, and `endPinNum`, and can access the voltage at those locations as measured by the board with `getVoltage()`. At each scan, it checks to see if those two measured voltages are equal. If so, this might indicate that the button is stuck in a closed position, or that the button has been unintentionally rotated 90 degrees (since four-pin buttons look the same in any orientation, it's easy to place them with two pairs of leads shorted out). When an error message is returned, a small warning indicator is placed on the component in the GUI with error text that can be viewed when the user mouses over the indicator. If the tester returns `undefined`, no error is shown.

DEBUGGING WITH THE TOASTBOARD

In this section we show several examples of common debugging scenarios and how Toastboard information can help users distinguish between different possible problems in their circuits.

Simple LED Circuit

Even a simple circuit such as the LED in Figure 7 can have a variety of problems. Two common issues are shown in the figure: On the left (7.2, 7.3), the ground connection is made to an incorrect row. On the right (7.4, 7.5), the LED has been inserted with the incorrect polarity. The user can correctly identify the respective issues with the Toastboard in two different ways: on the physical device via the LED bar patterns (which work without any additional information provided by the user) and in the software client using the integrated component checkers (which require the user to model the circuit by placing matching virtual components in software).

More Complex LED Circuit

As previously noted, prototyped circuits can quickly become too visually complex for efficient debugging. Even the relatively simple circuit in Figure 8.1 has become a challenge to read. Circuits with dynamic elements like buttons present an additional logistical hurdle to measurements. Figure 8 shows the advantages of having a virtual circuit representation populated with real hardware signals; the entire circuit is annotated without having to painstakingly probe with a multimeter, and the full voltage characteristics during a button press can be easily captured. In 8.3, a wire test function indicates that a connection has not been made. Once it is fixed, the current through the resistor can be read (8.4).

Circuits with IC Packages

In this example, we look back at the sensor used in the case study. This time we would like to increase its output signal using an operational amplifier (op-amp); the circuit schematic is shown in Figure 9. The "Load Schematic" button in the software client (Figure 9) allows for population of the virtual breadboard based on a user selected file. If a web tutorial or

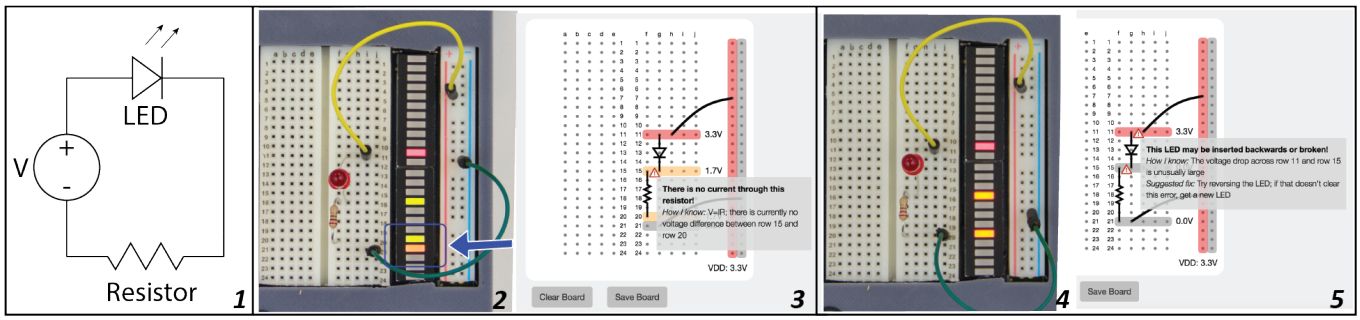

Figure 7. 1) A schematic for a simple LED circuit. 2) Indicator lights show that the leg of the resistor and the ground wire are in different rows. 3) "There is no current through this resistor! How I know: V = IR; there is currently no voltage difference between row 15 and row 20." 4) Indicator lights show that contact has been made to all the correct rows but there is no intermediate voltages present (only ground and power). 5) "This LED may be inserted backwards or broken! How I know: The voltage drop across row 11 and row 15 is unusually large. Suggested fix: Try reversing the LED. If that doesn't clear the error, try a new LED."

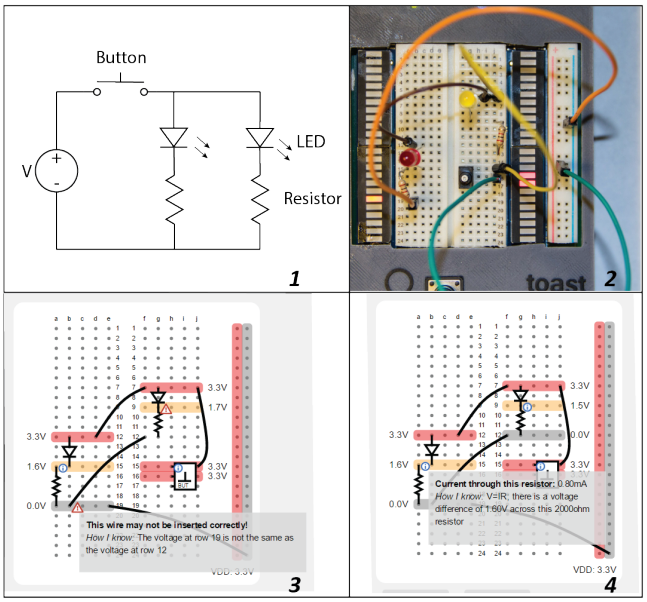

Figure 8. 1) A complex LED circuit. The current through each LED is set by its own resistor. The LEDs will not turn on until the button is pressed. 2) The physical circuit is complex enough to be difficult to read. 3) Test functions can point out problem areas for complex circuits: "This wire may not be inserted correctly! How I know: The voltage at row 19 is not the same as the voltage at row 12." 4) The info icon displays information about the resistor: "Current through this resistor: 0.80mA. How I know: V = IR; there is a voltage difference of 1.60V across this 2000ohm resistor."

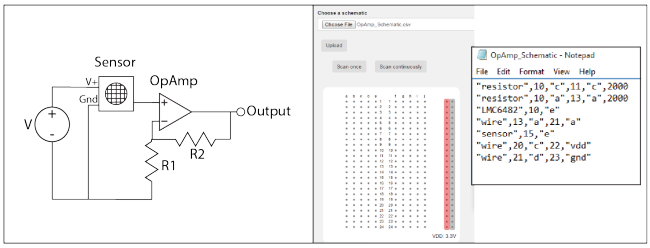

Figure 9. Left: Schematic for an ultrasonic sensor boosted with an amplifier. Right: A user loads this schematic into the Toastboard UI.

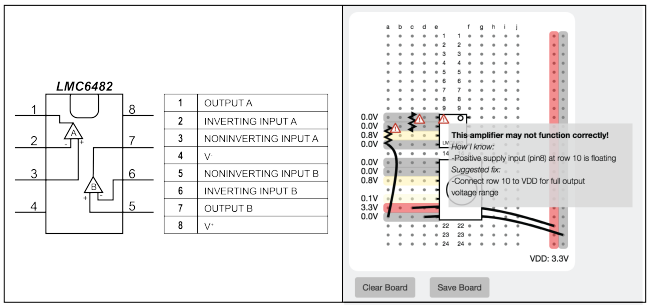

Figure 10. Left: Datasheet information such as component pinouts can be used to generate checker functions unique to a specific part. Right: "This amplifier may not function correctly! How I know: Positive supply input (pin 8) at row 10 is floating. Suggested fix: Connect row 10 to VDD for full output voltage range."

lab assignment has an included schematic, the automatic test functions and immediate feedback of the Toastboard can help users reconstruct the physical circuit.

We show a case where test functions can diagnose an issue related to the internal pin functions of the IC/breakout board packages. In Figure 10.2, the function notifies the user that the positive supply to the amplifier is floating; it tells the user both the breadboard row this corresponds to as well as the internal pin number of the part. The test function output also attempts to teach the user what this input is used for.

INFORMAL USER FEEDBACK

To test the usability of the Toastboard we recruited seven participants (four female, three male). Experience levels ranged from novices with little formal circuit education (four participants), to moderately experienced builders (one participant), to experts who have instructed circuit lab class sections (two participants).

After a brief introduction to the system, participants were asked to complete the following tasks:

- Given a schematic, a description of intended functionality, a pre-populated software client, and a pre-built, incorrect circuit on the Toastboard, identify and fix the problem.

- Given a schematic, a description of intended functionality, a clear software client, and a pre-built, incorrect circuit on the Toastboard, identify and fix the problem.
- Given a schematic, build the described circuit on the Toastboard. If it has errors, identify and fix them. If it does not have errors, hand the board to the experimenter who will make one or more changes to create errors that must be identified and fixed.

We asked each participant to complete the task using the think-aloud method and observed their progress. At the end of each session we interviewed the participant about their impressions of the hardware and software; what features they found helpful or frustrating; how it compared to standard tools such as a multimeter; what type of users would find the device helpful; and what other features they might like to see.

Findings

All participants reported that it was easy to get started using the Toastboard, as it was no different than using an ordinary breadboard. All of the participants noted that the immediate visual feedback provided by the LEDs was useful, with several commenting that having this kind of visibility of loose connections or miswirings would save a good deal of time debugging. Several participants also appreciated access to voltage information of their entire circuit without having to use a multimeter, since the process of probing a delicate circuit can cause additional problems: “you almost have to take apart your own circuit to get a wire in there and probe it."

During the circuit construction task, all participants used the software client's voltage information for debugging; however, not all participants placed virtual components as they built the circuit. In the experienced users' cases this seemed to be a conscious choice, as they only needed the voltage information for this relatively simple task. One participant noted that the system for placing components seemed cumbersome. One of the less experienced participants became stuck when wiring a physical button in the circuit until prompted by the study administrator to try placing the virtual button for help.

The majority of users found the component test functions helpful; one liked the test for a backwards diode since that happens “all the time"; another more inexperienced user hadn't realized that diodes could be placed backwards before seeing a failing test. One case showed the test functions could be potentially confusing: a participant designed their LED circuit to be open only when a button was pressed, which triggered the tester error stating the button could be shorted out. As to whether the testers would help novices learn, one expert participant worried that users would become over-reliant on the testers: “you don't have to think about it, it thinks about it for you." Another expert disagreed, saying that the system would be “like training wheels" — perfectly suited for an introductory course.

Multiple participants expressed a desire for more available breadboard rows and a second available power rail; our approach can easily accomplish this in future iterations. The physical separation between the breadboard rows and the LEDs also caused confusion for some participants. More experienced study participants noted the potential pitfalls of the checkers when attempting to implement components in non-standard ways (e.g., an LED as a Zener diode), while negative feedback from the novice users was centered on hardware limitations.

All participants thought the Toastboard would be useful for education, such as in high school or introductory college electrical engineering classes. Two participants suggested it would be especially useful for hobbyists if included as a package with tutorials.

LIMITATIONS

In this section, we explore and discuss the limits of the current Toastboard system. For the hardware, we focus on board space, automatic component detection, power supply options, and minimizing interference caused by our instrumentation. Meanwhile for the software, we discuss the limits of the flexibility of our current checker implementation.

Toastboard Hardware

Although it would be convenient for the end user if the board were able to detect the type and location of placed components, it is not possible to identify components based solely on their passive electrical connectivity. To our knowledge, any such solution would require active electrical probing of the board, which could damage or disrupt a user-created circuit. We believe that our board editor introduces a small overhead compared to the potential time savings the automatic error testing modules provide.

The current Toastboard device has its power rail connected directly to a regulated 3.3V output from the CC3200 Launchpad for convenience. One problem with allowing an arbitrary user supply is the input range of the instrumentation ADCs. There is a tradeoff between allowable maximum voltage, scan delay, and sample resolution as described in the ADC pulldown network section.

Despite attempts to limit loading of the circuit by the instrumentation electronics a leakage path exists to ground during scans. This is evident in the examples when an LED voltage drop will be measured despite no user-wired ground path. The leakage current is on the order of microamps due to the large impedance of the pulldown network and would not influence the majority of circuits. This could be an issue with sensitive analog circuits (which are not our current focus), but the major practical concern would be any confusion this causes a more experienced user. Currently, the breadboard and power rails are connected to the Toastboard PCB with Z-axis electrical transfer tape (3M 9703). The conductivity of this adhesive is pressure sensitive, which can lead to intermittent connectivity during handling. We are investigating more reliable methods of ensuring stable contact.

The system rounds measurements during the full board scan to the nearest 100mV and during the oscilloscope mode to the nearest 10mV due to the competing nature of sample speed and resolution. Increasing the resolution without affecting speed by using a different ADC architecture and/or driving circuit is being investigated.

Tester Functions

Due to the relatively low sampling rate of the board in continuous refresh mode, we currently only support checks for static conditions (on or off, proper connectivity, proper supply voltage, etc.). This limits our ability to provide useful information about time-varying components such as capacitors and inductors. We leave the creation and evaluation of time-sensitive tests for future work.

Because we allow the construction of arbitrary circuits on the Toastboard, more advanced developers may use components in unforeseen ways. Although we provide a good baseline of commonsense tests, there may be many error conditions that our tests do not detect, or situations in which an error is reported where there is no problem with the circuit. More comprehensive studies of common user errors in building circuits, a crowd-sourced library of testers similar to the part library in Fritzing, or the ability for instructors to add their own custom checkers, might address this issue.

CONCLUSIONS AND FUTURE WORK

This paper presented the Toastboard, a system designed to simplify the process of circuit debugging. We used a series of formative interviews to identify key problems faced by both novice and advanced circuit builders to inform the design of an integrated circuit debugging platform consisting of a ubiquitously instrumented breadboard, a web-based circuit visualization, and an approach for flexibly creating circuit behavior testers. The Toastboard system was received positively by users in an informal evaluation. While the Toastboard is useful for learning to assemble and validating simple circuits, more complex systems involve rapidly changing values and embedded software whose behavior cannot be captured by the current board. In the future, we hope to explore and evaluate new solutions to address debugging in these new domains.

ACKNOWLEDGEMENTS

This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and by NSF CNS 1505728 and IIS 1149799.

REFERENCES

1. http://www.adafruit.com, 2016. [Online; accessed 26-July-2016].
2. Akiyama, Y., and Miyashita, H. Projectron mapping: The exercise and extension of augmented workspaces for learning electronic modeling through projection mapping. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST'14 Adjunct, ACM (New York, NY, USA, 2014), 57–58.
3. Autodesk. Circuits. https://circuits.io/, 2016. [Online; accessed 26-July-2016].
4. Booth, T., Stumpf, S., Bird, J., and Jones, S. Crossed wires: Investigating the problems of end-user developers in a physical computing task. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM (2016), 3485–3497.
5. Cao, J., Kwan, I., White, R., Fleming, S. D., Burnett, M., and Scaffidi, C. From barriers to learning in the idea garden: An empirical study. In *Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on*, IEEE (2012), 59–66.
6. Chan, J., Pondicherry, T., and Blikstein, P. Lightup: An augmented, learning platform for electronics. In *Proceedings of the 12th International Conference on Interaction Design and Children*, IDC '13, ACM (New York, NY, USA, 2013), 491–494.
7. Chmiel, R., and Loui, M. C. Debugging: from novice to expert. In *ACM SIGCSE Bulletin*, vol. 36, ACM (2004), 17–21.
8. http://www.circuitlab.com, 2016. [Online; accessed 26-July-2016].
9. Conradi, B., Lerch, V., Hommer, M., Kowalski, R., Vletsou, I., and Hussmann, H. Flow of electrons: An augmented workspace for learning physical computing experientially. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, ACM (New York, NY, USA, 2011), 182–191.
10. Digilent. Digilent electronics explorer board, 2016. Online; accessed 30-March-2016.
11. Gugerty, L., and Olson, G. Debugging by skilled and novice programmers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, ACM (New York, NY, USA, 1986), 171–174.
12. Hartmann, B., MacDougall, D., Brandt, J., and Klemmer, S. R. What would other programmers do: suggesting solutions to error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), 1019–1028.
13. Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., and Lui, D. A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *Trans. Comput. Educ. 14*, 1 (Mar. 2014), 1:1–1:20.
14. Knörig, A., Wettach, R., and Cohen, J. Fritzing: a tool for advancing electronic prototyping for designers. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, ACM (2009), 351–358.
15. Ko, A. J., and Myers, B. A. Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2004), 151–158.
16. Ko, A. J., Myers, B. A., and Aung, H. H. Six learning barriers in end-user programming systems. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, IEEE (2004), 199–206.

17. Lieber, T., Brandt, J. R., and Miller, R. C. Addressing misconceptions about code with always-on programming visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 2481–2490.

18. Martin, F. G. Ideal and real systems: A study of notions of control in undergraduates who design robots. In *In Y. Kafai and M. Resnick (Eds.), Constructionism in Practice: Rethinking the Roles of Technology in Learning*, Citeseer (1996).

19. Mellis, D., Banzi, M., Cuartielles, D., and Igoe, T. Arduino: An open electronic prototyping platform. In *Proc. CHI*, vol. 2007 (2007).

20. Mellis, D. A., Buechley, L., Resnick, M., and Hartmann, B. Engaging amateurs in the design, fabrication, and assembly of electronic devices. In *Proceedings of the Designing Interactive Systems Conference*, DIS '16, ACM (New York, NY, USA, 2016).

21. Mellis, D. A., Jacoby, S., Buechley, L., Perner-Wilson, H., and Qi, J. Microcontrollers as material: Crafting circuits with paper, conductive ink, electronic components, and an "untoolkit". In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*, TEI '13, ACM (New York, NY, USA, 2013), 83–90.

22. Nagel, L. W., and Pederson, D. O. *SPICE: Simulation program with integrated circuit emphasis*. Electronics Research Laboratory, College of Engineering, University of California, 1973.

23. Ochiai, Y. *Virtual, Augmented and Mixed Reality. Applications of Virtual and Augmented Reality: 6th International Conference, VAMR 2014*. Springer International Publishing, Cham, 2014, ch. Visible Breadboard: System for Dynamic, Programmable, and Tangible Circuit Prototyping with Visible Electricity, 73–84.

24. Portugal, R. Breadboard for electronic components or the like, Aug. 14 1973. US Patent D228,136.

25. http://www.sparkfun.com, 2016. [Online; accessed 26-July-2016].

26. Tawfik, M., Sancristobal, E., Martin, S., Gil, R., Diaz, G., Colmenar, A., Peire, J., Castro, M., Nilsson, K., Zackrisson, J., et al. Virtual instrument systems in reality (visir) for remote wiring and measurement of electronic circuits on breadboard. *Learning Technologies, IEEE Transactions on 6*, 1 (2013), 60–72.

27. Tetteroo, D., Soute, I., and Markopoulos, P. Five key challenges in end-user development for tangible and embodied interaction. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, ACM (2013), 247–254.

28. Van den Bout, D. E., Morris, J. N., Thomae, D., Labrozzi, S., Wingo, S., and Hallman, P. Anyboard: An fpga-based, reconfigurable system. *IEEE Design & Test of Computers*, 3 (1992), 21–30.

29. Victor, B. Seeing spaces, 2014. Accessed 10-04-2016.