Wire Your Way: Hardware-Contextualized Guidance and In-situ Tests for Personalized Circuit Prototyping

Punn Lertjaturaphat*
KAIST
Department of Industrial Design
Daejeon, Republic of Korea
punnlert@kaist.ac.kr

Jungwoo Rhee*
KAIST
Department of Industrial Design
Daejeon, Republic of Korea
jwoorhee@kaist.ac.kr

Jaewon You
KAIST
Department of Industrial Design
Daejeon, Republic of Korea
ampersand328@kaist.ac.kr

Andrea Bianchi
KAIST
Department of Industrial Design
Daejeon, Republic of Korea
andrea@kaist.ac.kr

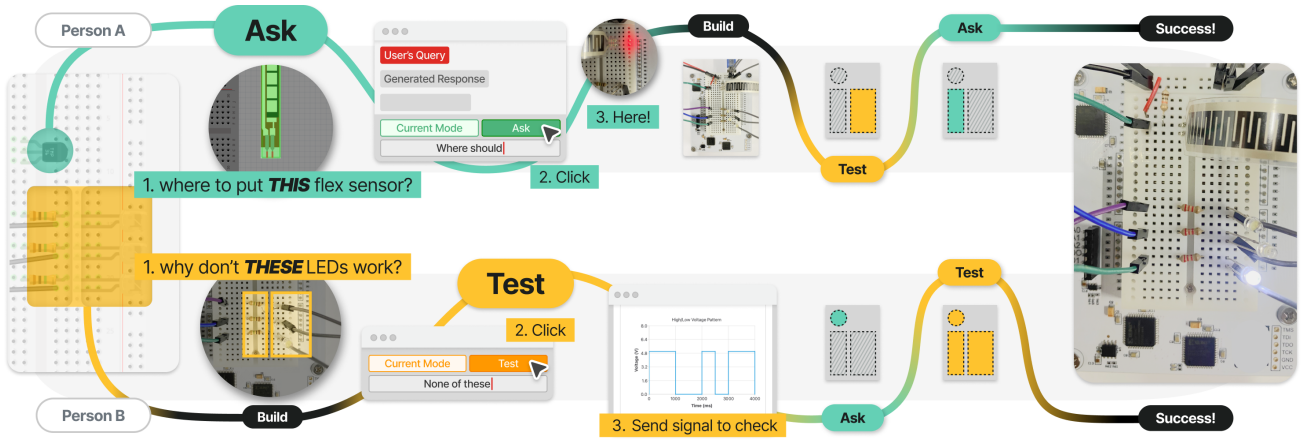

Figure 1: Two distinct personalized circuit-building workflows enabled by WireWay. Person A employs context-aware guidance by asking where to place a flex sensor, and receives highlighting assistance on the augmented breadboard. Person B follows a build-first approach, constructing the circuit from a schematic, asking "why don't these LEDs work?" and receiving automated test generation. Both workflows demonstrate successful circuit completion through individualized interaction strategies.

Abstract

The increasing popularity of microcontroller platforms like Arduino enables diverse end-user developers to participate in circuit prototyping. Traditionally, follow-along tutorials serve as an essential learning method for makers, and in fact, several prior toolkits leveraged this format as a way to engage new makers. However, literature and our formative study (N=12) show that makers have unique preferences regarding the construction of their circuits and idiosyncratic ways to assess and debug problems, which contrasts with the step-by-step instructional nature of tutorials and those systems leveraging this method. To address this mismatch, we present a prototyping platform that supports personalized circuit construction and debugging. Our system utilizes an augmented breadboard, which is circuit-aware and supports on-the-fly hardware reconfiguration via contextualized guidance and in-situ circuit validation through interactive tests. Through a usability study (N=12), we demonstrate how makers leverage circuit-aware guidance and debugging to support individual building patterns.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**; • **Hardware** → *Analysis and design of emerging devices and systems*; • **Applied computing** → Computer-assisted instruction.

Keywords

Physical computing, Circuit prototyping, Maker tools, Hardware development

*Both authors contributed equally to this research.

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/2026/04
https://doi.org/10.1145/3772318.3791371

ACM Reference Format:
Punn Lertjaturaphat, Jungwoo Rhee, Jaewon You, and Andrea Bianchi. 2026. Wire Your Way: Hardware-Contextualized Guidance and In-situ Tests for Personalized Circuit Prototyping. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3772318.3791371

1 Introduction

The growing popularity of microcontroller platforms like Arduino[1] has enabled diverse end-user developers to create interactive electronic projects [7] and has contributed to the widespread adoption of *physical computing* [54]. Creators often utilize popular open-source tools, such as the authoring editor Fritzing [35], to easily design electronic diagrams and then assemble circuits on physical breadboards. In this process, step-by-step instructions gathered online on how to design, modify, prototype, and debug the desired circuit are crucial. Tutorials are particularly effective in helping break down complex circuit construction tasks into manageable steps, and this strategy has been adopted by numerous toolkits and research prototypes that scaffold circuit construction via interactive tutorials [15, 56, 63] and precompiled guided tests to verify the correctness of the final prototypes [29, 33, 49].

While these structured tutorials and tools are effective in breaking down complex circuit assembly tasks into manageable steps, this approach, however, conflicts with the practitioner's natural building processes and individual preferences [18, 51]. In the real world, creators employ highly idiosyncratic and bottom-up prototyping strategies, from selecting suitable circuits from online resources based on the availability of components to determining a specific order of implementation based on individual learning goals or iterative processes [10]. As such, while tutorials clearly scaffold users, they fail to leverage the creators' sense of agency and intuition and do not foster the development of problem-solving techniques or learning [51].

This mismatch between prototyping tools and user behavior creates barriers that prevent effective exploration and debugging of physical computing projects. The root of this problem lies in the fragmented nature of existing support systems, which address specific aspects of physical computing development but fail to provide integrated solutions that adapt to individual circuit contexts. Tutorial-based systems [30] and generative approaches [3] provide structured guidance, but they require makers to follow predetermined paths that may not align with the intended circuit topology. Circuit visualization tools [66, 67] excel at representing electrical states but operate independently from the construction process itself. Augmented breadboard systems [8, 32] demonstrate promising spatial guidance capabilities through LED-based instruction delivery, yet they remain disconnected from circuit design software and testing practices. Debugging tools [29, 46] bridge the monitoring between hardware and software to allow user-authored testing, yet cannot assist users in building personal applications with progress in hardware context integrated instructions. In sum, existing solutions fall into two categories: offer guidance through structured tutorials that need to be pre-authored by an instructor, enforcing a specific way to prototype and test a circuit [e.g., 30, 63], or provide specific solutions or tools for the design [e.g., 3, 42], construction [e.g., 32, 62], or debugging [e.g., 19, 67] of electronic circuits, leaving the burden of deciding what to do and which strategy to adopt for different situations completely to the end developer.

To address this mismatch and bridge the gap, we present WireWay, an integrated development environment that enables personalized circuit prototyping workflow through hardware-contextualized guidance and in-situ testing capabilities, effectively linking the hardware with the digital representation of the system (Figure 1). Unlike fragmented existing tools, WireWay provides unified support by bridging physical computing design, construction, and validation processes. The system delivers adaptive guidance through a conversational agent that maintains real-time awareness of both visual circuit schematics and physical configurations. This dual awareness enables WireWay to provide circuit-specific visual cues directly on an augmented breadboard through real-time hardware sensing and automated testing capabilities. We demonstrate the design, implementation, and evaluation of WireWay through (1) a literature review and formative study ($N = 12$) that informed the system's design goals and technical requirements, (2) the implementation details of our integrated prototyping platform, and (3) a usability study ($N = 12$) demonstrating how hardware-contextualized guidance and validation enhance personalized circuit prototyping.

2 Related Works

Building physical computing systems requires makers to navigate complex relationships between code, schematics, and physical hardware while following an often non-linear and exploratory process. We survey prior work across three dimensions that shape the physical computing experience: maker workflows, hardware-aware instructions, and circuit validation.

2.1 Maker Workflows

Empirical studies consistently demonstrate that makers naturally employ highly personalized and exploratory prototyping strategies that promote a growth mindset and effective learning [10, 18, 51]. Theoretical frameworks like "reflection-in-action" describe design as a "thinking-by-doing" activity [57], emphasizing a conversational relationship between designer and medium where concrete prototypes lead to unexpected realizations [24, 34]. These learning principles, where knowledge is built through hands-on tangible construction, were proven effective for supporting diverse learners in physical computing education [27].

However, this natural approach often conflicts with existing tools that impose structured, predetermined paths. Makers face substantial challenges in physical computing, with most fatal faults due to incorrect circuit construction that are often misdiagnosed as software bugs [2]. Current educational approaches fail to provide comprehensive frameworks for identifying context-specific sources of bugs, highlighting systems thinking perspectives that support diverse debugging pathways [16, 48]. The complexity stems from needing to understand coding, electronics, and their interconnections, whereas current tools rely on error-prone manual processes that create barriers to scaling beyond prototypes [23]. Research

[1] https://www.arduino.cc/

reveals that creative engineering work occurs during system architecture, yet tools operate at lower abstraction levels, creating tedious work and calling for more adaptive approaches [7, 42].

While existing research advocates for personalized workflows and documents these challenges, available tools remain fragmented and misaligned with natural exploratory processes. WireWay directly addresses this by enabling personalized circuit prototyping through unified support that bridges design, construction, and validation while enhancing makers' exploratory nature.

2.2 Hardware-aware Instructions

Hardware guidance has evolved from structured to more adaptive, context-aware systems. While effective at breaking down complex tasks, tutorial-based methods [30, 63] require predetermined paths that may not match individual circuit topologies or preferences. Research into how-to videos shows the value of rich, contextual information for effective learning beyond mere instructions [13, 69]. Augmented breadboard systems offer spatial guidance through LED matrices that visualize component placement and electrical connections [8, 32, 53], representing a shift toward hardware-contextualized guidance with immediate visual feedback. Hardware redesigns [17] address cognitive load by bringing visibility to underlying breadboard connections, while CircuitStack [62] supports rapid and iterative circuit evolution. Intelligent assistance tools include AutoFritz [44] for circuit autocomplete, Trigger-Action-Circuits [3] for generative design from behavioral descriptions, and CircuitStyle [15] for reinforcing construction best practices.

Advanced approaches focus on bridging abstraction levels and offering novel interaction paradigms through different modalities. Conversational agents like FritzBot [12] take natural language descriptions from novice users and dynamically generate corresponding Arduino circuits and code, addressing component selection challenges. Bani Yusuf et al. [6] automates Arduino programming by generating hardware configurations and API usage patterns from natural language queries. Visual programming environments such as Flowboard [11] introduce flow-based programming that is conceptually closer to circuit diagrams than imperative code, providing immediate feedback that better reflects circuit behavior. High-level design tools [41] and Hardware Description Language-based editors [43] allow designers to work while maintaining synchronization with visual representations. Hybrid approaches, like VirtualComponent [31], assist with component value tuning through augmented reality (AR) overlays on physical breadboards, and VirtualWire [40] enables programmatic reconfiguration of breadboard connections. SpatIO [22] uses XR to support spatial component placement with virtual-to-physical transitions, while Proxino [65] blends virtual and physical circuit elements to facilitate distributed prototyping collaboration. Finally, sensor-specific tools like SensorViz [33] provide visualization across prototyping stages, from datasheet specifications to AR-based environmental interaction.

In summary, while there are rich hardware-aware systems and tools, they often require predetermined paths and operate in isolation from the real-time physical build context. WireWay offers a distinct advantage by providing adaptive guidance through a conversational agent that maintains awareness of both visual circuit schematics and physical configurations. This dual understanding enables to deliver contextualized support and circuit-specific visual cues directly on an augmented breadboard in a way that is integrated and responsive to the maker's actual physical progress.

2.3 In-Situ Circuit Validation

Physical computing debugging is complex as bugs can reside in software, hardware, or their intersection, with novices often misdiagnosing hardware issues as software problems [10]. Circuit visualization tools [66, 67] provide real-time flow visualization and automatic component recognition, but operate independently from construction workflows.

Integrated debugging environments bridge hardware-software gaps through systems like Bifröst [46] for linked visualizations, Heimdall [29] for remote inspection, and Toastboard [19] for ubiquitous instrumentation. Test-driven approaches like ElectroTutor [63] extend software engineering concepts to hardware, but often require tests to be pre-designed by an instructor, which cannot be universally applied to any circuit. HeyTeddy [30] automatically suggests users complete tasks throughout a tutorial, but also requires these tutorials and tests to be pre-specified. Many solutions necessitate external instrumentation or separate monitors, removing information from the immediate code context. Recent developments address these limitations through in-context approaches like Inline [9] for direct code editor visualization, WiFröst [47] for networked system instrumentation, and conversational debugging [4] for guided localization through natural language interaction.

Many debugging visualizations remain decoupled from the immediate construction context, requiring users to mentally map between separate interfaces and their physical circuits, or rely on complex setups or pre-existing exercises and tests compiled by an instructor. WireWay addresses these limitations by providing a unified debugging experience within the circuit prototyping procedure. Our approach provides automated testing tailored specifically to individual hardware through real-time input and output reads without additional setup or pre-made content.

3 Formative Study

We conducted a formative study to explore makers' challenges when developing and debugging physical computing systems with embedded platforms such as Arduino. We recruited twelve participants (7 female, 5 male) via our institution's online community posting, with an average age of 23.75 ± 3.05 years (range 19-29 years). We report means with standard deviations as M ± SD. They had diverse backgrounds in electrical engineering ($n = 2$), computer science ($n = 2$), mechanical engineering ($n = 2$), industrial design engineering ($n = 5$), and an exploratory major ($n = 1$). Participants reported an average of 2.29 ± 3.47 years of experience with embedded systems (range <1 to 13 years), see Table 1 for details.

Method Each study session lasted approximately 80 minutes and comprised three parts: introduction and consent, a demographics survey (10 minutes), a circuit-building task (50 minutes), and a semi-structured interview (20 minutes). Participants received compensation equivalent to $15 USD in local currency.

Circuit Prototyping Task We adapted *Project 4: Color Mixing Lamp* from the Arduino Projects Book [21] as it was rated a beginner-to-intermediate, recommended for 45 minutes. The task

Table 1: Participant demographics and prior experience

ID	Gender	Age	Major	Coding Exp.	Program Lang.	Phys. Comp. Exp.
P1	M	29	EE	13 yrs	SystemVerilog	13 yrs
P2	M	21	CS/Math	3–4 yrs	C	<1 yr
P3	F	26	EE	3–4 yrs	Verilog	<1 yr
P4	F	19	Exploratory	1–2 yrs	Python	<1 yr
P5	F	20	CS	4–5 yrs	Python	1–2 yrs
P6	F	21	ME	1–2 yrs	Python	<1 yr
P7	F	25	ID	<1 yr	Python	<1 yr
P8	F	23	ID	1–2 yrs	Python	2–3 yrs
P9	M	27	ME	2–3 yrs	C++	2–3 yrs
P10	F	24	ID	2–3 yrs	Python	1–2 yrs
P11	M	26	ID	1–2 yrs	C#	1–2 yrs
P12	M	24	ID	3–4 yrs	JavaScript	2–3 yrs

required reading brightness values from three photoresistors and control an RGB LED's red, green, and blue channels. We allotted 50 minutes with a 5-minute grace period and provided an official schematic from the book. Participants used the internet for information search and debugging, but were restricted from using large language models beyond Google's default AI-summarized results.

Task Result Four of twelve participants successfully built and coded the circuit. The participants who succeeded took an average time of 42'00" ± 6'47" minutes. The reasons for failure consist of 1) misidentifying the type of the LED (common cathode to common anode), 2) wrong resistance value to construct the photoresistor voltage divider, and not accommodating for it in the code, 3) misconstructing the voltage divider for the photoresistor, 4) code discrepancy (wrong functions, not declaring pin mode).

Semi-Structured Interview The following debrief examined participants' information practices during the task, focusing on (1) tutorial search strategies, (2) format preferences and switching, (3) trustworthiness and accuracy judgments, (4) strategies for keeping track of multiple information sources, and (5) approaches to verification, error recovery, and reflection (Appendix A). We recorded physical circuit-building processes during the task, screen-captured programming and information searches, and voice-recorded the interview sessions. After transcribing and translating the interviews into English, we applied open, axial, and selective coding for qualitative analysis [61]. As a result, we introduce three key challenges identified across participants during physical computing projects, particularly highlighting the mismatch between existing tool design and makers' natural building processes.

3.1 Challenge 1: Distinct Building and Information Retrieval Strategies

Participants showed idiosyncratic strategies in interactive circuit prototyping, including starting component choices and task segmentation, that is to say, there was no single pattern shared across all participants. Some participants broke down complex tasks into smaller, manageable segments that reflected their individual problem-solving approaches (P1-P2, P9-P12). For example, P1 described breaking down the task into reading the photoresistors' brightness value first, setting up the LED, and later merging them. P2, P9, P11, and P12 echoed this incremental segmentation strategy; P9 referred to *"dividing the task into some fundamental elements,"* while P12 used the metaphor of doing *"baby steps, little achievements along the way."* P10 separated the task into component types and tried using a single photoresistor first, then added the remaining two photoresistors. In stark contrast, other participants (e.g., P4, P6-P7) chose to build everything at once without breaking down the circuit into smaller parts or tasks. They preferred to build the entire circuit and test it at once at the end. This was not a particularly effective strategy, as acknowledged by P4: *"I think that was a problem... I have to basically fix all of the parts at once."* However, P4 recognized the value of component-level testing for more complex projects, explaining: *"For [what I'm currently working on]... just at first try to test one motor and one wire and then if that is right, then I use the same value to calculate the whole thing."*

When we asked how they previously selected relevant tutorials and information for the circuit, we also received descriptions of numerous and diverse approaches. Some participants prioritized images (P1-P2, P6, P8-P10, P12), while others preferred video tutorials because they supported *"understanding the whole process"* (P4) but also because they could pause them anytime if they wanted to screen capture images of circuits or schematic diagrams (P3, P5-P6, P10, P12). Another notable difference among participants was the choice of preferred language, as not all participants were native English speakers. As such, people (P4, P7-P8, P11) explained feeling more comfortable using their native language for information retrieval. P4 explained their keyword selection strategy: *"I chose my keyword based on my experience because I'm not familiar with English, so I learned in school how this [photoresistor] is called in [my mother tongue]."* There was also the issue of domain-specific languages (i.e., the technical jargon or the name of specific hardware components), as not every participant shared the same engineering knowledge. Participants referred to elements in this circuit using pronouns or determiners such as "it," "this," and "that" (P3, P5, P7, P10-P12). P10 highlighted the communication challenges when searching online: *"It's hard to communicate what this is in texts on the Internet, saying 'What is this box with two arrows?', or 'What does that mean?' They won't understand what that is."*

Takeaway: Our interviews revealed that creators adopt very different strategies for constructing circuits or searching for relevant information online, and that they all value different types of information (images or videos) or choose vocabulary that reflects their prior knowledge, preferences, and expertise.

3.2 Challenge 2: Mismatched Between Circuit and Tutorial Diagrams

All participants discussed how differences between the visual representation of a circuit and the circuit diagram, or even between components on their breadboard and those in the tutorials, created confusion and misunderstanding. This separation between equivalent representations (i.e., digital vs. physical) was already studied in the literature [32] and was further corroborated in our findings. For example, nine participants reported struggling to identify components and pins (P1-P4, P7-P10, P12) when tutorial visuals did not match their prior knowledge or the components' physical appearance. P1 experienced wiring errors due to incorrect assumptions about LED pin ordering, initially expecting "ground, red, green, blue" but finding the actual arrangement was "red, ground, green, blue." Two participants (P6, P7) expressed frustration with the way

power connections (VCC and GND pins) were rerouted directly to the components without passing through the breadboard, or how different components should have been grounded to the same node of the net. Similarly, P3 made a *"huge mistake"* by misidentifying photoresistors as a diode in the provided schematic, stating, *"I was familiar with arrow symbols for diode,"* and only realizing the error after watching a clarifying video. P12 observed that the diagram in chosen tutorial had pins arranged differently from what they had built based on general knowledge, leading them to tear down and rebuild the circuit to match the tutorial exactly.

Another source of frustration was identifying components and their intended usage. P2 observed that different online sources often presented conflicting visual representations, *"the documentation... was different from what you're expecting from the components that you have""*. Several participants (P2, P8, P10-P11) struggled with tutorial applicability when similar components were used for different purposes, *"two tutorials look similar but were made for different purposes"*–P8. As a solution, users turned to the text descriptions in the tutorial to get a better grasp of the circuit topology and its intended usage (P4, P6-P11), leaving some to conclude (P10, P11) that constant adjustment and modification are needed to extract relevant information from tutorials.

Takeaway: All the above comments highlight the frustration of creators, especially those with little engineering expertise, in forming a clear mental model that bridges the diagram representation of a circuit and its physical instantiation in the hardware. Any mismatch in terms of visual representation between the two, in terms of circuit structure or components, creates uncertainties, confusion, or simply slows down the process and confidence with which the circuit can be assembled.

3.3 Challenge 3: Manual and Iterative Circuit Debugging

Circuit validation emerged as the most significant barrier to successful physical computing prototyping. Multiple participants (P5, P6, P7, P8) lacked systematic testing knowledge and resorted to guesswork rather than methodical verification approaches. P6 explicitly acknowledged this limitation: *"I think it's hard to do [verify]. So it's more of a guess that it will work"*, while P7 admitted forgetting fundamental testing procedures: *"But I forgot that there is that kind of [testing individual components before building the whole thing] process."* This finding aligns with prior work documenting debugging challenges in physical computing education, where circuit construction errors often compound software bugs, creating complex failure scenarios that novices struggle to diagnose systematically.

Existing testing approaches required external tools that operated independently from circuit design contexts, contrasting sharply with participants' requests for integrated solutions. While participants relied on disconnected debugging methods–serial monitor observations (P1-P2, P4-P5, P9-P11), manual component commanding through temporary code modifications (P2, P9), or external measurement tools like multimeters (P1)–six participants (P1-P3, P5, P10, P12) explicitly desired automated validation systems integrated directly into circuit design workflows. P9 exemplified current fragmented approaches: *"turn[ing] each LED one by one directly through the code without anything connected... turning each (LED) 'high' one by one"* to verify connections, while P10 conceptualized integrated alternatives: *"a LEGO-style tutorial with step-by-step checkpoints that would show the simulated values of a circuit that should be expected."* However, even general-purpose AI assistance proved inadequate, as P10 experienced: *"There were times when I followed everything that the [AI] was telling me, and it still didn't work... they kept telling me very general answers that I had already checked."* P5's request for immediate contextual feedback—*"if the program could tell if I'm wiring the components wrong"*, alongside P1's advocacy for *"built-in instrumentation"* with *"measurement ports"*, highlights the need for circuit-aware debugging support that provides specific, actionable guidance rather than requiring manual correlation of disparate information sources or generic troubleshooting advice.

Takeaway: Overall, users expressed frustration in identifying the source of errors and showed interest in methods that would be aware of the circuit under test, suggest strategies to narrow down the possible cause of errors, and guide them through strategies to measure the expected output.

4 WireWay System

WireWay is a circuit design and construction support tool that makes the realization of a circuit design easier by connecting software visualization with physical circuits and integrating expressive agent chat queries with context inclusion (Figure 2). The system was structured in response to the circuit realization challenge noted during the literature review and the formative study. WireWay was designed with three goals: 1) **let the user ask questions about their own circuit and components** without having to conform to any preprocessed tutorial; 2) the user can receive **guidance on where to physically wire and/or apply tests directly in the hardware**; and, 3) **the system can suggest and perform hardware tests** that help the user find faults in a circuit by probing it (see Appendix B for details).

4.1 Design Goal 1: Supporting Idiosyncratic Prototyping

To better scaffold makers in their own circuit modifications and explorations, WireWay provides personalized guidance that adapts to the current circuit configuration at any arbitrary starting point. By integrating a written query pipeline with included circuit context into the system interface (Figure 3), the user can have an interactive conversation about the construction strategy and process for this specific circuit, or the characteristics (e.g., pinout) of a specific component. With the circuit's context, the user can ask general questions about the circuit configuration (e.g., *Is the circuit correct? Are there any misconnections?*), and by visually selecting specific components or a portion of the circuit on the interface (Figure 3A, B), direct questions about the highlighted parts alone. Context-awareness also allows users to refer to components using generic pronouns and terms such as "this," "here," and "these" in queries (Figure 3C and 2C). For example, the user can ask *What is this component?* or *How shall I connect this component?*, or *Where is the ground pin of this LED?* and the system can produce an informed response using as context the current circuit topology and the specified subset of components highlighted through the system interface (Figure 3D).

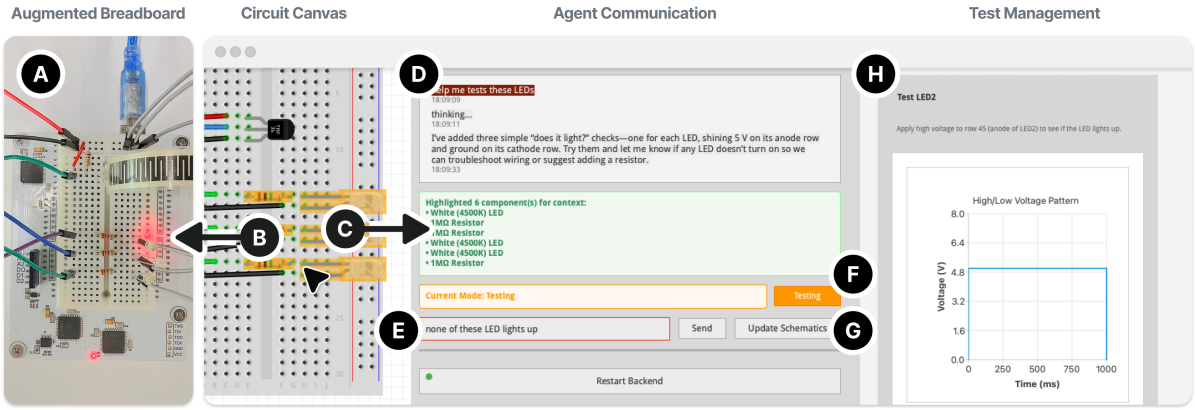

Figure 2: Overview of WireWay interface architecture: (A) an augmented breadboard with LED row indicators for physical guidance, (B) a circuit design canvas enabling circuit configuration, visualization, and breadboard highlighting, (C) component selection for conversational context reference, and (D) an AI agent communication. Users interact through (E) natural-language text input to query circuit configurations and components. The interface provides (F) mode switching between *Ask* and *Test*, (G) schematic synchronization to update circuit context, and (H) ad-hoc test for systematic hardware verification.

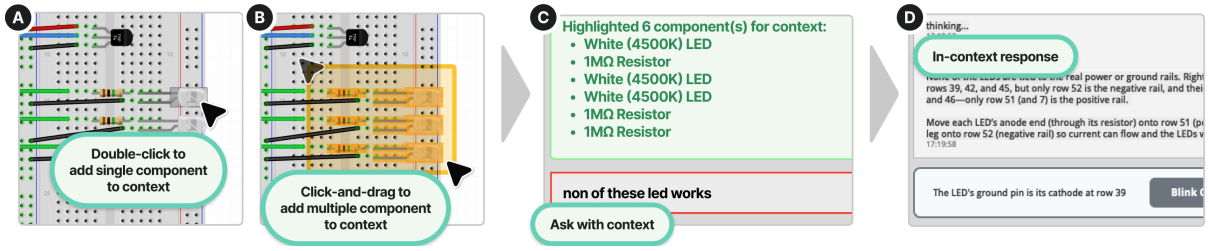

Figure 3: Context-aware component selection workflow for conversational interaction. Users can add circuit components to the conversational context through (A) double-click individual components in the circuit design interface, or (B) click-and-drag for multiple components. (C) With highlighted components as added context, users can query using natural language and pronoun references. (D) The system parses the contextual component information to provide targeted guidance and responses.

These mechanisms address Challenges 1 and 2 from the formative study by providing ways for individual builders to have different execution strategies, formed through a dialogue with the agent, for any arbitrary tutorial that serves as a starting point.

4.2 Design Goal 2: Guiding with Hardware-Context Awarness

WireWay supports in-context physical hardware guidance by utilizing an open-sourced augmented breadboard with a built-in LED row indicator. WireWay can make an explicit row indication to the user by blinking the LED embedded in rows that it needs to make reference to (Figure 4C). The system supports this position visualization in two ways (Figure 4): **1) Active Mode** The user clicks on a specific component on the GUI breadboard and see the rows on the augmented breadboard lit up corresponding to the location where the selected component should be (Figure 4A), similar to the previous works [32]. **2) Dialog Mode** Another option is through dialogue with the agent, where the user asks the agent to highlight a specific pinout of an IC or simply highlight the connections that the user should make. Then the system will display a GUI button

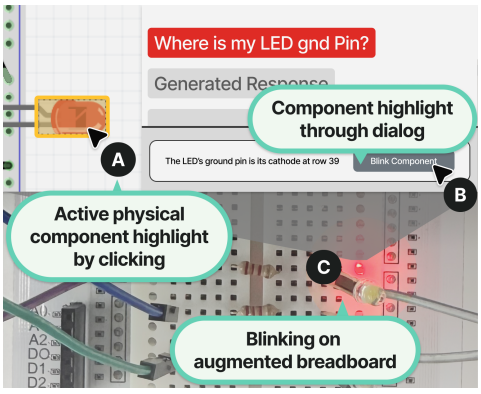

Figure 4: Two methods for providing in-situ breadboard guidance: (A) users click circuit components to illuminate corresponding breadboard rows. (B) the conversational agent automatically activates (C) LED indicators and provides interface buttons to repeat the highlighting sequence.

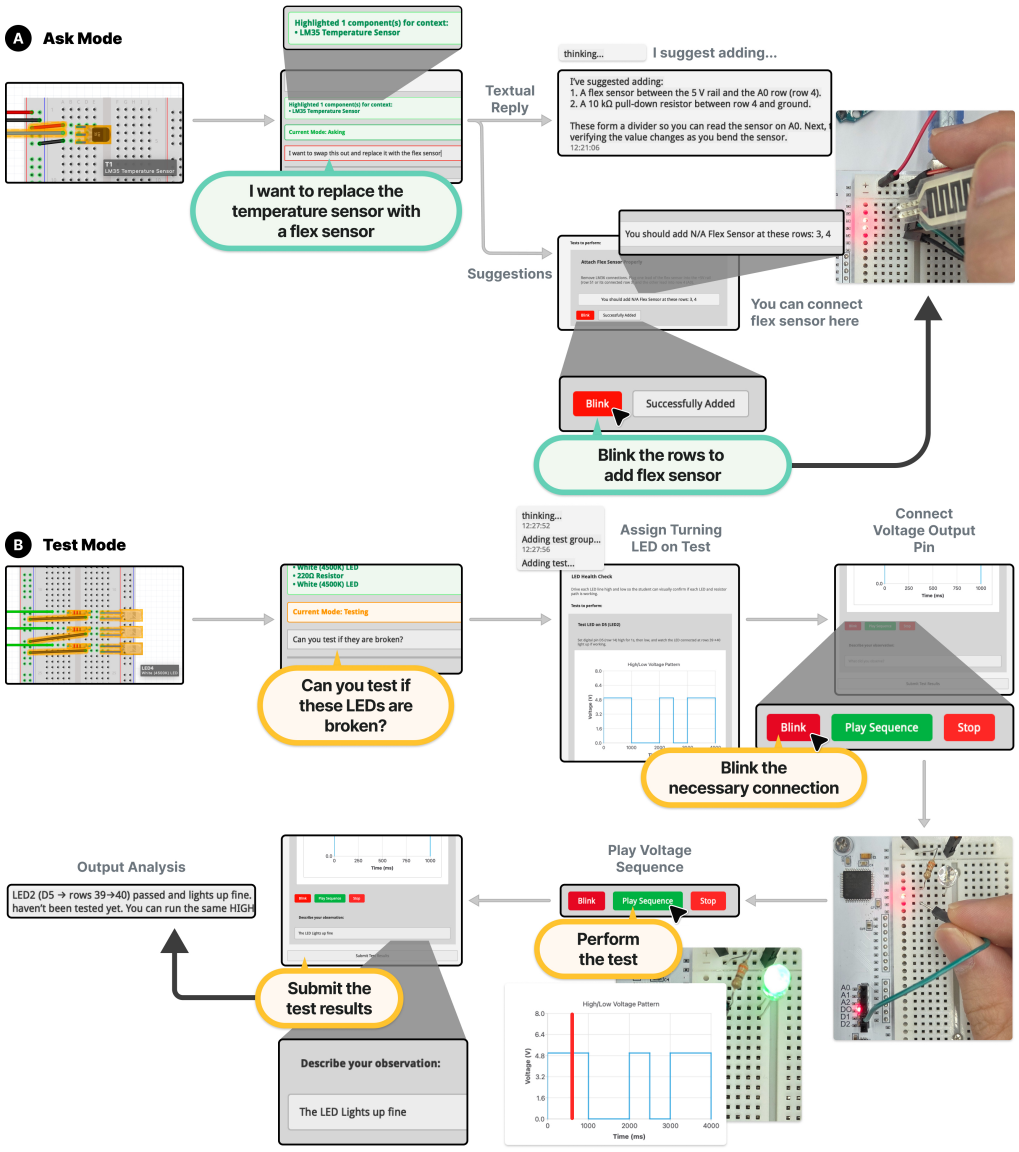

Figure 5: Interaction flow of WireWay. (A) *Ask Mode*: users can select components in context and submit a query. The system responds with either textual guidance (no physical action required) or suggestions highlighting where to modify the circuit (Figure 6B). (B) *Test Mode*: users can select which components to test. The system generates test procedures, groups related tests, and may require probe connections (Figure 6A). Users can again highlight required I/O pins before running a test sequence. Users can submit results to receive processed feedback or interpretation.

under the chatbox, where the user can click to light up on the physical breadboard's rows of interest (Figure 4B). For example, the user might select a 555 IC and ask the system to display where the "discharge" pinout is on the physical breadboard.

These interactions address Challenge 2 from the formative study by providing a one-to-one, easy-to-follow, explicit mapping for the user to realize the software design in physical hardware. And an explicit way that the system can make reference to the in-situ location of the component.

4.3 Design Goal 3: Verifying Circuit via In-Situ Tests

As a response to the formative study Challenge 3, WireWay allows users to identify circuit errors via a set of tests dynamically generated to suit the current circuit, without requiring end developers to augment the code or instructors to prepare tests in advance. Specifically, WireWay helps users verify the current hardware configuration and, if it contains a potential mistake, suggests a fix accordingly. As a response to the formative study's Challenge 3,

WireWay allows users to identify circuit errors via a set of tests dynamically generated to suit the current circuit, without requiring end developers to augment the code or instructors to prepare the tests in advance. Specifically, WireWay helps users verify the current hardware configuration and, if it contains a possible mistake, suggests to fix accordingly. There are two modes in which the system can operate: **1) *Ask mode***, the default mode where the system's goal is to provide information about the circuit and is limited to only making suggestions without assigning any tests. Figure 5A shows how user could inquire the system on how they should replace the temperature sensor as well as how the system would response with text and suggestions; or **2) *Test mode***, where the system's goal shifts to help users localize bugs by providing physical circuit tests that ask the user to perform an action and collect relevant data. For example, Figure 5B shows how the user can request the system to test one of the LEDs in the circuit. The system replies back with a simple voltage output test that the user can perform. The user can highlight where to connect output probes with the "blink" button. After connecting all the required connections, the user then performs the test and sees how the LED responds to the voltage output graph. After seeing that the LED lights up fine, the user can submit the result, and the system replies with a result analysis. The user can choose the system operation mode via a toggle button (Figure 2F).

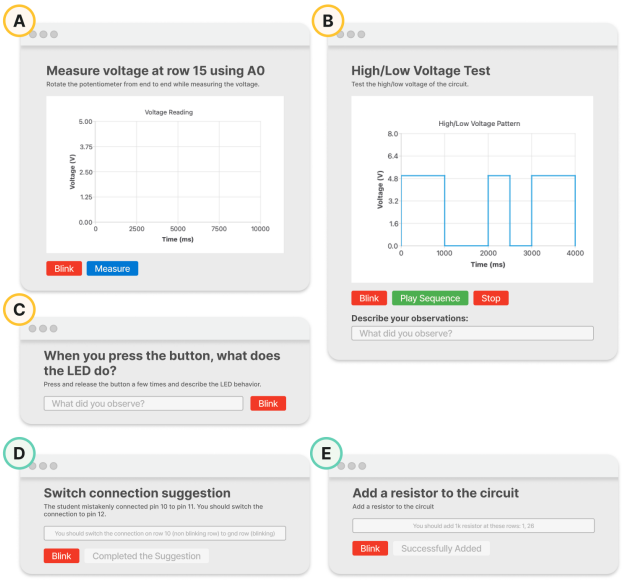

Figure 6: Five types of adaptive tests and suggestions generated by WireWay. *Test Mode* provides three diagnostic test types: (A) Voltage Measurement Test for probing circuit voltages at specific locations, (B) High/Low Voltage Pattern Test for analyzing the circuit response to digital signal behavior over time, and (C) Visual Inspection Test for observing component responses to user interactions. *Ask Mode* generates: (D) Connection Suggestion for correcting miswired, and (E) Component Suggestion for adding missing circuit elements.

4.3.1 Test and Suggestion Types. The system provides 3 types of in situ tests and 2 types of suggestions that the conversation agent is capable of (Figure 6). These tests and suggestions are generated on demand upon user requests or suggested by the system as a result of a previous question. The three tests consists of **1) Voltage Measurement Test** is assigned when the system wants to probe a part of the circuit for its voltage value (Figure 6A) **2) Signal Output Test** is assigned when the system wants to see how a certain component respond to a digital high and low (Figure 6B). **3) Visual Inspection Test** is assigned when the system wants the user to observe a certain behavior of a circuit like whether an LED respond to a button press (Figure 6C); Voltage and Signal Output tests probe the existing circuit by generating a time-varying voltage signal through the augmented breadboard built-in Digital-to-Analog Converter (DAC). The two suggestions that the system can make are **1) Connection Suggestion**, which suggests the user to correct a possibly miswired connection at certain rows (Figure 6D); **2) Component Suggestion**, which suggests the user to add a component to the identified place on the breadboard (Figure 6E).

4.4 Implementation

WireWay integrates a modified Fritzing application [35] with unmodified BlinkBoard [8]; an open-sourced extended breadboard featuring row-indication LEDs and analog/digital I/O pins. The BlinkBoard [8] was chosen for simplicity in both user interaction and software implementation, controlled over a USB serial communication using JSON command format. The control commands consisted of: (1) executing row-indicator LED patterns (on, off, blinking, or blinking slowly); (2) outputting voltage or pulse-width modulation (PWM) from output pins; and (3) reading voltage inputs from the input pins. The architecture comprises: (1) a user frontend application developed as a Fritzing extension (Figure 7A); (2) a Node.js backend server to communicate with the OpenAI API and the hardware (Figure 7D), and (3) a BlinkBoard that communicates with the server (Figure 7E). The server effectively bridges between the Fritzing frontend (via stdin/stdout), the augmented hardware (via serial), and the OpenAI o4-mini model[2].

The Fritzing frontend extension (Figure 7A) provides an interface for the user to 1) enter a query prompt; 2) select between 'ask' and 'test' mode; 3) add component contexts to the query prompt; 4) update the schematic configuration context; 5) interact with all the tests; and 6) control the augmented breadboard LED activity and I/O pins. All of which are sent to be processed in the backend. To include component context in the query, the frontend application extracts component metadata, including component identifiers, names, and pin connection mappings, and passes it to the server along with the subsequent query that the user sends. The frontend displays the response generation status like "thinking" or "adding tests" (Figure 7B) to give visual latency feedback. The frontend was developed and implemented in C++, using the QT framework.

The backend (Figure 7D) handles forwarding and structuring the user's query prompt to the OpenAI response API. Before sending the query, the server evaluates system state changes (e.g., mode changes, schematic updates) and incorporates relevant component context (if any) into the conversation array with the 'developer' role

[2]https://openai.com/api/

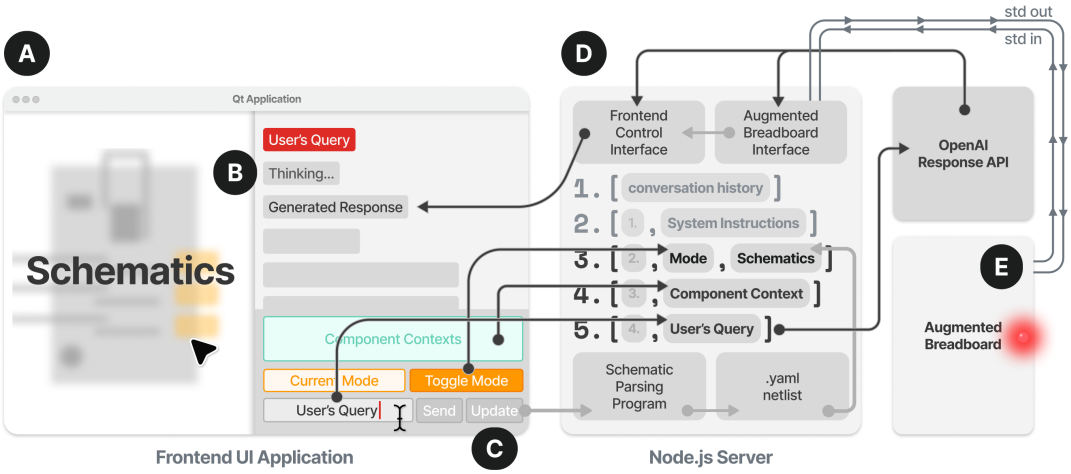

Figure 7: WireWay system architecture showing the data flow and processing pipeline. (A) The frontend Fritzing extension captures user interactions and (B) displays system state while preparing answers or tests (C) and circuit designs, generating XML netlists that are converted to YAML format by the backend. (D) Node.js server maintains conversation history, mode state, schematic context, and component selections while orchestrating AI queries through OpenAI API integration. (E) The augmented breadboard receives control commands and component highlighting instructions for real-time physical guidance.

designation. User input messages are subsequently appended with the 'user' role designation. Initial system activation triggers the inclusion of comprehensive system instructions to establish operational parameters. The conversational agent extends its capabilities beyond textual responses through structured function calls that enable hardware control operations, including augmented breadboard LED activation sequences and schematic state queries. Function call implementations utilize predefined parameter schemas to ensure predictable agent behavior and maintain system reliability. Function call also allows the agent to generate context-aware tests for the user. Testing functions are exclusively available during *Test mode* operation, preventing inadvertent activation during standard conversational interactions. Each test function incorporates hardware constraint specifications within its parameter schema (e.g., valid breadboard row ranges 1-50, timing parameters specified in milliseconds) to ensure safe and bounded operation. Upon receiving the XML netlist from the frontend as a result of the user updating the schematic context (Figure 7C and 2G), the server passes the netlist to a Rust-based binary parser that processes the XML, removes redundant information (e.g., duplicates and empty rows), and converts it to YAML format.

The breadboard can be controlled via serial communication and JSON commands (Figure 7E). The server provides endpoints that accept augmented breadboard control parameters for the frontend to interface with. The conversational agent accesses identical functionality via structured function calls. After receiving a control request, the server generates the corresponding JSON and transmits it to the board via serial communication. The breadboard also supports voltage measurements through built-in ADC pins and digital signal output via configurable digital pins, with measurement results returned to the frontend application for display. The system achieves a latency of 8.23 ± 0.69 ms for voltage output commands and 12.25 ± 0.44 ms for analog read commands, measured across 500 repeated trials. Voltage values are addressed at millivolt resolution in the firmware, enabling theoretical precision to 1mV.

5 User Study

To evaluate how makers with diverse backgrounds adopt different physical prototyping practices, we conducted a usability study with WireWay. Our research question focused on understanding how users leverage hardware-contextualized guidance and in-situ testing capabilities during physical circuit prototyping, examining both individual workflow preferences and system effectiveness. We designed a controlled study where participants worked with the identical circuit containing deliberate bugs to observe and compare behavioral patterns and problem-solving approaches. Both quantitative metrics (completion times, error rates, feature usage, post-task survey) and qualitative insights through semi-structured interviews were collected to evaluate how the system supports personalized circuit construction workflows.

5.1 Participants

We recruited fifteen participants in total through our institution's online community post. Three were excluded from analysis: who could not understand the task within the allotted time; due to a system malfunction where the AI model failed to surface the required circuit bug; due to a conflict of interest. Our analysis, therefore, includes twelve physical computing makers (7 male, 5 female; age=24.92 ± 2.84). The participants had backgrounds in computer engineering/science (n = 2), electrical engineering (n = 1), industrial design (n = 7), culture technology (n = 1), and physics (n = 1); three of them participated in our formative study (P3: formative P1, P9: formative P10, P12: formative P8). All participants reported prior education in physical computing, moderate confidence in physical computing (2.9 ± 1.5 out of 5-point Likert scale), and similar confidence in software development (3.1 ± 1.2 out of 5-point Likert scale).

Table 2: Participant demographics and prior experience

ID	Gender	Age	Major	Programming		Physical Computing			AI Use
				Exp.	Lang.	Exp.	Edu.	Prj.	
P1	M	26	CE	4–5y	C++	3–4y	Univ. course	5+	ChatGPT[a, b]
P2	M	30	CS	4–5y	Go	1–2y	Univ. course	3	Copilot[c]
P3	M	29	EE	13y	C, SystemVerilog, Python	13y	Univ. course	5+	Chatbot[c]
P4	M	23	Physics	1–2y	Python	< 1y	High school	1	None
P5	M	27	ID, HCI	4–5y	Python	4–5y	Workshop, Univ.	5+	ChatGPT, Gemini[a, d]
P6	F	25	ID	3–4y	C	2–3y	Univ.	5+	ChatGPT[a]
P7	M	22	ID	< 1y	Python	1–2y	Univ.	2	ChatGPT[d]
P8	F	25	CT	5–6y	Java	5–6y	Self-taught	5+	ChatGPT[a]
P9	F	24	ID	1–2y	Python	1–2y	Univ.	2	ChatGPT[a]
P10	F	25	ID	3–4y	C	1–2y	Self-taught	2	ChatGPT[a]
P11	M	20	ID	4–5y	Python	4–5y	Univ., MOOC	5+	Claude, ChatGPT[b]
P12	F	23	ID	1–2y	C, Python	2–3y	Workshop, Univ.	4	ChatGPT[b]

[a] Circuit debugging and troubleshooting [b] Code generation or refinement [c] Data analysis/library assistance [d] Circuit assembly and design

All have successfully built more than 2.33 ± 1.03 physical computing projects; eleven participants have used a Large Language Model aid for hardware prototyping (Table 2).

5.2 Materials and Method

Each study session lasted approximately 90 minutes and consisted of five parts: informed consent with demographics survey (10 minutes), system introduction (5 minutes), a circuit building task (55 minutes), post-task evaluations (5 minutes), and a semi-structured interview (15 minutes). All participants received compensation equivalent to $15 USD in local currency. Two authors were present throughout each session, with one leading the study and providing system tutorials while the other documented participant activities and behaviors for analysis.

System Introduction After collecting demographic information, one author introduced participants to WireWay using a prepared software schematic in a live demonstration format. Participants were systematically walked through each system feature, including schematic editing, component highlighting, LED blinking functionality, *Ask mode* for conversational guidance, and *Test mode* for circuit validation. One author answered any questions about the system, ensuring participants understood all available functionality throughout the construction task. Participants explored the system features freely to familiarize themselves with the interface.

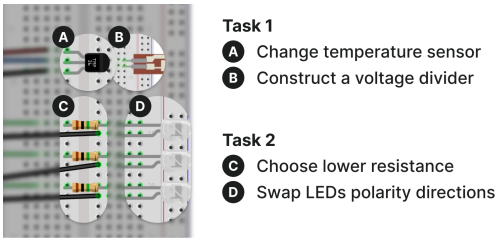

Figure 8: Circuit modification tasks. Task 1 focuses on flex sensor subcircuit: (A) replacing the temperature sensor with a flex sensor, and (B) constructing a voltage divider. Task 2 addresses LED subcircuit troubleshooting: (C) adjusting resistance values, and (D) correcting LED polarity connections.

Circuit Construction Task We adapted Project 3: Love-O-Meter from the Arduino Projects Book [21], selecting it as a beginner-to-intermediate project to simulate the common maker scenario of discovering an online circuit diagram and adapting it for personal use. This workflow typically requires debugging, component substitution, and circuit modification based on available components and project requirements. Participants received a partially completed Love-O-Meter circuit within our schematic design interface and were asked to: (1) replace the original temperature sensor with a flex sensor, and (2) identify and fix bugs within the LED circuit (Figure 8). The task was designed to resemble an ecologically valid scenario with participants leveraging WireWay for circuit modification while working with a schematic diagram that partially matches their needs. Participants could complete these tasks in any order within the allotted timeframe and were not informed about the specific number of bugs to find. We allocated 55 minutes for task completion, providing additional time beyond the original 45-minute recommendation to account for the system learning curve and debugging requirements. To focus the study scope on hardware construction, functional code was provided and accessible throughout the task, and all necessary electronic components were supplied at the beginning of each session.

Post-Task Debrief Following task completion, participants completed post-task measurements including the System Usability Scale (SUS), NASA Task Load Index (TLX), Trust in Automation (TiA) matrix, and seven custom questionnaires evaluating system capabilities (Appendix C). We conducted semi-structured interviews covering task outcomes, prior experience with LLM-assisted hardware prototyping, and system usage observations (Appendix D).

5.3 Data Collection and Analysis

We recorded participant actions through screen capture and external camera footage to document their building process for subsequent analysis. We analyzed quantitative data using descriptive statistics and Pearson correlation analysis to examine relationships among experience measures, usability ratings, trust levels, and task performance. Scale reliability was assessed using Cronbach's alpha, with significance testing at $p < 0.05$ for correlation analysis. Interviews were transcribed, translated to English, and analyzed using a general inductive approach with coding consistency verification

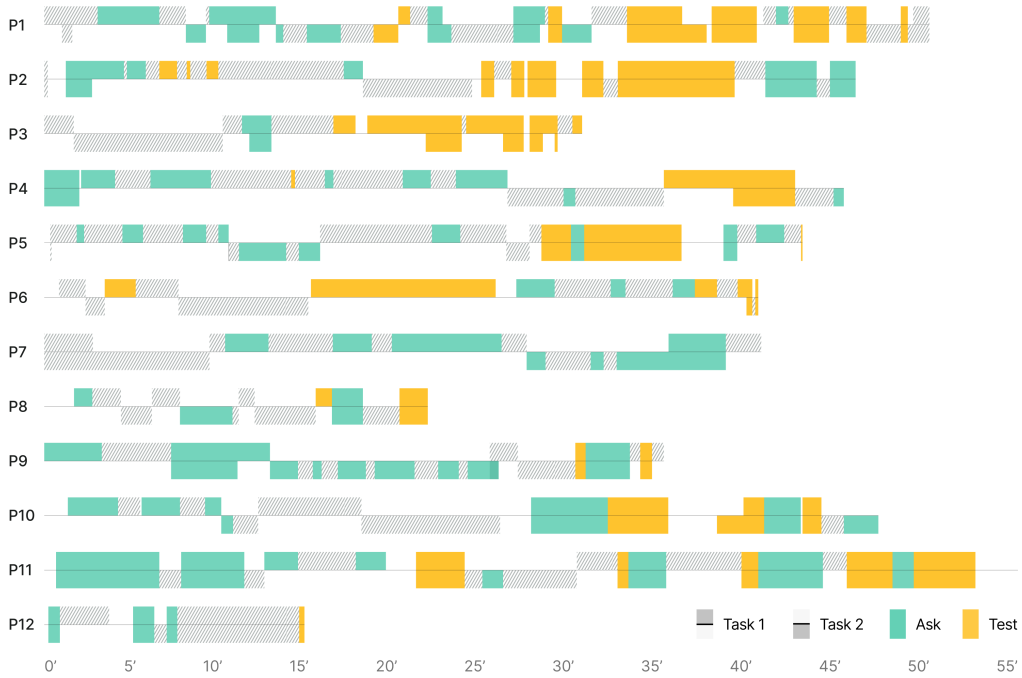

Figure 9: Comprehensive timeline of all participant sessions. Each row represents one complete session: green (*Ask mode* usage), yellow (*Test mode* usage), gray striped Task 1 and Task 2.

between two authors [59]. When quotes received multiple codes, authors discussed until reaching consensus, prioritizing codes based on user intent and system design relevance. Following coding completion, we grouped codes into higher-level themes for presentation alongside quantitative results in the subsequent section.

6 Results and Findings

Nine out of twelve participants successfully completed working circuits (37′34″ ± 12′54″, range 14′42″-50′00″), with a temporal distribution of system feature usage (*Ask mode*, *Test mode*) and task progression across all participants (Figure 9). Three participants failed: P2 and P4 could not design the replacement voltage divider, and P10 used incorrect resistor values due to color band misidentification. The system chat response time averaged 8.91 ± 5.13 seconds, with maximum response time being 36 seconds and minimum being 2 seconds. The overall usability of WireWay was evaluated with an average SUS score of 70.4 ± 15.2 out of 100, indicating above-average usability [5]. The reported cognitive load from NASA TLX averaged 42.8 ± 10 out of 100, representing a moderate workload with balanced mental and physical demands. TiA scores averaged 3.74 ± 1.04 out of 5, indicating moderate to high trust levels. All scales demonstrated acceptable to excellent reliability: SUS (α = .843), TiA (α = .82), and system evaluation questionnaire (α = .707).

6.1 Diverse Circuit Construction Approaches

Participants demonstrated highly individualized circuit construction approaches with substantial variation in entry points, task progression (Figure 10), and circuit design (Figure 11). Behavioral analysis revealed three workflow archetypes: Linear-progressors (P1, P5, P8, P10-P12) maintained longer activity blocks with moderate guidance-seeking; Test-integrated participants (P2, P3, P6) frequently alternated between construction and validation; Conversation heavy participants (P4, P7, P9) relied extensively on *Ask mode* guidance with minimal testing.

Task-Switching Frequency Participants exhibited significant variation in task-switching complexity, with an average of 9.3 ± 5.3 activity transitions per session (range 2-17). Nine of twelve participants began their session using the Ask feature to gather information about Task 1 or the overall study context, while the remaining three immediately started by editing the schematic, indicating different cognitive entry points into the problem space (Figure 9). The system's support for idiosyncratic workflows is clearly demonstrated through the distinct color patterns and directional progressions in the timeline data, ultimately resulting in diverse yet functionally equivalent circuit implementations (Figure 11). The most frequent initial question concerned Task 1 (P3-P10, P12), with a subset of these questions (P7-P8, P10, P12) specifically asking about replacing the temperature sensor with it.

Progression Patterns Experienced participants (4+ years: P1, P8, P11) demonstrated non-sequential task progression, moving freely between circuit sections based on debugging insights, with higher task-switching rates (M=0.26 switches/min) compared to novices with under two years of experience (P2, P4, P7; M=0.17 switches/min). Figure 10 illustrates these patterns across three representative participants. P6 exhibits a test-integrated approach, frequently alternating between construction and testing validation. P8 demonstrates a linear-progressive workflow, systematically

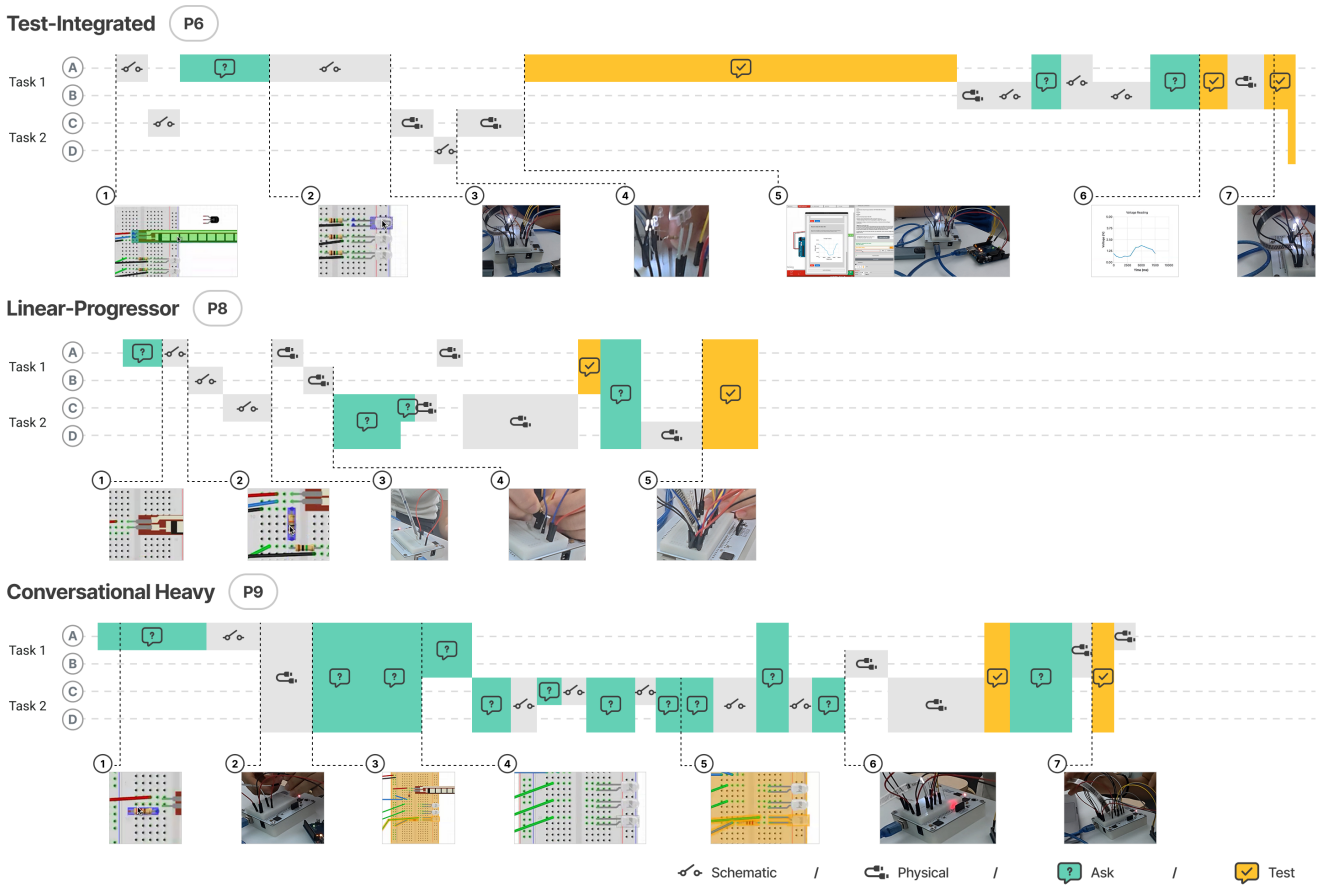

Figure 10: Timeline showing distinct workflow patterns of three representative participants. Each row represents one participant's session, with icons for different activities (schematic editing, physical construction, testing, and asking the system).

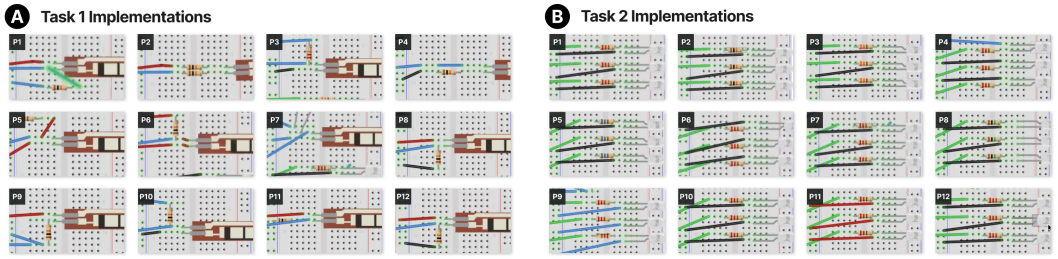

Figure 11: Breadboard circuit implementations by all study participants (N=12), demonstrating individual variation in component placement, wiring strategies, and circuit organization approaches for identical functional requirements. Panel A shows Task 1 implementations, Panel B shows Task 2 implementations.

progressing through tasks with sustained activity blocks and minimal testing. P9 represents a conversational-heavy archetype, with extensive *Ask mode* usage distributed throughout the timeline.

Multilingual Usage Participants demonstrated personalized interaction preferences with both software and hardware components. Some participants (P10, P12) interacted in their native language without instruction, demonstrating the system's natural language processing capabilities. P7 explicitly requested broader language support: *"There's nothing else besides English (that I want to change with the system); I wanted language support other than English."*

Schematic-Hardware Visual Matching Participants showed strong preferences for maintaining visual consistency between schematic representations and physical implementations. P9 appreciated exact connection matching: *"This was very convenient because*

if I clicked [the component in the schematic software], it blinked. So I could check exactly." Some participants even implemented color-coding strategies to maintain visual consistency. When asked about changing wire colors in the software, P11 explained: *"Yeah, so I think it's also related to my working habit. This time I was only able to use the jumper cables from the box, but originally whenever I'm working, I tried to use color coding so that I can select and identify the wire that I'm working on."*

6.2 Hardware-Contextualized Guidance

Hardware-software integration proved valuable for reducing the communication overhead typically required with AI assistants on physical computing tasks. All participants utilized contextual guidance features, sending 166 messages, including 77 with highlighted components for enhanced context.

Contextual Component Reference All participants used pronouns as references when communicating about circuit components, with 70 pronoun instances across 77 highlighted component interactions (Table 3). Most frequent references were *"this"*, *"it"*, and *"here"* with their equivalents, demonstrating reliance on contextual awareness rather than precise component identifier. Communication logs revealed 8 instances of native language contextual references occurring naturally without multilingual instruction. In 100% of component-related communications with component selection, participants used deictic references instead of specific names, indicating significant communication overhead reduction compared to traditional text-based AI assistants. Examples include P12 asking *"I am not sure where to add this"* while highlighting a 220Ω resistor, and P11 inquiring *"Is it properly connected?"* when referencing a highlighted resistor.

Table 3: Contextual pronoun usage in component-highlighted conversations

Pronoun Type	Uses	Percentage	Examples
"this" & equivalents	29	41.4%	"I am not sure where to add this."
"it" & equivalents	19	27.1%	"Is it properly connected?"
"here" & equivalents	7	10.0%	"Can I put the component here?"
"these"	5	7.1%	"How do I connect these?"
"they/them"	4	5.7%	"Where should they go?"
"that"	3	4.3%	"What does that mean?"
"there"	2	2.9%	"Put the wire there."
"now" equivalent	1	1.4%	"What about now?"
Multilingual	8	11.4%	Native-language contextual references.

Eliminating Manual Circuit Explanation The system's circuit state awareness eliminated extensive verbal descriptions typically required for general-purpose AI assistants. P2 highlighted this advantage: *"It is very uncomfortable using ChatGPT [to manually say about the context all the time]. I think that is the advantage of the system that I can just add context by clicking the button, and I don't need to explain details of pins or preferences."* P7 emphasized: *"Using an assistant like this means I don't have to explain to GPT how the circuit is structured, and that's the difference."* Contextual awareness from schematic updates and component highlighting enabled automatic error detection without explicit requests. P8 appreciated, *"I don't remember the color of each resistor, which color means which value, but system automatically found that I used the wrong resistor... that automatic thing was quite convenient for me."*

Direct Hardware Guidance Through LED The component blinking on the breadboard demonstrated high adoption rates and significant correlations with system trust measures. All participants except P12 utilized the Blink feature, generating 297 total highlighting interactions across sessions. The system supported three distinct highlighting modalities: self-initiated position verification ($n = 84$), AI-guided blinking ($n = 11$), and test-integrated highlighting ($n = 32$). Position-based highlighting had the highest adoption, with 11 of 12 participants using it to verify component locations. Usage patterns varied significantly: P2 engaged the highest (21 position clicks, 34 total interactions) while P12 showed no usage. P12 explained: *"Checking whether resistors or LEDs are properly connected in a straight line on a specific row seems more important."*, preferring manual connection checking. P2 contrasted: *"I used to get lost in wiring the board, but the visual assistant is very comfortable."* System-guided highlighting was used by only 3 participants (P1, P2, P5), while test-integrated highlighting was adopted by 7 participants. Strong correlations emerged between highlighting usage and system trust: AI-guided blinking users reported lower system malfunction concerns ($r = -0.71$, $p = .009$), while frequent position verification users demonstrated higher confidence in system capabilities ($r = 0.62$, $p = .030$).

Validation of AI Recommendations LED highlighting spatial guidance received widespread appreciation, with participants finding visual cues more intuitive than text-based instructions alone, supporting both autonomous exploration and system-guided assistance. Participants appreciated contextual awareness but remained appropriately critical of AI suggestions. P3 exemplified this balance: *"I did not trust the chatbot with the actual value of this pressure sensor... And I was kind of right. My gut feeling was right because in the end, we saw that I had to change the resistor a couple of times."* Participants who better understood system reasoning found schematic-board action connections clearer ($r = 0.78$, $p = .003$). Those appreciating the system's complex task handling were less concerned about potential errors ($r = -0.77$, $p = .003$). P3 described: *"The LED highlighting was pretty nice... the most helpful thing... if I just had a schematic, it would be an initially much more overwhelming task to start."*

6.3 Ad Hoc Test Generation for Circuit

The testing capabilities proved highly popular, with 7 out of 12 participants actively using *Test mode*, demonstrating the system's ability to automatically generate appropriate tests based on schematic awareness.

Simplified Testing Setup Participants valued testing without code modifications or complex hardware reconfiguration (P2-P10). P6 explained: *"I also like the test parts, so I don't need to re-upload test codes."* P7 emphasized workflow benefits: *"When testing a set, if you're working alone and just building something, you'd normally have to remove it, make a test setup, try it again, then upload it. But with this, you just plug in one wire, and that's it."* P3 added: *"[I want*

to have more test types] to be platform specific ... for sweeping the voltage for the analog input pin." Contextual awareness of circuit topology enabled automatic test procedure generation. The system eliminated overhead from creating additional testing setups with multimeters and writing dedicated testing code by understanding component testing needs and automatically suggesting relevant measurement approaches.

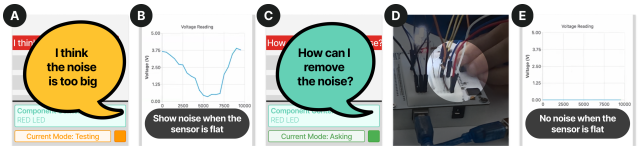

Figure 12: P6's circuit debugging workflow demonstrates: (A) initial problem identification reporting excessive sensor noise, (B) conducting system-suggested diagnostic tests to measure flex sensor noise levels, (C) requesting solution guidance, (D) implementing the recommended circuit modification, and (E) verification tests to confirm noise reduction.

Component-Specific and Whole-Circuit Testing The system's contextual awareness further streamlined testing by understanding circuit context without manual explanation. P6 appreciated not having to explain their setup textually nor visually with pictures: *"I usually use ChatGPT and I ask it to make some skeleton calls when I have some error on the code, I asked AI to fix it. And sometimes when the circuits, I think when the error comes from the circuit, I take pictures and let AI figure out what."* The ability to test individual components proved invaluable for problem isolation, as demonstrated in Figure 12, where P6 systematically identified and resolved noise issues with the flex sensor. P6 expressed a desire for even smarter automation: *"I want to say I want to test this sensor and it can automatically switch to Test mode and test for me."* Strong correlations emerged between testing effectiveness and overall system trust, with participants who found tests helpful for fault isolation showing higher confidence in system capabilities ($r = 0.67$, $p = .017$).

Test Results Analysis Participants who could distinguish reliable from inconclusive test results showed stronger correlations with system-suggested measurement accuracy ($r = 0.79$, $p = .002$), indicating that effective result interpretation was crucial for leveraging the testing capabilities. However, during the testing experience, several participants struggled with interface confusion and inconsistent results. P2 noted: *"I am confused about the ability of Ask mode and testing mode because I didn't spot the major difference."* P6 experienced frustration when testing failed: *"I tried adding the pull-down resistor, but I failed actually because ChatGPT recommended me the wrong resistor value. And I tried to use the serial monitor to see the noise for the flex sensor, but it was not a very good experience."* This confusion led some participants (P8, P11) to bypass *Test mode* entirely, preferring the serial monitor instead. Despite these interface challenges, the underlying functionality successfully supported debugging workflows when participants understood how to interpret the results. Additional correlations demonstrated that participants who found measurement suggestions accurate were more confident in system capabilities ($r = 0.64$, $p = .024$), and those who understood test outcomes showed higher system reliability ratings ($r = 0.66$, $p = .020$).

6.4 System Reliability and Usage Context Feedback

System Trust and Acceptance Strong correlations emerged between system acceptance and trust mechanisms, revealing specific relationships relevant to system design. Participants' intention for frequent use was strongly associated with system reliability ($r = 0.88$, $p < .001$), while those who found system functions well-integrated showed reduced unpredictability concerns ($r = -0.83$, $p < .001$). Prior AI experience showed no relationship with task performance ($r = 0.00$, $p = .994$), while physical computing experience did not significantly predict system evaluation ratings ($r = 0.25$, $p = .440$), indicating WireWay successfully accommodated users regardless of technical background.

Educational and Professional Applications Interview analysis revealed distinct usage patterns across educational and professional contexts. Most participants identified educational potential with conditions. P3 explained: *"I would really see this as at a pre-university electronics courses, such as in science high school or Robotics classes... especially teenagers would love this idea."* However, P9 countered: *"I would not recommend using this project if you're a kid and you want to learn hardware, because... they tell us everything, and you just copy what it is. So at the end of the day, you end up learning nothing."* This highlights the importance of balancing assistance with learning opportunities. Beyond education, participants identified value for non-engineers in creative projects. P3 and P11 noted applications for interaction artists, with P11 mentioning: *"when they don't know about the detailed technical aspect. They will find it really helpful."* The system's value extended to personal project development. P6 exemplified, explaining how the system solved her flex sensor project: *"I am using this flex sensor for my personal project now, and this sensor was a little tricky because I cannot see what's happening and why the noise is huge. And I tried the same method, adding the pulldown resistor, but I failed... ChatGPT recommended me the wrong resistor value."* Through guided testing, she learned that the flex sensor required a proper voltage divider configuration. P6 appreciated reduced trial-and-error: *"was really helpful for educational things because before I went through many trials and errors, I would make so many mistakes, like name changes, wrong resistors, or broken circuits. So it's beneficial because it firstly makes the circuits in the software and then goes to here [breadboard], and I really like the blinking for where you should put the wire, because sometimes I think that I did it correctly, but actually I did it wrong."* This demonstrates how WireWay bridged formal learning in practical applications.

Circuit Complexity Considerations Participants had mixed perspectives on complex circuit usage. Multiple participants indicated they would not use the system for highly complex circuits. P9 emphasized utility for smaller components: *"if it's a small part of the task of the bigger picture, I think it's fine."* also P10 explained, *"It'd not be useful for people who've already designed so much [because then they need to add all of their circuit design in software], would be better just to casually make from the beginning."* However, some participants suggested usefulness for complex circuits, noting potential applications in custom PCB design. P8 mentioned: *"I can*

imagine also having a custom PCB board to use with this system, as I can just upload that PCB file to use the system." This suggests diverse application opportunities while highlighting the need for improved interface clarity and assistance-learning balance.

7 Discussion

The formative study highlighted three key challenges among makers for physical computing prototyping, leading to our three design goals: DG1) personalized guidance adapting to individual circuit contexts; DG2) hardware-contextualized visual guidance bridging schematics with physical construction; and DG3) context-aware testing generating validation procedures for specific configurations. All participants adopted these core capabilities, revealing distinct workflow archetypes through personalized approaches, widespread use of pronoun-based contextual communication over technical terminology, and effective hardware-aware debugging. Participants valued WireWay's usefulness while noting mode confusion and concerns about balancing assistance with learning opportunities in educational contexts.

7.1 Physical Circuits as Spatial Language

Our approach presents hardware-software interaction as a communication problem, building on Miyake [50]'s work explaining that conceptual viewpoints are reflected in *spatial language* during collaborative understanding of complex physical devices. The distinctive contribution lies in combining real-time schematic parsing with conversational interaction to support makers' idiosyncratic workflows [10] from arbitrary starting points. This addresses the formative study's first challenge of diverse construction strategies, as evidenced by the distinct workflow archetypes (test-integrated, linear-progressive, conversational-heavy) that emerged during evaluation. Addressing the second challenge of schematic-physical mismatch, WireWay maintains awareness of exact circuit topology rather than requiring users to manually bridge between representations while enabling natural spatial communication patterns. This dual capability of understanding both the precise technical state and the conversational context enables support for individual preferences and non-linear exploration that characterizes authentic maker practices [48]. The key insight is that when systems understand circuit context, users intuitively abandon technical identifiers in favor of deictic references, mirroring a natural communication strategy of humans in physical spaces.

Previous hardware guidance approaches span instruction delivery systems [8, 32, 53], conversational interfaces for step-by-step guidance [12, 30], structured tutorial frameworks [56, 63], and example-based exploration tools [49]. Our communication paradigm builds on foundational work in constructive interaction [50] and conversation with materials [60], yet differs fundamentally by combining real-time schematic parsing with contextual communication. While existing tools assume predetermined workflows, WireWay combines technical analysis and natural dialogue from arbitrary circuit origins. This distinction allows complementary use: providing contextual guidance wherever users begin without disrupting exploration patterns, alongside structured instruction systems.

Our work suggests a fundamental design principle: automated context understanding enables personalized communication. Systems that parse domain-specific state can support natural human communication patterns rather than forcing conformity to predetermined workflows. For HCI, this implies combining deep technical awareness with adaptive interaction paradigms that accommodate diverse user preferences, from multimodal interfaces supporting different sensory modalities [52] to tutorial mediums that affect physical skill transfer [20]. By supporting inquiries about arbitrary component models rather than requiring predetermined sets, WireWay enables makers to utilize existing materials, potentially supporting sustainable making practices through component reuse and reduced e-waste in prototyping activities [45, 68]. These findings extend beyond electronics to broader physical manipulation concerns [55], contributing to physical computing's foundational goals [54] through intelligent mediation rather than rigid instruction.

7.2 Debugging Circuits as an Inquiry

Addressing the third challenge of manual and iterative debugging, we reframed circuit debugging from a diagnostic skill to a collaborative inquiry process, building upon Interrogative Debugging [36], where programmers ask "why did" and "why didn't" questions about program failures. Instead of requiring users to hypothesize error sources and manually construct test procedures [16], our approach treats the circuit itself as an active participant in the debugging conversation. This shifts the cognitive burden from error localization to result interpretation, democratizing debugging capabilities while preserving learning opportunities [25].

Physical computing debugging approaches typically require significant technical expertise [29, 46], explicit hypothesis formulation [1], predefined testing frameworks [63], or manual interpretation of visualization data [66, 67]. Software-focused advances [9, 47] and real-time manipulation tools [58] with structured logging approaches [28] operate independently from circuit construction contexts. Our collaborative inquiry approach, inspired by foundational "why" questioning work [36, 37], transforms debugging from diagnostic skill to guided exploration. Educational research [26] emphasizes diverse debugging pathways, aligning with our democratization goals. WireWay complements existing tools by providing circuit-aware test generation that adapts to any hardware configuration, reducing expertise barriers while preserving learning value and maintaining appropriate trust in automated systems [38].

Our findings reveal that effective debugging tools should function as mediators of inquiry rather than diagnostic instruments, suggesting a design philosophy where tools scaffold exploration rather than prescribe solutions. This points toward debugging interfaces that emphasize collaborative sense-making between human intuition and computational analysis, with broader implications for any domain requiring iterative refinement of complex systems. However, three critical obstacles emerge for LLM-based assistance in maker contexts. First, over-reliance risks may undermine the productive struggle essential for developing debugging expertise [25, 51], a concern increasingly recognized in educational AI systems [14]. Next, while LLMs help bridge vocabulary gaps, the system may respond with technical terminology (e.g., cathode) that

novices may struggle to translate without visual or contextual support. Lastly, this challenge could be compounded by cultural and dialectal biases in language models [64] that may exclude non-English speaking makers. Future tools could balance automation with agency through adaptive support that fades as competence develops [56], while addressing language barriers to ensure users remain active participants rather than passive observers in troubleshooting processes.

7.3 Limitations and Future Works

We acknowledge that WireWay's current implementation presents software and hardware limitations that offer opportunities for enhancement.

Software Design: While the *Ask/Test mode* separation functions as a fail-safe, this distinction created confusion among participants, suggesting mode selection based on user query and conversational context. Enhanced AI prompt pipelines could infer user intent by simultaneously processing natural language input, schematic state, breadboard connections [e.g., 67], and code environments. Added to current pin highlighting for voltage outputs, future systems could introduce explicit feedback mechanisms: mode clarification through confirmation dialogs when users switch modes (e.g., "Proceed for testing?") to help users understand system state, and safety warnings before voltage outputs (e.g., "Outputting 5V at A0. Proceed?") to prevent component damage and encourage safety practices [39].

Hardware State: Our system cannot physically sense component presence, identity, or connection topology unlike CircuitSense [67] or CurrentViz [66]. Users still need to verify the synchronization manually to ensure the schematics match those on the breadboard, compounded by LED visibility issues, as participants reported eye strain and difficulty distinguishing specific breadboard rows, particularly under varying lighting conditions. Future works could explore automated circuit sensing or alternative indication methods such as directional lighting, improved LED diffusion, or augmented reality overlays [22, 31] that could both guide component placement and assist users in verifying circuit state. The pins can output basic electronic signals (e.g., analog voltage, PWM), and read analog voltage. Subsequent iterations can integrate more advanced input/output protocols, like UART, I2C, or SPI, to assist testing digital components. The 25-row breadboard constrains the complexity of circuits. Future hardware iterations could house a larger breadboard or enable multi-board configurations via communication ports. Software could also support virtual component placement beyond the physical board [31].

Evaluation Methodology: Our usability study introduces three constraints that limit generalizability and comparative assessment with other systems. Firstly, the study involved simple circuits with moderate component density, limiting transferability to more complex configurations. Future studies should investigate more complex circuit configurations in longitudinal settings to examine skill transfer and retention over extended periods. Second, the absence of baseline comparisons prevents direct quantification of WireWay's advantages over traditional or existing tools like [12, 30]. Additionally, the formative study restricted participants from using LLMs beyond Google's default AI summaries to establish baseline challenges, while WireWay's core support derives from LLMs. Further studies should incorporate controlled comparisons with conventional approaches, established tutorial systems, general-purpose AI assistants, and separate tool combinations (e.g., LLMs with standalone schematic design software) to validate the integration value. Third, participant recruitment focused primarily on university students with moderate physical computing experience, limiting insights about effectiveness across diverse skill levels and cultural contexts. Expanding evaluation to include professional makers, educators, and participants from varied educational backgrounds would strengthen claims about universal applicability.

8 Conclusion

We presented WireWay, an integrated development environment that bridges circuit design software and physical construction, through hardware-contextualized guidance and in-situ testing capabilities. It allows makers to interact with an AI that maintains real-time awareness of both visual circuit schematics and physical configurations, supported by LED-based spatial guidance on an augmented breadboard and automatically generated context-aware tests. The system accommodates personalized building workflows by enabling natural contextual references to circuit components and eliminating the need for separate testing setups. The system was designed based on formative studies with makers and was evaluated with 12 participants across circuit construction tasks. Results show that participants successfully adopted all main features, with three distinct workflow archetypes emerging while maintaining high completion rates and universal accessibility across technical backgrounds. Future work will explore automatic mode change through enhanced prompt pipelines, hardware improvements such as circuit sensing and augmented reality overlays, and longitudinal evaluations with diverse maker populations.

Acknowledgments

This work was supported by the IITP(Institute of Information & Communications Technology Planning & Evaluation)-ITRC(Information Technology Research Center) grant funded by the Korea government(Ministry of Science and ICT)(IITP-2026-RS-2024-00436398).

References

[1] Abdulaziz Alaboudi and Thomas D. Latoza. 2023. Hypothesizer: A Hypothesis-Based Debugger to Find and Test Debugging Hypotheses. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 73, 14 pages. doi:10.1145/3586183.3606781

[2] Andrea Alessandrini. 2022. A Study of Students Engaged in Electronic Circuit Wiring in an Undergraduate Course. *Journal of Science Education and Technology* 32 (10 2022), 1–18. doi:10.1007/s10956-022-09994-9

[3] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-Action-Circuits: Leveraging Generative Design to Enable Novices to Design and Build Circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 331–342. doi:10.1145/3126594.3126637

[4] Yasharth Bajpai, Bhavya Chopra, Param Biyani, Cagri Aslan, Dustin Coleman, Sumit Gulwani, Chris Parnin, Arjun Radhakrishna, and Gustavo Soares. 2024. Let's Fix this Together: Conversational Debugging with GitHub Copilot. In *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–12. doi:10.1109/VL/HCC60511.2024.00011

[5] Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual SUS scores mean: adding an adjective rating scale. *J. Usability Studies* 4, 3 (May 2009), 114–123.

[6] Imam Nur Bani Yusuf, Diyanah Binte Abdul Jamal, and Lingxiao Jiang. 2023. Automating Arduino Programming: From Hardware Setups to Sample Source

Code Generation. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 453–464. doi:10.1109/MSR59073.2023.00069

[7] Andrea Bianchi, Steve Hodges, David J. Cuartielles, Hyunjoo Oh, Mannu Lambrichts, and Anne Roudaut. 2023. Beyond prototyping boards: future paradigms for electronics toolkits. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, Article 333, 6 pages. doi:10.1145/3544549.3573792

[8] Andrea Bianchi, Kongpyung (Justin) Moon, Artem Dementyev, and Seungwoo Je. 2024. BlinkBoard: Guiding and monitoring circuit assembly for synchronous and remote physical computing education. *HardwareX* 17 (2024), e00511. doi:10.1016/j.ohx.2024.e00511

[9] Andrea Bianchi, Zhi Lin Yap, Punn Lertjaturaphat, Austin Z. Henley, Kongpyung Justin Moon, and Yoonji Kim. 2024. Inline Visualization and Manipulation of Real-Time Hardware Log for Supporting Debugging of Embedded Programs. *Proc. ACM Hum.-Comput. Interact.* 8, EICS, Article 248 (June 2024), 26 pages. doi:10.1145/3660250

[10] Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. 2016. 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* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3485–3497. doi:10.1145/2858036.2858533

[11] Anke Brocker, René Schäfer, Christian Remy, Simon Voelker, and Jan Borchers. 2023. Flowboard: How Seamless, Live, Flow-Based Programming Impacts Learning to Code for Embedded Electronics. *ACM Trans. Comput.-Hum. Interact.* 30, 1, Article 2 (March 2023), 36 pages. doi:10.1145/3533015

[12] Taizhou Chen, Lantian Xu, and Kening Zhu. 2021. FritzBot: A data-driven conversational agent for physical-computing system design. *International Journal of Human-Computer Studies* 155 (2021), 102699. doi:10.1016/j.ijhcs.2021.102699

[13] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 93–102. doi:10.1145/2380116.2380130

[14] Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jingheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S. Yu, and Qingsong Wen. 2025. LLM Agents for Education: Advances and Applications. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 13782–13810. doi:10.18653/v1/2025.findings-emnlp.743

[15] Josh Urban Davis, Jun Gong, Yunxin Sun, Parmit Chilana, and Xing-Dong Yang. 2019. CircuitStyle: A System for Peripherally Reinforcing Best Practices in Hardware Computing. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 109–120. doi:10.1145/3332165.3347920

[16] David DeLiema, Jeffrey K. Bye, and Vijay Marupudi. 2024. Debugging Pathways: Open-Ended Discrepancy Noticing, Causal Reasoning, and Intervening. *ACM Trans. Comput. Educ.* 24, 2, Article 28 (May 2024), 34 pages. doi:10.1145/3650115

[17] Kayla DesPortes, Aditya Anupam, Neeti Pathak, and Betsy DiSalvo. 2016. BitBlox: A Redesign of the Breadboard. In *Proceedings of the The 15th International Conference on Interaction Design and Children* (Manchester, United Kingdom) *(IDC '16)*. Association for Computing Machinery, New York, NY, USA, 255–261. doi:10.1145/2930674.2930708

[18] Kayla DesPortes and Betsy DiSalvo. 2019. Trials and Tribulations of Novices Working with the Arduino. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 219–227. doi:10.1145/3291279.3339427

[19] Daniel Drew, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. 2016. The Toastboard: Ubiquitous Instrumentation and Automated Checking of Breadboarded Circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 677–686. doi:10.1145/2984511.2984566

[20] Shreyosi Endow and Cesar Torres. 2021. "I'm Better Off on my Own": Understanding How a Tutorial's Medium Affects Physical Skill Development. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference* (Virtual Event, USA) *(DIS '21)*. Association for Computing Machinery, New York, NY, USA, 1313–1323. doi:10.1145/3461778.3462066

[21] S. Fitzgerald, M. Shiloh, and T. Igoe. 2012. *Arduino Projects Book*. Arduino. https://books.google.co.kr/books?id=4Bxz0AEACAAJ

[22] Seung Hyeon Han, Yeeun Han, Kyeongho Park, Sangjun Lee, and Woohun Lee. 2025. SpatIO: Spatial Physical Computing Toolkit Based on Extended Reality. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 978, 22 pages. doi:10.1145/3706598.3713747

[23] Kobi Hartley, Joe Finney, Steve Hodges, Peli De Halleux, James Devine, and Gabriele D'Amone. 2023. MakeDevice: Evolving Devices Beyond the Prototype with Jacdac. In *Proceedings of the Seventeenth International Conference on Tangible, Embedded, and Embodied Interaction* (Warsaw, Poland) *(TEI '23)*. Association for Computing Machinery, New York, NY, USA, Article 37, 7 pages. doi:10.1145/3569009.3573106

[24] Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (Montreux, Switzerland) *(UIST '06)*. Association for Computing Machinery, New York, NY, USA, 299–308. doi:10.1145/1166253.1166300

[25] Colin Hennessy Elliott, Alexandra Gendreau Chakarov, Jeffrey B. Bush, Jessie Nixon, and Mimi Recker. 2023. Toward a debugging pedagogy: helping students learn to get unstuck with physical computing systems. *Information and Learning Sciences* 124, 1-2 (01 2023), 1–24. arXiv:https://www.emerald.com/ils/article-pdf/124/1-2/1/1147515/ils-03-2022-0051.pdf doi:10.1108/ILS-03-2022-0051

[26] Colin Hennessy Elliott, Jessie Nixon, Alexandra Gendrau Chakarov, Jeffrey B. Bush, Michael J. Schneider, and Mimi Recker. 2024. Characterizing Teacher Support of Debugging with Physical Computing: Debugging Pedagogies in Practice. *ACM Trans. Comput. Educ.* 24, 4, Article 48 (Dec. 2024), 28 pages. doi:10.1145/3677612

[27] Steve Hodges, Sue Sentance, Joe Finney, and Thomas Ball. 2020. Physical Computing: A Key Element of Modern Computer Science Education. *Computer* 53, 4 (April 2020), 20–30. doi:10.1109/MC.2019.2935058

[28] Peiling Jiang, Fuling Sun, and Haijun Xia. 2023. Log-it: Supporting Programming with Interactive, Contextual, Structured, and Visual Logs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 594, 16 pages. doi:10.1145/3544548.3581403

[29] Mitchell Karchemsky, J.D. Zamfirescu-Pereira, Kuan-Ju Wu, François Guimbretière, and Bjoern Hartmann. 2019. Heimdall: A Remotely Controlled Inspection Workbench For Debugging Microcontroller Projects. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3290605.3300728

[30] Yoonji Kim, Youngkyung Choi, Daye Kang, Minkyeong Lee, Tek-Jin Nam, and Andrea Bianchi. 2020. HeyTeddy: Conversational Test-Driven Development for Physical Computing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 4, Article 139 (Sept. 2020), 21 pages. doi:10.1145/3369838

[31] Yoonji Kim, Youngkyung Choi, Hyein Lee, Geehyuk Lee, and Andrea Bianchi. 2019. VirtualComponent: A Mixed-Reality Tool for Designing and Tuning Breadboarded Circuits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3290605.3300407

[32] Yoonji Kim, Hyein Lee, Ramkrishna Prasad, Seungwoo Je, Youngkyung Choi, Daniel Ashbrook, Ian Oakley, and Andrea Bianchi. 2020. SchemaBoard: Supporting Correct Assembly of Schematic Circuits using Dynamic In-Situ Visualization. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 987–998. doi:10.1145/3379337.3415887

[33] Yoonji Kim, Junyi Zhu, Mihir Trivedi, Dishita Turakhia, Ngai Hang Wu, Donghyeon Ko, Michael Wessely, and Stefanie Mueller. 2022. SensorViz: Visualizing Sensor Data Across Different Stages of Prototyping Interactive Objects. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference* (Virtual Event, Australia) *(DIS '22)*. Association for Computing Machinery, New York, NY, USA, 987–1001. doi:10.1145/3532106.3533481

[34] M. King and D. Winn. 2017. Chapter 8 - Cross Institutional Peer Coaching: A Case Study. In *Beyond Mentoring*, Dawn Lowe-Wincentsen (Ed.). Chandos Publishing, 93–106. doi:10.1016/B978-0-08-101294-9.00008-8

[35] André Knörig, Reto Wettach, and Jonathan Cohen. 2009. Fritzing: a tool for advancing electronic prototyping for designers. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (Cambridge, United Kingdom) *(TEI '09)*. Association for Computing Machinery, New York, NY, USA, 351–358. doi:10.1145/1517664.1517735

[36] Amy J. Ko and Brad A. Myers. 2004. Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 151–158. doi:10.1145/985692.985712

[37] Amy J. Ko and Brad A. Myers. 2008. Debugging reinvented: asking and answering why and why not questions about program behavior. In *Proceedings of the 30th International Conference on Software Engineering* (Leipzig, Germany) *(ICSE '08)*. Association for Computing Machinery, New York, NY, USA, 301–310. doi:10.1145/1368088.1368130

[38] Moritz Körber. 2019. Theoretical Considerations and Development of a Questionnaire to Measure Trust in Automation. In *Proceedings of the 20th Congress of the*

International Ergonomics Association (IEA 2018), Sebastiano Bagnara, Riccardo Tartaglia, Sara Albolino, Thomas Alexander, and Yushi Fujita (Eds.). Springer International Publishing, Cham, 13–30.

[39] Marcel Lahaye, Vivian Isabel Reinartz, Sarah Sahabi, and Jan Borchers. 2023. Towards Authoring Tools For DIY Tutorials: From Tutorial User Strategies to Guidelines (Free Template Included!). In *Proceedings of Mensch Und Computer 2023* (Rapperswil, Switzerland) *(MuC '23)*. Association for Computing Machinery, New York, NY, USA, 380–386. doi:10.1145/3603555.3608530

[40] Woojin Lee, Ramkrishna Prasad, Seungwoo Je, Yoonji Kim, Ian Oakley, Daniel Ashbrook, and Andrea Bianchi. 2021. VirtualWire: Supporting Rapid Prototyping with Instant Reconfigurations of Wires in Breadboarded Circuits. In *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Salzburg, Austria) *(TEI '21)*. Association for Computing Machinery, New York, NY, USA, Article 4, 12 pages. doi:10.1145/3430524.3440623

[41] Richard Lin, Rohit Ramesh, Connie Chi, Nikhil Jain, Ryan Nuqui, Prabal Dutta, and Björn Hartmann. 2020. Polymorphic Blocks: Unifying High-level Specification and Low-level Control for Circuit Board Design. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 529–540. doi:10.1145/3379337.3415860

[42] Richard Lin, Rohit Ramesh, Antonio Iannopollo, Alberto Sangiovanni Vincentelli, Prabal Dutta, Elad Alon, and Björn Hartmann. 2019. Beyond Schematic Capture: Meaningful Abstractions for Better Electronics Design Tools. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3290605.3300513

[43] Richard Lin, Rohit Ramesh, Nikhil Jain, Josephine Koe, Ryan Nuqui, Prabal Dutta, and Bjoern Hartmann. 2021. Weaving Schematics and Code: Interactive Visual Editing for Hardware Description Languages. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 1039–1049. doi:10.1145/3472749.3474804

[44] Jo-Yu Lo, Da-Yuan Huang, Tzu-Sheng Kuo, Chen-Kuo Sun, Jun Gong, Teddy Seyed, Xing-Dong Yang, and Bing-Yu Chen. 2019. AutoFritz: Autocomplete for Prototyping Virtual Breadboard Circuits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3290605.3300633

[45] Jasmine Lu, Sai Rishitha Boddu, and Pedro Lopes. 2025. ProtoPCB: Reclaiming Printed Circuit Board E-waste as Prototyping Material. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 888, 12 pages. doi:10.1145/3706598.3714095

[46] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, and Björn Hartmann. 2017. Bifröst: Visualizing and Checking Behavior of Embedded Systems across Hardware and Software. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 299–310. doi:10.1145/3126594.3126658

[47] William McGrath, Jeremy Warner, Mitchell Karchemsky, Andrew Head, Daniel Drew, and Bjoern Hartmann. 2018. WiFröst: Bridging the Information Gap for Debugging of Networked Embedded Systems. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 447–455. doi:10.1145/3242587.3242668

[48] Gözde McLaughlin and Amy Farris. 2025. A Systems Thinking Perspective on Building and Debugging Physical Computing Projects. *Technology, Knowledge and Learning* (05 2025), 1–25. doi:10.1007/s10758-025-09855-5

[49] Paul Methfessel, Tom Beckmann, Patrick Rein, Stefan Ramson, and Robert Hirschfeld. 2024. MµSE: Supporting Exploration of Software-Hardware Interactions Through Examples. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 936, 16 pages. doi:10.1145/3613904.3642186

[50] Naomi Miyake. 1986. Constructive interaction and the iterative process of understanding. *Cognitive Science* 10, 2 (1986), 151–177. doi:10.1016/S0364-0213(86)80002-7

[51] Luis Morales-Navarro, Deborah A Fields, and Yasmin B Kafai. 2021. Growing Mindsets: Debugging by Design to Promote Students' Growth Mindset Practices in Computer Science Class. *Proceedings of the 15th International Conference of the Learning Sciences - ICLS 2021* (2021). https://par.nsf.gov/biblio/10309425

[52] Aya Mouallem, Mirelys Mendez Pons, Ali Malik, Trini Rogando, Gene S-H Kim, Trisha Kulkarni, Charlene Chong, Danyang Fan, Shloke Nirav Patel, Lauren Aquino Shluzas, Helen L. Chen, and Sheri D. Sheppard. 2025. IncluSim: An Accessible Educational Electronic Circuit Simulator for Blind and Low-Vision Learners. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 338, 18 pages. doi:10.1145/3706598.3713437

[53] Yoichi Ochiai. 2014. Visible Breadboard: System for Dynamic, Programmable, and Tangible Circuit Prototyping with Visible Electricity. In *Proceedings of the 6th International Conference on Virtual, Augmented and Mixed Reality. Applications of Virtual and Augmented Reality - Volume 8526*. Springer-Verlag, Berlin, Heidelberg, 73–84. doi:10.1007/978-3-319-07464-1_7

[54] Dan O'Sullivan and Tom Igoe. 2004. *Physical computing: sensing and controlling the physical world with computers*. Course Technology Press, Boston, MA, USA.

[55] Raf Ramakers, Danny Leen, Jeeeun Kim, Kris Luyten, Steven Houben, and Tom Veuskens. 2023. Measurement Patterns: User-Oriented Strategies for Dealing with Measurements and Dimensions in Making Processes. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 214, 17 pages. doi:10.1145/3544548.3581157

[56] Nico Ritschel, Anand Ashok Sawant, David Weintrop, Reid Holmes, Alberto Bacchelli, Ronald Garcia, Chandrika K R, Avijit Mandal, Patrick Francis, and David C. Shepherd. 2023. Training industrial end-user programmers with interactive tutorials. *Software: Practice and Experience* 53, 3 (2023), 729–747. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3167 doi:10.1002/spe.3167

[57] Donald A. Schön. 1983. *The Reflective Practitioner: How Professionals Think in Action*. Ashgate. https://books.google.co.kr/books?id=E85qAAAAMAAJ

[58] Evan Strasnick, Maneesh Agrawala, and Sean Follmer. 2017. Scanalog: Interactive Design and Debugging of Analog Circuits with Programmable Hardware. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 321–330. doi:10.1145/3126594.3126618

[59] David R. Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation* 27, 2 (2006), 237–246. arXiv:https://doi.org/10.1177/1098214005283748 doi:10.1177/1098214005283748

[60] Cesar Torres, Molly Jane Nicholas, Sangyeon Lee, and Eric Paulos. 2019. A Conversation with Actuators: An Exploratory Design Environment for Hybrid Materials. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Tempe, Arizona, USA) *(TEI '19)*. Association for Computing Machinery, New York, NY, USA, 657–667. doi:10.1145/3294109.3295643

[61] Maike Vollstedt and Sebastian Rezat. 2019. *An Introduction to Grounded Theory with a Special Focus on Axial Coding and the Coding Paradigm*. Springer International Publishing, 81–100. doi:10.1007/978-3-030-15636-7_4

[62] Chiuan Wang, Hsuan-Ming Yeh, Bryan Wang, Te-Yen Wu, Hsin-Ruey Tsai, Rong-Hao Liang, Yi-Ping Hung, and Mike Y. Chen. 2016. CircuitStack: Supporting Rapid Prototyping and Evolution of Electronic Circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 687–695. doi:10.1145/2984511.2984527

[63] Jeremy Warner, Ben Lafreniere, George Fitzmaurice, and Tovi Grossman. 2018. ElectroTutor: Test-Driven Physical Computing Tutorials. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 435–446. doi:10.1145/3242587.3242591

[64] Azmine Toushik Wasi, Raima Islam, Mst Rafia Islam, Farig Sadeque, Taki Hasan Rafi, and Dong-Kyu Chae. 2025. Dialectal Bias in Bengali: An Evaluation of Multilingual Large Language Models Across Cultural Variations. In *Companion Proceedings of the ACM on Web Conference 2025* (Sydney NSW, Australia) *(WWW '25)*. Association for Computing Machinery, New York, NY, USA, 1380–1384. doi:10.1145/3701716.3715468

[65] Te-Yen Wu, Jun Gong, Teddy Seyed, and Xing-Dong Yang. 2019. Proxino: Enabling Prototyping of Virtual Circuits with Physical Proxies. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 121–132. doi:10.1145/3332165.3347938

[66] Te-Yen Wu, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, Jun-You Liu, Yu-Chih Lin, and Mike Y. Chen. 2017. CurrentViz: Sensing and Visualizing Electric Current Flows of Breadboarded Circuits. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 343–349. doi:10.1145/3126594.3126646

[67] Te-Yen Wu, Bryan Wang, Jiun-Yu Lee, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, Yu-Chih Lin, and Mike Y. Chen. 2017. CircuitSense: Automatic Sensing of Physical Circuits and Generation of Virtual Circuits to Support Software Tools.. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 311–319. doi:10.1145/3126594.3126634

[68] Zeyu Yan, Mrunal Sanjay Dhaygude, and Huaishu Peng. 2025. Make Making Sustainable: Exploring Sustainability Practices, Challenges, and Opportunities in Making Activities. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 886, 14 pages. doi:10.1145/3706598.3713665

[69] Saelyne Yang, Sangkyung Kwak, Juhoon Lee, and Juho Kim. 2023. Beyond Instructions: A Taxonomy of Information Types in How-to Videos. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 797, 21 pages. doi:10.1145/3544548.3581126

A Formative Study: Interview Protocol

We include here the full set of interview questions used in the formative study.

(1) How did you decide on the keywords or sites you used?
(2) Were there any search results you ignored right away? Why?
(3) Which media format of information did you go to first, and why?
(4) Were there moments you switched formats? What triggered the switch?
(5) How did you judge the trustworthiness or accuracy of the online tutorials? Can you give an example?
(6) Did you find any format misleading or confusing? What made it so?
(7) How did you keep track of different wiring diagrams or code fragments you encountered?
(8) Did you feel like you were jumping between multiple tutorials? Tell me about that experience.
(9) When you found a snippet that looked useful, how did you test or verify it before applying it?
(10) Tell me about one mistake you made (wiring vs. code) and how you diagnosed or fixed it.
(11) How would you redesign the tutorial to avoid the mistake you explained?
(12) Would you wish there were any hints or checkpoints that could have helped?
(13) At any point, did you wish all the information were in one place? What would that look like?
(14) Anything else you wish we had asked or that you'd like to share?

B System Interface

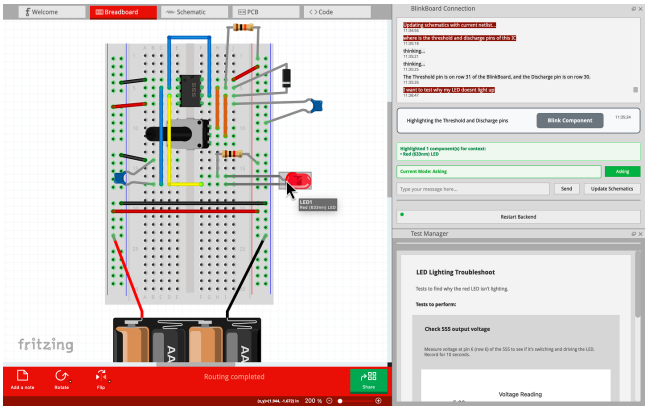

Figure 13: Unmodified user interface of WireWay

C User Study: Evaluation Questionnaires

Participants rated the following statements on a 5-point Likert scale. (*1 = Strongly Disagree, 5 = Strongly Agree*).

(1) The tests helped me isolate faults quickly. (1–5)
(2) Measurement suggestions matched what mattered on the board. (1–5)
(3) Test outcomes mapped clearly to what I should do next. (1–5)
(4) I could tell when the test was wrong or inconclusive. (1–5)
(5) I understood why the system suggested each step. (1–5)
(6) When results changed, I knew what caused the change. (1–5)
(7) The link between schematic and board actions was clear. (1–5)

D User Study: Interview Protocol

We include here the full set of interview questions used in the usability study.

D.1 Experience with AI for Physical Computing

(1) Do you have experience in using any AI for physical computing? What was it like?
(2) What would be different if you were using that AI compared to our system?

D.2 System Usage and Information Seeking

(3) Were there any moments you wanted to search online for anything else?
(4) What was the most helpful aspect of the system?
(5) What was the most frustrating aspect of the system?

D.3 System Improvement and Future Use

(6) If you could change one thing about the system, what would it be?
(7) Can you imagine using this system in any context? What projects would you use it for, and what projects would you not use it for?

D.4 Task-Specific Clarifications

Additional clarification questions were asked based on observed participant behaviors and task-specific challenges encountered during the session.