CircuitGlue: A Software Configurable Converter for Interconnecting Multiple Heterogeneous Electronic Components

MANNU LAMBRICHTS, Hasselt University - Flanders Make - Expertise Centre for Digital Media, Belgium
RAF RAMAKERS, Hasselt University - Flanders Make - Expertise Centre for Digital Media, Belgium
STEVE HODGES, Microsoft Research, United Kingdom
JAMES DEVINE, Microsoft Research, United Kingdom
LORRAINE UNDERWOOD, Lancaster University, United Kingdom
JOE FINNEY, Lancaster University, United Kingdom

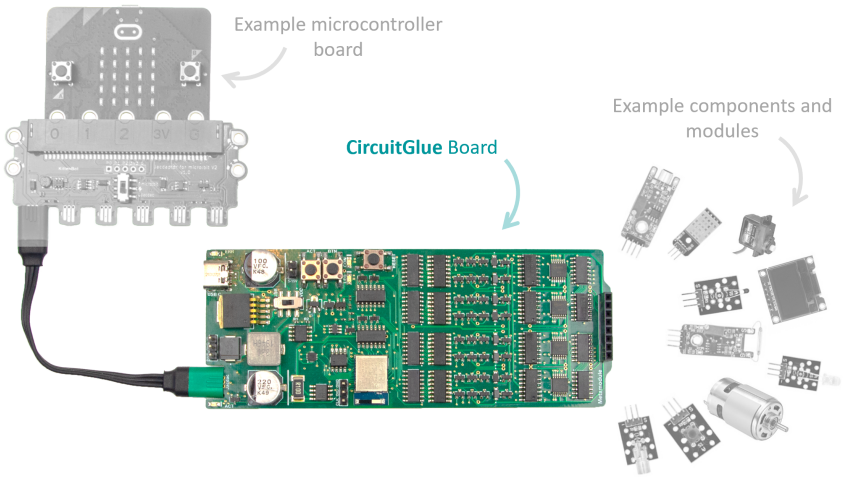

Fig. 1. CircuitGlue is a novel electronic prototyping board that allows a wide variety of off-the-shelf electronic components and modules to be connected to a software configurable header (at right). After configuration and connection, modules work instantly and are compatible with each other independent of the voltage levels, interface types, communication protocols, and pinouts they use.

Authors' addresses: Mannu Lambrichts, mannu.lambrichts@uhasselt.be, Hasselt University - Flanders Make - Expertise Centre for Digital Media, Diepenbeek, Belgium; Raf Ramakers, raf.ramakers@uhasselt.be, Hasselt University - Flanders Make - Expertise Centre for Digital Media, Diepenbeek, Belgium; Steve Hodges, shodges@microsoft.com, Microsoft Research, Cambridge, United Kingdom; James Devine, jamesdevine@microsoft.com, Microsoft Research, Cambridge, United Kingdom; Lorraine Underwood, l.underwood@lancaster.ac.uk, Lancaster University, Lancaster, United Kingdom; Joe Finney, j.finney@lancaster.ac.uk, Lancaster University, Lancaster, United Kingdom.

2474-9567/2023/6-ART63 $15.00
https://doi.org/10.1145/3596265

We present CircuitGlue, an electronic converter board that allows heterogeneous electronic components to be readily interconnected. Electronic components are plugged into an eight-pin programmable header on the board, and the assignment of each pin in the header is configured in software. CircuitGlue supports a variety of connections, including power, ground, analog signals, and various digital protocols at different voltages. As such, off-the-shelf electronic components and modules are instantly compatible no matter what voltage levels, interface types, communication protocols, and pinouts they use. In this paper, we demonstrate the use of CircuitGlue to ease and expedite prototyping with electronics and we explore new opportunities enabled by CircuitGlue. Finally, we reflect on the results of a preliminary user study evaluating the usability of CircuitGlue for people new to electronics.

CCS Concepts: • **Hardware → Programmable interconnect**; *Reconfigurable logic applications*; Digital signal processing; • **Human-centered computing → Human computer interaction (HCI)**.

Additional Key Words and Phrases: Electronics, Prototyping, Software Programmable Converter

ACM Reference Format:
Mannu Lambrichts, Raf Ramakers, Steve Hodges, James Devine, Lorraine Underwood, and Joe Finney. 2023. CircuitGlue: A Software Configurable Converter for Interconnecting Multiple Heterogeneous Electronic Components. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 63 (June 2023), 30 pages. https://doi.org/10.1145/3596265

1 INTRODUCTION

With the growing availability and popularity of physical computing, people with increasingly diverse backgrounds are prototyping electronic circuits. Breadboards and jumper wires are among the most common tools for this, but this electronic prototyping style, sometimes referred to as 'Type 1' electronics [7, 21], requires significant electronic expertise as all components are wired individually. For some components that are only available in small package sizes or need custom circuitry, breakout boards have been created, such as the Adafruit BNO055 [1] and ESP8266 [11] breakout boards. Lambrichts et al. [21] referred to prototyping using breakout boards and development boards as 'Type 2' electronics. Although this style partially eases prototyping because only the most essential connections are exposed, these modules are typically manufactured by different parties and their operating voltages, interface types, communication speeds and protocols and physical connections are often incompatible. Commercially available power and protocol conversion modules, such as the Sparkfun Buck-Boost Converter[1] and Adafruit FT232H[2] breakout board can help in this process, but selecting and interfacing between all components appropriately still requires a good understanding of electronics. Although breakout boards and development boards ease the design and creation of custom prototypes, selecting and interconnecting these components is often difficult for novices [25].

To empower more people—especially those with non-electronics backgrounds—to prototype sensor systems, various integrated modular platforms have been developed. These consist of a set of modules specifically designed to plug together without the need to study technical specifications in datasheets or to acquire any third party components. Popular examples include .NET Gadgeteer [14], littleBits [6], and LEGO Mindstorms [24]. Lambrichts et al. [21] refer to this style of prototyping as 'Type 3'. However, an online survey revealed that hobby makers and engineers are often non inclined to use these Type 3 prototyping kits as they do not want to lock themselves into ecosystems [21], and it can be hard for individuals to extend them with new modules,

To combine the versatility and extensibility of Type 2 electronics and the ease of use of Type 3 prototyping kits, we present CircuitGlue. CircuitGlue enables novices to quickly and easily prototype interactive systems without being locked into a particular ecosystem. CircuitGlue is an electronic prototyping board (Figure 1) that exposes eight "programmable" header pins, that directly interface with a wide variety of third-party components and modules. Each of the eight header pins can be programmed to either connect to ground, output a specific

[1] https://www.sparkfun.com/products/15208
[2] https://www.adafruit.com/product/2264

voltage, or support an analog or digital reading or signal generation. Once the CircuitGlue board is configured, modules are immediately operational for testing or integration in a prototype—they are powered and data and sensor readings are available on a digital bus. CircuitGlue also implements the Jacdac protocol stack [9], which means a standardized, abstracted representation of the component is presented, which allows the component to be integrated with code using any of the Jacdac-supported programming paradigms, including Microsoft MakeCode. Afterward, our complementary software environment further assists by showing the user how to connect the module directly to a development board. In this way, the CircuitGlue board is 'freed up' and can then be reused for interfacing with another module or component. This approach helps novices in electronics to assemble advanced prototypes in iterations.

The core contribution of this work is CircuitGlue, a novel intelligent software-configurable prototyping board that facilitates interconnecting and testing various heterogeneous electronic components and modules. More specifically, we contribute:

- The CircuitGlue board, exposing eight programmable header pins that are configurable in software to drive a wide variety of electronic components. We benchmark our board's design via a technical evaluation.
- A software architecture that makes the functionality of electronic modules and components available on the Jacdac communications bus. This makes many commercially available electronic modules compatible with each other and the Jacdac ecosystem.
- A demonstration of how CircuitGlue facilitates and enriches electronic prototyping workflows and helps with testing components as well as making electronic prototypes.
- A preliminary user evaluation reporting on the utility of CircuitGlue for novices in electronics.

2 WALKTHROUGH

This walkthrough demonstrates how Sam, a novice prototyping enthusiast, uses CircuitGlue to prototype a smart desktop fan. The fan is powered by a DC motor and consists of a temperature sensor and a presence sensor to automatically power the fan when the temperature is too high and presence is detected. Prototyping this interactive system with a microcontroller-based development board, such as the BBC micro:bit, would typically require inspecting the datasheet of all three components, finding and ordering an additional motor driver board as well as a DC-DC voltage converter, and figuring out the correct wiring of all these components as shown in Figure 4a. However, Sam is uncertain about the exact workings of all these components and oftentimes does not understand the myriad of characteristics provided in datasheets. Therefore, Sam uses the CircuitGlue platform in the prototyping scenario below, allowing him to interconnect the heterogeneous set of electronic components needed in this project.

Sam starts with the temperature sensor module. Instead of looking up detailed temperature sensor characteristics in its datasheet, Sam connects a CircuitGlue board to his computer with a USB cable and to the micro:bit with a Jacdac cable and a micro:bit Jacdac adapter (Figure 2a - 1). Next, he opens the CircuitGlue configuration tool in a browser and selects the temperature sensor (DHT11) from the list of components currently implemented in CircuitGlue. A visual representation of the CircuitGlue board and connected temperature sensor is shown (Figure 2a - 2 and Figure 2b). After the configuration tool has automatically configured the CircuitGlue board, Sam takes the temperature sensor and plugs it into the programmable header matching the position shown on-screen in the configuration tool (Figure 2a - 3). When browsing to the Jacdac dashboard[3], he immediately sees a "digital twin" of the temperature sensor with live sensor readings (Figure 3a). Sam now writes application logic for the temperature sensor by programming the micro:bit using, for example, Microsoft MakeCode[4]. MakeCode is one of

[3] https://microsoft.github.io/jacdac-docs/dashboard
[4] https://makecode.microbit.org/

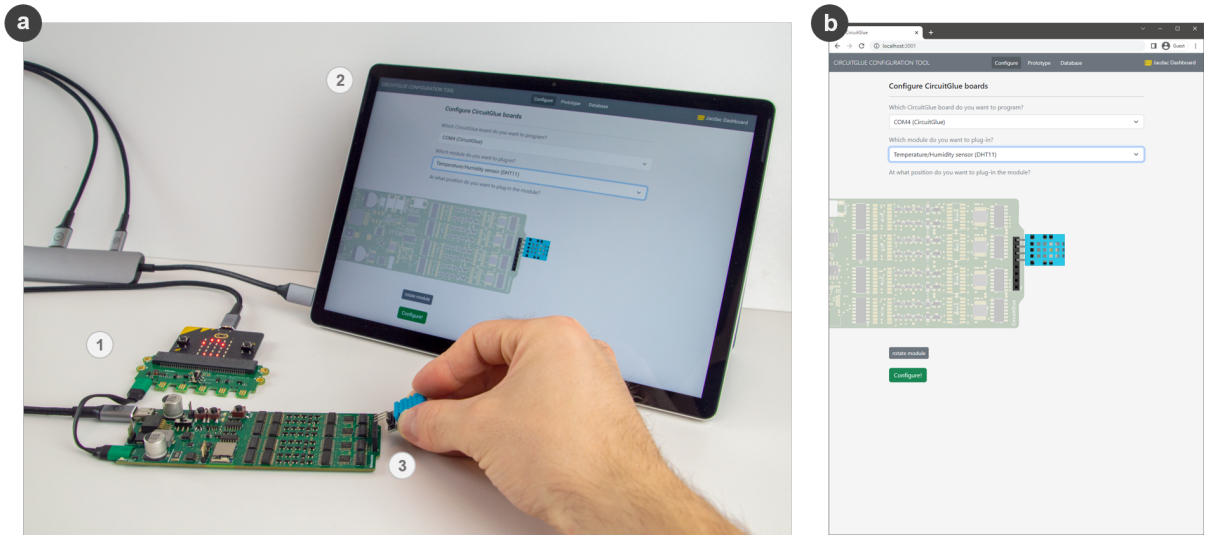

Fig. 2. a) Configuring the CircuitGlue board to drive the temperature sensor by (1) connecting the CircuitGlue board to the computer and micro:bit, (2) configuring CircuitGlue by selecting the temperature module in the configuration tool, and (3) plugging the temperature sensor module into the programmable header. b) Web-based CircuitGlue configuration tool.

several programming solutions that support the Jacdac communication protocol, and as such all components available on the digital Jacdac bus are available as blocks in MakeCode (Figure 3b).

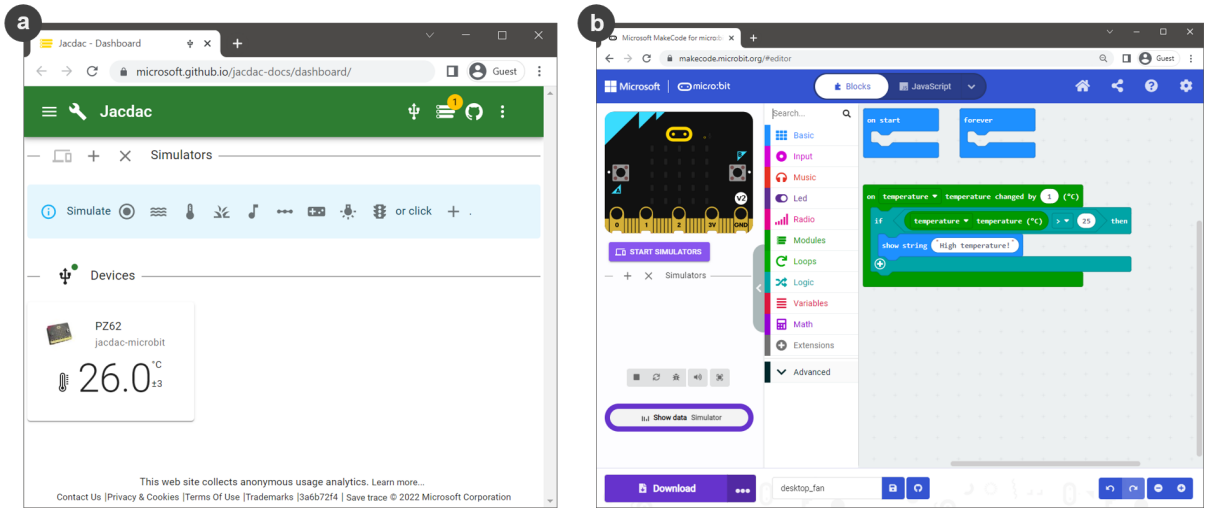

Fig. 3. a) Visualizing the reading of the temperature sensor module in the Jacdac dashboard. b) Writing the application logic on the micro:bit using MakeCode building blocks.

Next, Sam connects a second CircuitGlue board to his computer and the same micro:bit and uses the CircuitGlue configuration tool to select the DC motor (Brushed - 12V DC). He doesn't need to add a motor driver module

and external 12V power supply, the DC motor is instantly operational and visible via the Jacdac dashboard after plugging the motor directly into the CircuitGlue board. Like the temperature sensor—the DC motor is now also available as a block in MakeCode. Sam then programs the micro:bit to power the DC motor when the temperature reading exceeds 25° C.

When Sam is happy with his CircuitGlue-based prototype, he can use the *circuit diagram generator* in the CircuitGlue configuration tool to help him wire the temperature sensor and DC motor directly to the micro:bit, freeing up the CircuitGlue boards. As shown in Figure 4a, this feature uses the knowledge of the characteristics of all components in CircuitGlue to render a custom circuit diagram. Sam uses this diagram to connect the temperature sensor and the DC motor (using an L298N DC motor driver and 12V/3A power supply) to the micro:bit. Even though both sensors are now directly connected to the micro:bit without CircuitGlue boards, CircuitGlue ensures they are still available on the Jacdac bus as blocks in MakeCode. As such, Sam's prototype is still operational using the same application logic after taking out the CircuitGlue boards.

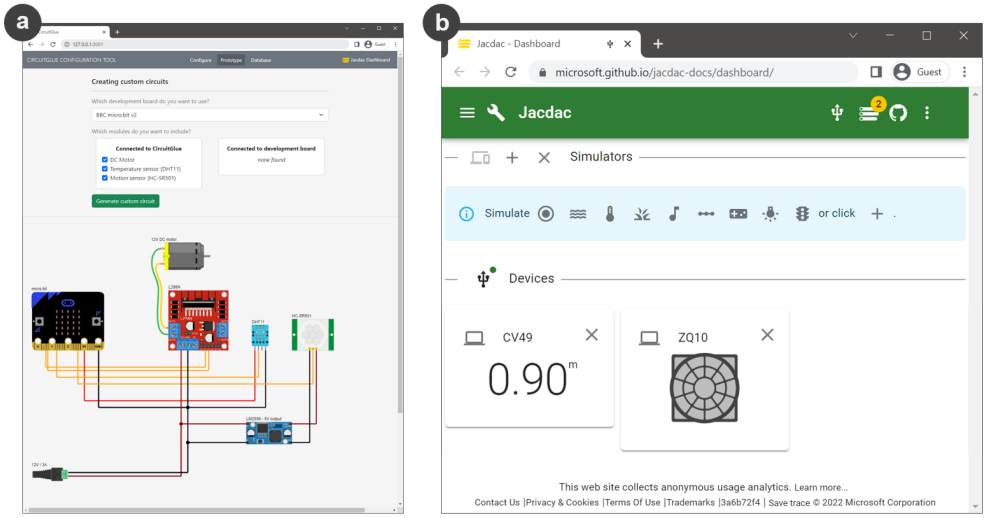

Fig. 4. a) Circuit diagram generated to facilitate building a custom circuit using the DC motor, temperature sensor and PIR motion sensor. b) Comparing the ultrasonic distance sensor and PIR motion sensor side-to-side in the Jacdac dashboard.

Sam is uncertain whether to use an ultrasonic distance sensor or a PIR motion sensor for detecting presence. To compare the behavior of both sensors, Sam configures one CircuitGlue board to interface with the ultrasonic distance sensor (HC-SR04) and the other for the PIR motion sensor (HC-SR501). After plugging in both sensors, readings of both of them are visible side-by-side via the Jacdac dashboard (Figure 4b). Sam experiments a bit and decides the PIR motion sensor is more effective for detecting presence in a room. He updates the application logic on the micro:bit to only power the DC motor when presence is detected and the temperature exceeds 25° C. When satisfied with the prototyped system, Sam can use the *circuit diagram generator* again to receive instructions for connecting the PIR motion sensor directly the micro:bit, making the CircuitGlue boards available for his next project.

3 RELATED WORK

This work draws from, and builds upon prior work on modules for electronic prototyping [21], tools to ease breadboarding and development [15], and reprogrammable integrated circuits.

3.1 Modules for Electronics Prototyping

Since the birth of the electronics industry, engineers have naturally sought ways to accelerate the prototyping process of electronic devices. Instead of working with individual electronic components only, many prototyping practices nowadays involve extending or interconnecting ready-made PCBs, such as breakout boards (e.g. for the BNO055 [1] and ESP8266 [11]), and development boards (including the BBC micro:bit [26], Arduino [5], and the Raspberry Pi [28]). Prototyping with such boards is usually more convenient than working directly with the ICs but still requires looking up technical details in datasheets, and the use of additional voltage and protocol conversion modules because these boards are typically manufactured by different companies.

Integrated modular systems, on the other hand, offer complete sets of modules specifically designed to work together without needing any other components. Examples include .NET Gadgeteer [14], SoftMod [22], Phidgets [13], and SAM Labs [20]. However, adding modules or components not compatible with the integrated modular system is usually hard—their technical specifications may not be exposed and even if they are, additional voltage and protocol converters are often necessary. In many ways, CircuitGlue brings the plug-and-play benefits of integrated modular systems to breakout boards by offering a converter that ensures compatibility between heterogeneous electronic components.

3.2 Tools to Ease Breadboarding and Development

As breadboards are one of the most popular tools for circuit prototyping, researchers frequently present tools to facilitate designing, building, and testing electronic prototypes on breadboards. Prototyping boards such as ToastBoard [10] and CurrentViz [38] enable circuit inspection by continuously measuring and visualizing voltage and current levels throughout a breadboard circuit. In addition to visualizing the state of a breadboard, SchemaBoard [19] instructs users how to connect components on a breadboard using integrated LEDs and HeyTeddy [17] guides users using text or voice conversations. Trigger-Action-Circuits [4] facilitates breadboarding by generating all the necessary circuitry, firmware, and assembly instructions based on simple behavioral descriptions. In contrast, Circuito.io [8] automatically generates breadboard connection diagrams based on a set of input/output modules and a microcontroller. Similar to these existing techniques, CircuitGlue facilitates building breadboard prototypes by demonstrating how a component can be connected to a development board using common off-the-shelf conversion modules instead of a CircuitGlue board.

Another important issue of breadboarding is tangling of wires which makes circuits fragile and error-prone. CircuitStack [36] contributes a new breadboard design that addresses this issue by interconnecting components via a printed circuit board instead of using jumper wires. Rather than using physical jumper wires, VirtualWire [23] takes a different approach and allows rows on a breadboard to be connected in software. Instead of requiring all components to be present on a breadboard, Proxino [37] injects software-generated signals to replace any missing components. Contributing to this line of research, CircuitGlue also avoids wired connections by allowing modules to directly connect to the CircuitGlue board.

Finally, prototypers frequently rely on a range of test and measurement equipment during the development process. Obvious examples are a huge variety of widely available power supplies, multimeters and oscilloscopes. Of particular note are low cost PC accessories such as the Bus Pirate serial communications monitor[5], Saleae logic analyzers[6] and Digilent's Analog Discovery unit[7].

[5] http://dangerousprototypes.com/docs/Bus_Pirate
[6] https://www.saleae.com
[7] https://digilent.com/reference/test-and-measurement/analog-discovery/start

3.3 Reprogrammable Integrated Circuits

Alphonsus et al. [2] offer an extensive overview of various types of reprogrammable integrated circuits and their applications, including field-programmable gate arrays (FPGAs) [31] and programmable system-on-chips (PSoCs) [16]. Over the past few years, reprogrammable integrated circuits have become more common in HCI. Scanalog [35], for example, uses a field-programmable analog array (FPAA) [3] to facilitate interactive design and debugging of analog circuits by using direct manipulation. A special type of reprogrammable integrated circuit is the crosspoint switch, which has been used in several interactive systems lately. VirtualWire [23], for example, uses a crosspoint switch to allow users to interconnect different rows on a breadboard in software. Similarly, VirtualComponent [18] uses a crosspoint switch to connect and disconnect specific component banks on a printed circuit board. CircuitGlue goes beyond software configurable connections between header pins [23] and peripherals [16] by offering programmable voltage power delivery and conversion of digital protocols.

Several microcontrollers also embed features of reprogrammable integrated circuits. The nRF52 series [32], for example, implements a programmable peripheral interconnect (PPI) [33] that allows dynamic pin mapping, configuration and allocation of resources, and enables peripherals to communicate autonomously independently of the CPU. Similarly, the RP2040 processor of the Raspberry Pi Pico [27] includes a form of software programmable digital hardware called programmable input/output (PIO) [29]. While these approaches are very versatile, they can only be used to route low-current signals—unlike the programmable header pins of CircuitGlue, they are not suitable for powering external electronics.

4 DESIGN RATIONALE

To guide the design of our CircuitGlue and gather early feedback on the prototyping styles that are interesting to potential user groups, we conducted informal interviews using conceptual renderings of the CircuitGlue board (Figure 5).

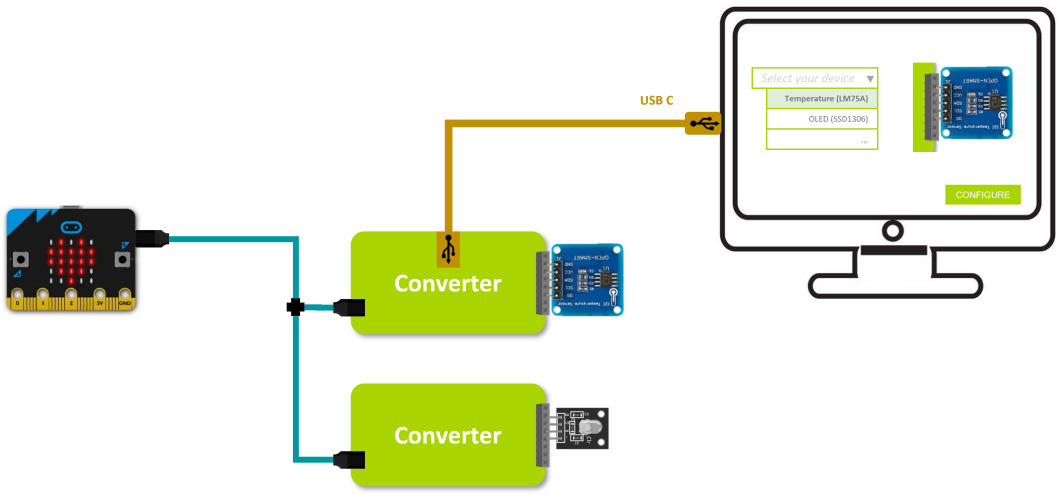

Fig. 5. Example of the conceptual renderings used in the interviews to gather early feedback of the CircuitGlue concept.

4.1 Early Feedback on the CircuitGlue Concept

We conducted informal interviews with three potential users: a teacher, a maker, and an electronics engineer, all recruited from our network. Given the limited number of interviews, the results reported in this section simply offer initial feedback on CircuitGlue to gauge perceptions of the concept and do not necessarily reflect interest from the broader community. The interviews were conducted online and lasted 30 minutes.

To correctly interpret and contextualize participants' comments, we first invited them to fill in a short questionnaire asking about their previous prototyping experiences. The teacher reported using integrated modular systems, such as .NET Gadgeteer [14] frequently, whereas the maker and electronics engineer only used breakout boards or off-the-shelf electronic components.

At the start of the interview, we introduced the CircuitGlue concept as a black box for interfacing with any type of electronic component, including modules. For the teacher, this already had "*massive potential if I could hook up stuff without having to worry about the polarity*", and the maker also saw potential in CircuitGlue for rapid prototyping. We then introduced various potential functionalities and prototyping styles (covered in detail in Section 8), using ten conceptual renderings. After explaining the prototyping style, we asked participants how they valued the scenario and when it could be useful in their prototyping practices.

Both the maker and the teacher were enthusiastic about using CircuitGlue to quickly test components without needing to create complex circuits (Section 8.2). While the maker saw a lot of potential in using CircuitGlue to work with more complex components, such as motors and valves, the teacher's initial response was that s/he would still prefer using standard motor shields in the classroom as these shields are very cheap and familiar to him/her. The opportunity to automatically generate a custom circuit diagram, based on the knowledge of electronic components built into CircuitGlue, was especially appealing to the maker who liked "*ending up with a custom circuit without needing to do the thinking*". Finally, we presented a scenario in which CircuitGlue was used in a similar way to a development board (Section 8.5). The maker in particular saw potential in writing custom software while still using the programmable voltage and logic level conversions provided by CircuitGlue. In contrast, the teacher appreciated the compatibility of CircuitGlue with micro:bit as s/he already used this platform in the classroom.

Throughout the interview, the electronics engineer did not find many scenarios appealing as s/he was used to looking in datasheets and did not trust libraries developed by third parties as "*they can potentially blow up modules*." The engineer, however, thought CircuitGlue would be great in educational settings.

4.2 Design Decisions

The CircuitGlue board exposes eight programmable header pins of which the pin assignments are programmed in software; each can be configured to either output digital signals, read analog or digital signals, connect to ground or deliver power, with the latter being programmable in steps of 0.1V. Our board supports digital communication signals via the programmable header pins at both 3.3V and 5V and supports analog readings up to 12V. The board is powered and can be programmed via an onboard USB-C connector. In situations where a module's pinout does not fit our single row eight header pin configuration, such as modules that expose two rows, a small breadboard or adapter board can be used to connect to CircuitGlue. The CircuitGlue board supports components and modules that operate between 1.8V to 12V and supplies up to 3A. To allow for easy interconnection between a CircuitGlue board and Jacdac compatible devices, such as the micro:bit, our board embeds the Jacdac edge connector[8]. In addition to USB-C, the CircuitGlue board can also be programmed using Jacdac.

[8] https://microsoft.github.io/jacdac-docs/ddk/design/electro-mechanical

4.3 Jacdac as Bus Protocol

To allow a wide variety of electronic components and modules to intercommunicate, CircuitGlue needs to translate analog readings and digital signals, such as I2C and SPI, to a common bus protocol. This is realized using a short snippet of translation code for every electronic component, see Section 5 for more details.

CircuitGlue implements the Jacdac [9] communication protocol, which is built on top of Single Wire Serial (SWS), as the common bus protocol. By translating all analog and digital signals into Jacdac packets, every module or component connected to the CircuitGlue board can be recognized as a new device on the Jacdac bus. This enables seamless intercommunication between heterogeneous electronic components and modules, as demonstrated in our Walkthrough (Section 2).

CircuitGlue could have also used a different existing protocol, such as I2C, SPI, or a CAN bus, but supporting Jacdac on CircuitGlue further facilitates working with electronics. Firstly, Jacdac announces the components and their functionality as soon as they are connected. This allows instant interaction and experimentation with components and modules via the Jacdac dashboard[3]. Secondly, Jacdac introduces a software abstraction via its "services" layer. Each Jacdac service exposes a basic function, such as a button or accelerometer. This abstraction makes it easy to interface with any given type of component in exactly the same way independently of the specific hardware. For example, the ADXL345 is an accelerometer that makes readings available over I2C or SPI, while the MMA7361 accelerometer outputs analog voltages. By using the Jacdac accelerometer service, acceleration readings for both sensors use exactly the same methods. As such, one can substitute a component with a similar one from a different supplier without making changes to application logic.

5 SUPPORTING NEW MODULES

As demonstrated in the Walkthrough (Section 2), the CircuitGlue configuration tool allows users to simply select the electronic module they want from a list of supported modules. The CircuitGlue configuration tool then uploads the specifications for all eight programmable header pins to the CircuitGlue board and flashes the driver, referred to as "translation code", to correctly interface with the connected module and expose its functionality on the Jacdac bus.

To add support for additional modules to CircuitGlue, the new module's pin configuration must be specified, and a translation code snippet has to be written. Writing the translation code requires implementing the respective Jacdac service layer[9], such as the accelerometer service for an ADXL345 accelerometer. The translation code must call the initialization function of the appropriate Jacdac service and passing function pointers to custom-written `initialization()`, `update()`, and `read()` functions. In these three functions, custom code can be written or calls can be made to an existing library, such as an Arduino library. The example in Appendix A shows translation code for supporting the Jacdac temperature service using a DHT11 temperature sensor. More advanced components sometimes require additional functions to be implemented, as specified in the Jacdac documentation[10]. When a module offers multiple functionalities, multiple services should be implemented.

Although writing translation code requires more technical expertise than simply using CircuitGlue, being able to leverage the Jacdac programming paradigm and potentially also re-use Arduino driver code makes it relatively straightforward. We also note that it's a one-time effort—we envision users sharing these configurations and drivers via community platforms in the future.

Translation code is platform agnostic; this means the same code can be used for controlling a module via the Jacdac protocol even if a CircuitGlue board is not used. This is essential to ensure prototypes remain functional when modules are connected to a development board with custom circuitry with the help of the CircuitGlue *circuit diagram generator* feature.

[9] https://microsoft.github.io/jacdac-docs/services/
[10] https://microsoft.github.io/jacdac-docs/ddk/services/#implementing-service-firmware

In addition to writing translation code to expose a new module on the Jacdac bus, it's pinout must be specified. The CircuitGlue configuration tool streamlines the process of supporting new modules by offering an interface for specifying the pinout and required voltages of a new module (Figure 6). The configuration tool also embeds additional intelligence that encodes and checks various heuristics relating to common protocols. For example, when only one of the two pins required for the I2C protocol is specified, the interface provides a warning.

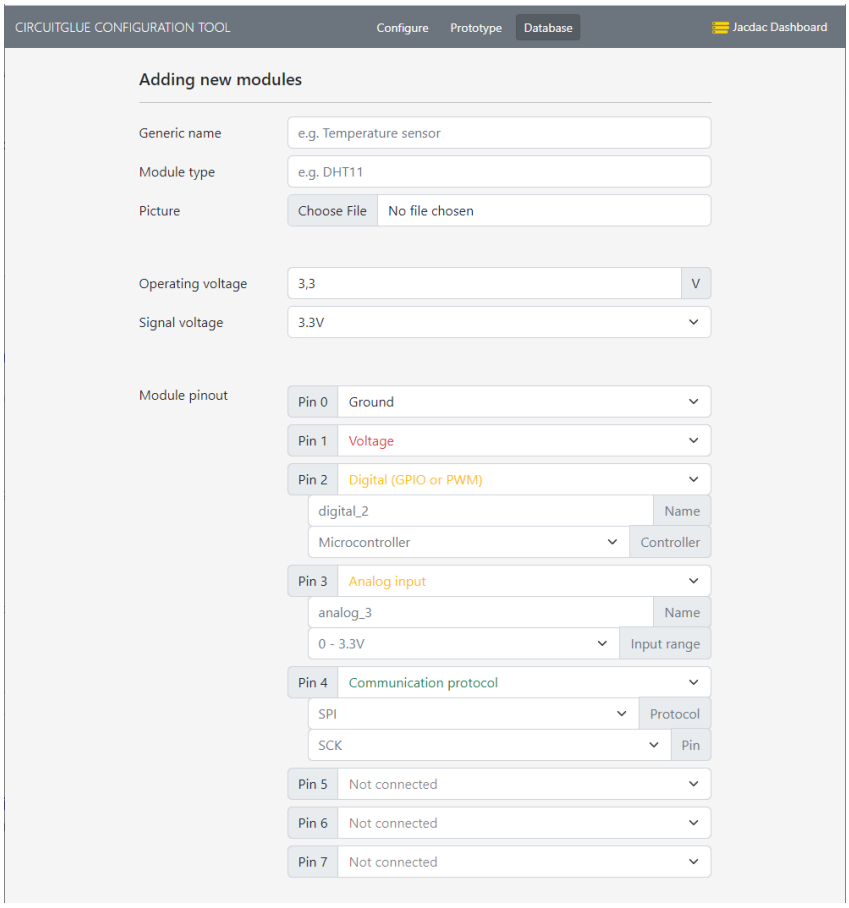

Fig. 6. Adding a new module to the database using the configuration tool.

6 CIRCUITGLUE HARDWARE DESIGN

To structure a discussion of the hardware design of the CircuitGlue board, we split it into three sections (Figure 7): (1) a system-on-chip (SoC) controlling and monitoring the board, (2) voltage regulation and power delivery, and (3) hardware components for configuring each of the eight programmable header pins.

6.1 System-on-Chip

We decided to use a microcontroller SoC to control the CircuitGlue board instead of configurable integrated logic chips, such as programmable system-on-chips (PSoCs) and field-programmable gate arrays (FPGAs), as many

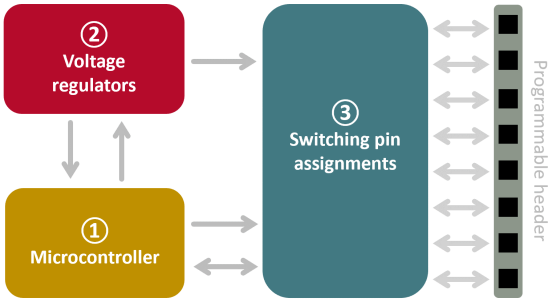

Fig. 7. The design of the CircuitGlue board with (1) a System-on-Chip (SoC) controlling and monitoring the board, (2) voltage regulation and power delivery, and (3) hardware components for switching the assignment of programmable pins.

existing software libraries for prototyping with electronics, including Jacdac, are available for microcontrollers. The CircuitGlue board is built around the nRF52840; this SoC allows dynamic assignment of peripherals to pins and therefore does not need additional hardware components, such as crosspoint switches, to re-map the functionality of CircuitGlue's programmable header pins. To reduce the number of electronic components on the board, we decided to use Raytac's nRF52840 module. This module embeds the nRF52840 SoC and complementary components, such as capacitors and an antenna.

6.2 Regulating Power

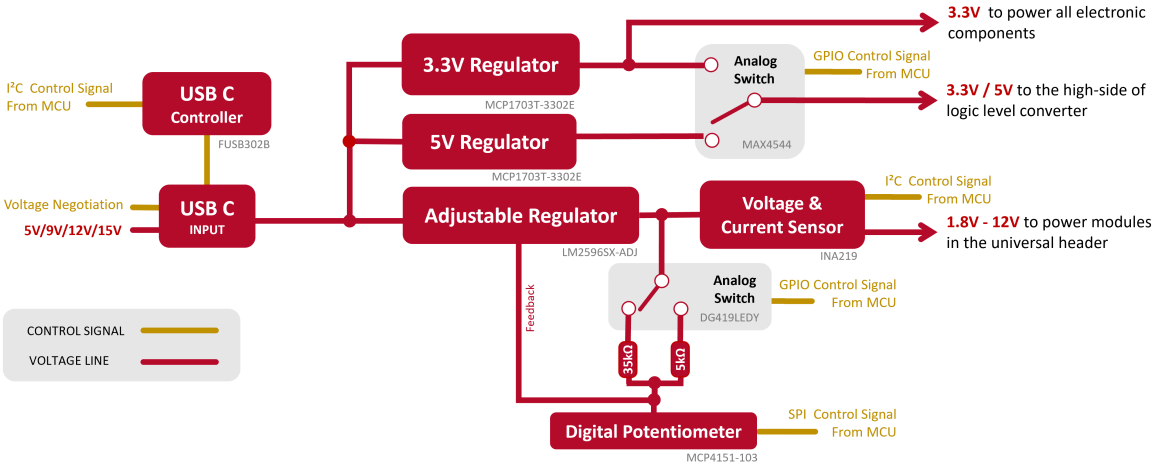

Fig. 8. Block diagram with all regulators responsible for providing the three voltage levels used by the CircuitGlue board.

The CircuitGlue board is powered through the onboard USB-C connector using Power Delivery (PD) and accepts voltages between 5V and 15V. A USB-C PD controller on the CircuitGlue board allows voltage negotiations

with USB-C PD compatible power supplies and is configured to always deliver the highest available voltage (with a maximum of 15V) to the CircuitGlue board. When Power Delivery is unavailable, the CircuitGlue board can either be powered from the 5V USB connection or by connecting an external power supply to the board. Using the input voltage, the CircuitGlue board internally regulates three voltage levels for its operation:

- **Board voltage:** A low voltage of 3.3V to power all electronic components on the CircuitGlue board. As shown in Figure 8, this voltage is supplied by a low-dropout 3.3V regulator.
- **Signal voltage:** Used by the logic level converters (described in Section 6.3) to translate digital communication signals from the board voltage to the voltage required for communicating with the electronic module plugged into the programmable header. As the majority of electronic modules require communication signals of either 3.3V or 5V, the signal voltage can toggle between these two voltages using a digital switch, as shown in Figure 8.
- **Programmable voltage:** The voltage used to power modules or components plugged into the programmable header. To supply a large selection of electronic modules, this voltage line is programmable from 1.8 to 12V in steps of 0.1V via an an adjustable voltage regulator controlled by a 10kΩ digital potentiometer driven by the nRF52840 SoC. This can deliver a current up to 3A. See Figure 8 for details. While the digital potentiometer has 256 steps, it is impossible to achieve the required voltage range of 1.8V to 12V in steps of 0.1V due to the logarithmic scale in the output of the voltage divider. Therefore, we dynamically switch between two ranges using an analog switch. The graph shown in Figure 9 shows that swapping between our two ranges (by switching between a resistance of 35kΩ and 5kΩ) offers the full voltage range with the desired accuracy. As shown in Figure 8, a voltage and current sensor is added to measure the actual voltage and current usage of a module connected to the programmable header. This information is later used to verify the programmed voltage and provide basic protection against shorts and faults (Section 7).

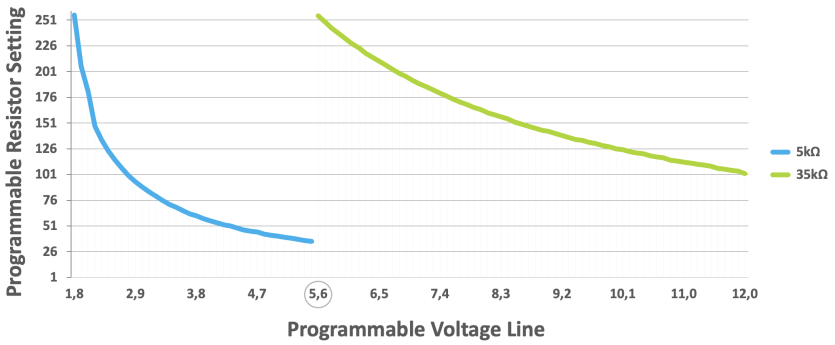

Fig. 9. The required setting in the digital potentiometer based on the requested output voltage and selected top resistor.

6.3 Changing the Assignment of a Programmable Header Pin

As conceptualized in Figure 7, the CircuitGlue SoC controls everything necessary to correctly assign the eight programmable header pins on the CircuitGlue board. Figure 10 shows this in more detail. The dashed blocks in this figure represent the electronic circuit for setting the assignment of exactly one programmable header pin; eight such identical blocks are present on the CircuitGlue board. Each of these embeds the circuitry to either configure the programmable header pin (1) as a general purpose input/output (GPIO) pin, (2) as an analog input pin, (3) as a connection to ground, or (4) as a programmable voltage supply.

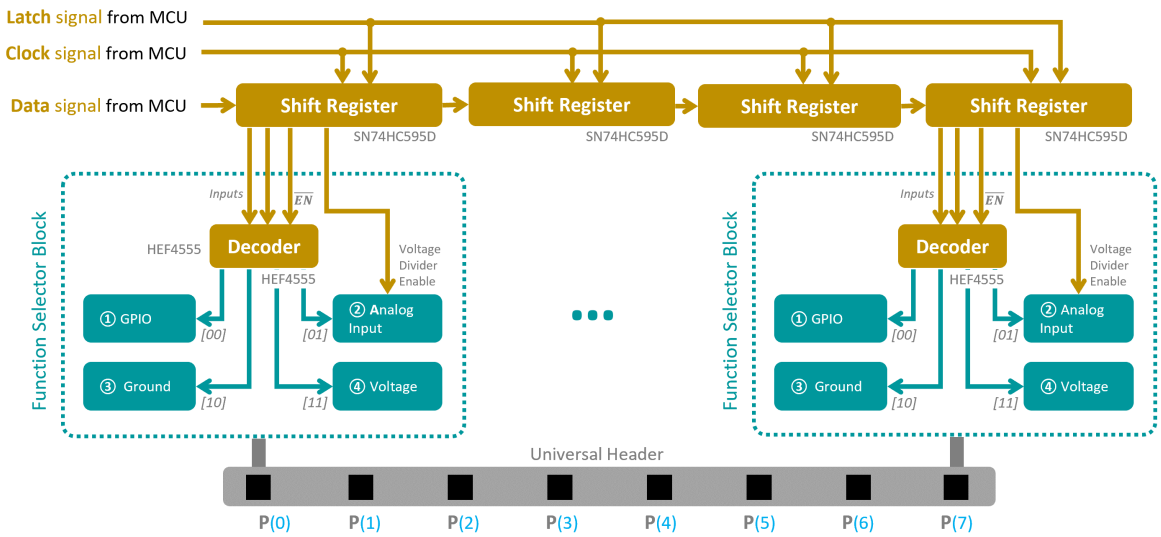

Fig. 10. Block diagram of all components required for changing the assignments of the programmable header pins of the CircuitGlue board.

Shift registers connected in series control all of the eight identical dashed blocks and are driven by the nRF52840 SoC. For each dashed block, two shift register output pins activate the appropriate circuit in a block by using a 1-of-4 decoder to ensure only one of the four circuits is active at any time. As shown in Figure 10, a third shift register output is used to disable the decoder when the programmable header pin needs to be in a high impedance state. As we will further explain below, a fourth shift register output is used to activate a voltage divider which is part of the circuit for configuring the programmable header pin as an analog input pin.

6.3.1 Programmable Header Pin as GPIO. As shown in Figure 11, an analog switch controlled by the decoder connects each of the programmable header pins of the CircuitGlue board to a high-speed GPIO pin on the nRF52840 SoC. While all components on the CircuitGlue board work with 3.3V signals, a bidirectional logic level converter allows 5V signaling, as dictated by the plugged-in module (as discussed in Section 6.2).

The logic level converter supports both push-pull and open-drain applications and can achieve speeds up to 24Mbps and 2Mbps respectively, sufficient for the popular digital communication protocols. Two methods are built-in to protect the board when users accidentally apply a higher voltage on one of the programmable header pins. First, a comparator and AND gate instantly overwrite the signal from the decoder to disconnect the analog switch when the voltage applied to the programmable header pin exceeds the signal voltage. A non-inverting summing amplifier slightly increases the signal voltage to prevent undesired cut-offs of the logic signal. Second, a clamping diode, shown in Figure 11, further prevents voltages higher than 3.3V to pass to the nRF52840 SoC.

6.3.2 Programmable Header Pin as Analog Input. As shown in Figure 12, an analog switch controlled by the decoder connects the programmable header pin of the CircuitGlue board to an analog input pin of the nRF52840 SoC. The nRF52840 embeds a 12-bit analog-to-digital converter and supports voltages up to 3.3V. Analog input readings up to 12V are supported at the expense of the accuracy using a voltage divider controlled by the nRF52840 SoC, see Figure 12. This allows CircuitGlue to make resistance measurements to support resistive sensors such as

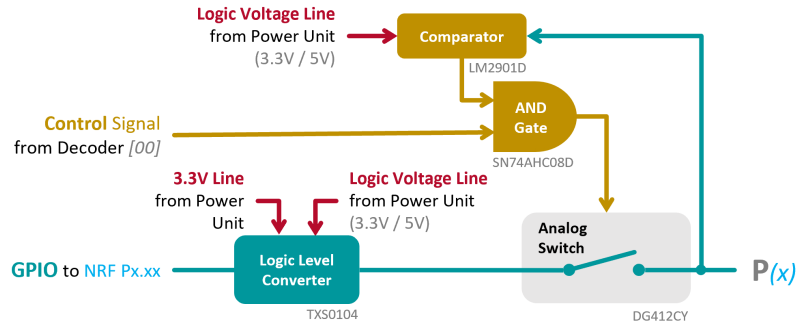

Fig. 11. Circuit for connecting the programmable header pin to a digital high-speed GPIO pin on the nRF52840 SoC.

photoresistors. A comparator and OR gate automatically turn on the voltage divider when the voltage supplied on the programmable header pin exceeds 3.3V, and a non-inverting summing amplifier circuit allows small voltage peaks to surpass the threshold. Finally, a feedback signal informs the nRF52840 SoC when the voltage divider is activated, so the actual voltage on the programmable header pin can be calculated.

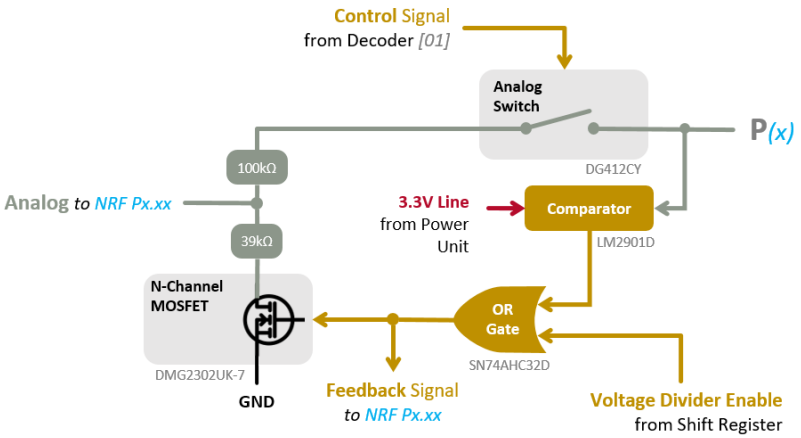

Fig. 12. Circuit for connecting the programmable header pin to an analog pin on the nRF52840 SoC.

6.3.3 Programmable Header Pin as Ground. As shown in Figure 13, an N-channel MOSFET controlled by the decoder is used for connecting a programmable header pin to ground. While using an analog switch would

entirely disconnect a line, MOSFETs are more suitable in this part of the circuit as they support higher currents and are available in smaller package sizes. CircuitGlue uses MOSFETs that support up to 2.8A of continuous current.

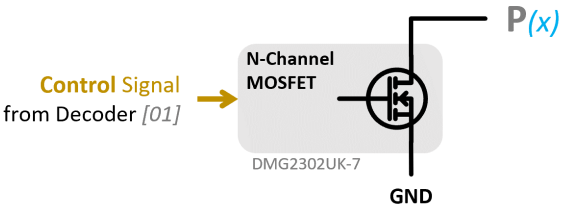

Fig. 13. Circuit for connecting the programmable header pin to ground.

6.3.4 Programmable Header Pin as Power. Similar to the connection to ground, the P-channel MOSFET in Figure 14 is used for connecting a programmable header pin to the programmable voltage line (Figure 8). A signal translator consisting of an N-channel MOSFET translates the signal from the decoder as a P-channel MOSFET requires the control voltage to be in the same range as the source voltage (1.8V-12V). A diode prevents reverse current through the P-channel MOSFET avoiding issues when the programmable header pin is used for digital or analog signals of a voltage higher than the programmable voltage.

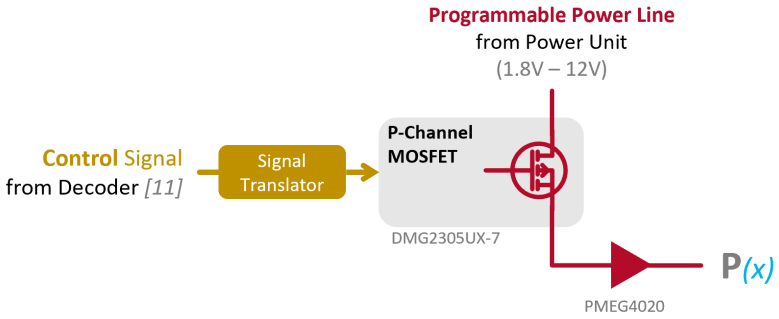

Fig. 14. Circuit for connecting the programmable header pin to the programmable voltage level.

6.4 Circuit Board Design and Manufacturing

To realize the CircuitGlue board, we combined all building blocks on a single, four-layer PCB measuring 125 by 50mm. The final iteration uses 229 electronic components on both sides, and for this version of our prototype, components were manually placed and soldered using a reflow oven. We were careful to chose commodity components wherever possible, in order to minimize cost and maximize supply chain flexibility. When manufacturing

ten prototypes, the cost of the CircuitGlue PCB is around $8 (from JLCPCB[11]) plus $49 for all components (from OctoPart[12]). This drops to around $0.50 per PCB and $26 for components quantities of 1k pieces. Given the predominance of commodity components, the component cost is likely to be significantly less on the Chinese market.

7 CIRCUITGLUE SOFTWARE ARCHITECTURE

7.1 CircuitGlue Firmware

The CircuitGlue firmware runs on the nRF52840 SoC and drives all functionalities of the board. If an internal error occurs, the CircuitGlue board disconnects its programmable header pins to avoid damaging any electronics. It furthermore notifies the user by blinking two LEDs on the CircuitGlue board. Such errors include internal components that are not responding as expected, or when the measured voltage or current is outside the required range due to a short circuit or component failure.

The CircuitGlue firmware is written in the C programming language as it offers easy access to all peripherals and registers of the nRF52840 SoC. We built on top of several existing software libraries: the standalone nrfx drivers[13] and libraries provided by the nRF5 SDK[14] ease setting up peripherals, such as timers and communication protocols; and the existing platform-agnostic implementation of the Jacdac protocol[15] and is extended with a platform-specific implementation of peripherals for the nRF52 family (e.g., half-duplex RS232, I2C, SPI, GPIOs, ...) that we wrote ourselves. The CircuitGlue firmware is available on GitHub[16].

The firmware is compiled using the arm-none-eabi compiler[17]. To flash the compiled hex file, we use Device Firmware Upgrade (DFU) over USB. To generate a DFU package from the compiled hex file, we use the "nrfutil pkg generate" tool and flash the DFU package using "nrfutil dfu USB-serial". To enable DFU on the nRF52840 SoC, we flashed the precompiled bootloader which is provided by Nordic Semiconductor. Alternatively, the hex file can be flashed using the "nrfjprog" command or by connecting an external programmer to the Single Wire Debug (SWD) pins on the CircuitGlue board.

7.2 CircuitGlue Configuration Tool

The CircuitGlue configuration tool is implemented as a web interface. The configuration tool uses Jacdac[18] over either USB CDC or WebUSB[19] to communicate with connected CircuitGlue boards. A JSON database contains all technical specifications, such as required voltages and pinouts, and the UF2 firmware file for each supported module. When users configure a module, the configuration tool loads its specifications from the database and sends it to the CircuitGlue board. After that, the firmware containing the translation code is flashed to the CircuitGlue board.

Before sending the specifications and firmware, the CircuitGlue configuration tool automatically enables the CircuitGlue DFU bootloader on the CircuitGlue board. This bootloader allows for convenient firmware updates over Jacdac and disables all programmable pins to protect the connected module from previous configurations.

After flashing, the firmware sets the signal and programmable voltage (Section 6.2) and waits until the programmable voltage reaches its configured setting. Next, the firmware programs each pin assignment in the

[11] https://jlcpcb.com
[12] https://octopart.com
[13] https://github.com/NordicSemiconductor/nrfx
[14] https://www.nordicsemi.com/Products/Development-software/nrf5-sdk
[15] https://github.com/microsoft/jacdac-c
[16] https://github.com/MannuLambrichts/CircuitGlue
[17] https://developer.arm.com/downloads/-/gnu-rm
[18] https://github.com/microsoft/jacdac-ts
[19] https://wicg.github.io/webusb/

shift registers using four bits per programmable pin, as described in Section 6.3. After activating all shift registers, the function of each programmable header pin is assigned. The firmware then initializes all necessary peripherals for the digital communication protocols. Finally, the firmware instantiates the translation code necessary for translating the module's functionality to Jacdac services which makes the module available on the Jacdac bus.

7.3 Circuit Diagram Generator

As demonstrated in the Walkthrough (Section 2), once a module is working as desired, the CircuitGlue configuration tool offers a *circuit diagram generator* feature to assist in connecting it directly to a development board. To support this feature, we developed a pin mapping algorithm. For every pin on every module, our algorithm first assigns a compatible pin on the development board without considering specific voltages. For the final pin assignment, the algorithm aims to maximize the number of modules that can be connected similar to routing strategies of PaperPulse [30]. For example, the SCL and SDA pins used for I2C will only be used for digital signals when all other digital pins are already in use. The algorithm then compares voltages available on the development board with voltages required by the modules and adds logic-level converters and an external power supply when needed. This power supply is selected based on the highest voltage needed and DC-DC converters are additionally added to step-down to other voltage levels.

Some electronic components, such as a DC motor, require additional driver circuits. To support this, we extended our JSON database, detailing the pinout and operating voltages for all modules, with an optional field which details the required driver module needed when converting from CircuitGlue to a custom circuit diagram. The *circuit diagram generator* uses this field to initiate additional driver components, such as the L298N DC motor driver.

In order to reuse application logic after converting to a custom circuit diagram, modules still need to be operational over Jacdac even though they are directly connected via digital, analog, I2C, or SPI peripherals. To realize this, the micro:bit development board runs a custom version of the CircuitGlue firmware which previously ran on the CircuitGlue Board, and the CircuitGlue configuration tool activates the same translation code on the micro:bit.

8 PROTOTYPING STYLES AND BENEFITS

The unique features of CircuitGlue facilitate prototyping with electronics and thus enable several novel prototyping styles. Below, we discuss these novel opportunities.

8.1 Understanding, Testing and Comparing Modules

It is often challenging for novices in electronics to select appropriate electronic components when prototyping an interactive system [39]. Example decisions include: whether to use an accelerometer or gyroscope, ultrasonic distance sensor or infrared distance sensor, or what type of temperature sensor is more suitable. To make an informed decision, novices traditionally consult online articles, books, or datasheets; this can be time-consuming, especially as many such resources are not written for novices [39]. A complementary approach that may be useful in some cases is to simulate certain electronic components or sub-circuits, although this is rarely a substitute for experimenting with and verifying the operation of real components, especially in interactive systems where physical behavior, such as measuring acceleration or movement, often impacts the overall user experience. Using CircuitGlue, electronic components are instantly operational, and their basic operations can be tested immediately via the Jacdac dashboard—which renders real-time digital twins of all components on the Jacdac bus—without writing application logic. As such, novices build an intuitive understanding of components and can decide between components by observing differences in output (Figure 15). Testing components with CircuitGlue

can therefore complement existing resources for understanding electronics, because these practical tests confirm basic knowledge or help fill in knowledge gaps.

The Jacdac dashboard is primarily designed to support exploration and fault-finding; a self-contained interactive system requires custom application logic to define the desired behavior based on its various components. This application logic can be written to run on a PC, for example in python, .NET or a web app, or on an embedded microcontroller; in all cases apps are typically simple compositions of the Jacdac services provided by the relevant modules, simplifying application development. This alleviates many compatibility issues and reduces the complexity of the testing process. We refer the reader to Jacdac [9] for more details.

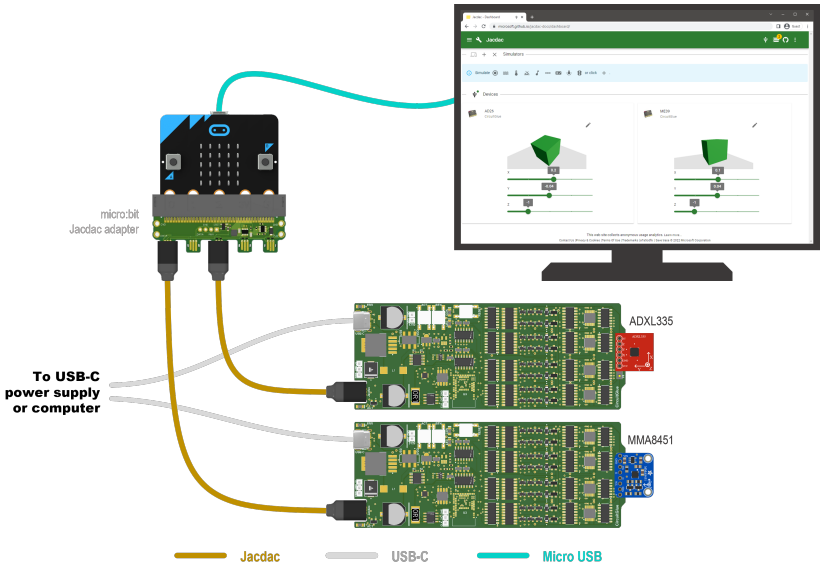

Fig. 15. Comparing two different types of accelerometers in the Jacdac dashboard by using two connected CircuitGlue boards.

8.2 Rapid Prototyping with Heterogeneous Modules

In addition to quickly getting a single module up-and-running, multiple CircuitGlue boards are easily interconnected using Jacdac. As such, interactive systems consisting of components working with different protocols and voltages (Figure 16) are realized in a few minutes. Small electronic modules, requiring only a few pins and the same operating voltage, can even connect to a single CircuitGlue board. In many ways, this style of prototyping combines the ease of use of Type 3 integrated modular systems, with the versatility and flexibility of Type 2 breakout boards and modules.

The *circuit diagram generator*, covered in the next section, helps with reusing CircuitGlue boards within a prototyping process. Application logic, defining the behavior of all components in the sensor system, is easy to write on Jacdac compatible development boards, such as the micro:bit [26], Raspberry Pi [28], or ESP32 [12]. When using MakeCode, which offers seamless support for Jacdac, all components on the Jacdac bus are available as blocks in the programming environment. When using other programming languages, such as .NET, Python, or JavaScript/TypeScript, modules become available by instantiating the Jacdac client service corresponding to the module. A client service provides the interface for interacting with the Jacdac service used by a module.

While it's theoretically possible to use the *circuit diagram generator* to design and implement electronic prototypes without using CircuitGlue as an intermediate evaluation step, this approach is slower, more fiddly and may therefore not be suitable for those who are less experienced. By using CircuitGlue to test and verify individual components, users can gain a practical understanding of how different components work together and ensure compatibility between them before integrating them into the prototype. This not only saves time and reduces the risk of errors, but we believe it will also help users build an intuitive understanding of electronics, which would be beneficial for future projects.

Even users who only prototype with electronics occasionally often amass quite a few electronic components and modules; when creating a new prototype, it is often more convenient to buy new because existing ones might not be compatible with each other or have become deprecated, and thus harder to get operational again. CircuitGlue significantly helps with getting such components operational and ensures compatibility with different generations of components. Therefore, we believe CircuitGlue can help reduce the ecological footprint of electronics prototyping.

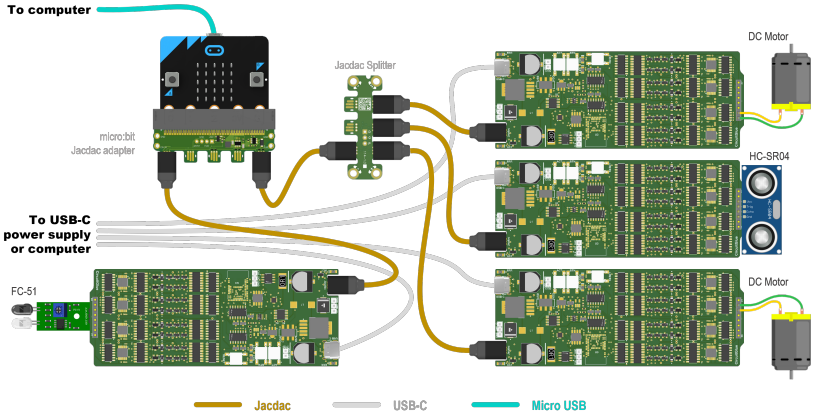

Fig. 16. Building a prototype using multiple CircuitGlue boards.

8.3 Facilitate Breadboarding

When a component is powered by a CircuitGlue board, the *circuit diagram generator* offers a visual guide on connecting all components on the Jacdac bus directly to a development board without using CircuitGlue boards (Figure 17). During this process, CircuitGlue ensures all components are still available on the Jacdac bus and compatible with the application logic, even when they are connected via analog pins or alternative digital protocols, such as I2C or SPI. This allows for a gradual transition from CircuitGlue boards to custom breadboard designs and is especially useful when running out of CircuitGlue boards or when starting with breadboard circuit prototyping. We see this as a top-down style to prototyping electronics as components are first operational using CircuitGlue before revealing lower-level wiring details to turn it into a more traditional breadboard prototype.

8.4 Use Third Party Modules with Jacdac Ecosystem

Instead of prototyping an entire system with CircuitGlue, users of the Jacdac modular system[20] can occasionally turn to CircuitGlue to make third party modules compatible with this ecosystem. Figure 18 demonstrates such a

[20] https://microsoft.github.io/jacdac-docs/devices/kittenbot/jacdacstarterkitawithjacdaptorformicrobitv2v10/

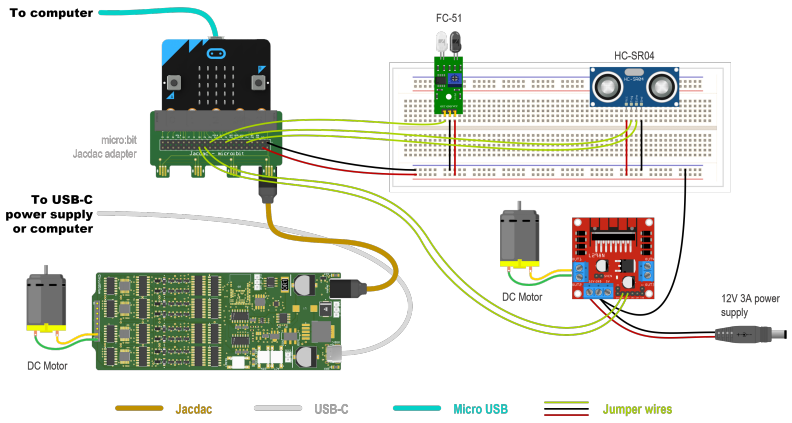

Fig. 17. Building a prototype using a single CircuitGlue board in combination with the *circuit diagram generator* to facilitate building the breadboard circuits.

setup in which a Jacdac button, RGB LED, and temperature sensor interconnect with the SSD1306 display module, currently not part of the Jacdac eco-system, via a CircuitGlue board.

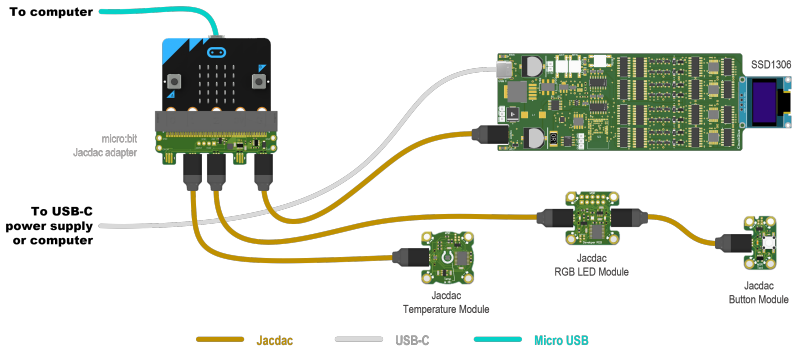

Fig. 18. CircuitGlue used to extend the Jacdac ecosystem with new modules.

8.5 Advanced Use

Users who are more knowledgeable about electronics and embedded programming can use only parts of the CircuitGlue system in their prototyping practices. For example, instead of using a separate development board, such as the micro:bit, for running application logic, one of the CircuitGlue boards can run application logic on its SoC. When using a CircuitGlue board as such, users have to manually modify the CircuitGlue firmware to integrate their application logic and compile and flash it to the CircuitGlue board. Future research can investigate how to make such features also available to novices.

To facilitate writing application logic directly into the CircuitGlue firmware, we provide a basic template using the “setup” and “loop” constructs used by Arduino. In the future, we plan on adding support for the Arduino

programming language to allow the use of Arduino libraries, similar to PlatformIO[21]. To enable or disable features provided by the CircuitGlue firmware, such as programmable voltages or translation to Jacdac, users can modify a header file containing compile-time configuration options.

9 TECHNICAL BENCHMARK

In this section we report on CircuitGlue's throughput, latency, analog reading accuracy, output voltage accuracy, and PWM signal characteristics, to help the reader understand the capabilities and characteristics of CircuitGlue and its impact when used in a circuit design. All characteristics, except throughput and PWM characteristics, were benchmarked using the Saleae Logic Pro 8 logic analyzer and averaged over ten trials for consistency. The circuit design for each of the programmable header pins is identical, so the reported characteristics below are the same for every pin. We present six separate requirements; if a module or component fits all these requirements, it is compatible with CircuitGlue.

(1) **Voltage range**: CircuitGlue supports a programmable voltage range of 1.8 to 12V with 0.1V resolution. The breakout board or component should operate within this voltage range to be compatible with CircuitGlue. We benchmarked the output voltage of the programmable power supply used when the programmable pin is configured as Power. Figure 19D compares the requested voltages with the voltage measured at the programmable header pins. As demonstrated by the results, CircuitGlue can accurately output a requested voltage within a few millivolts. As the voltage is programmable from 1.8 to 12V in steps of 0.1V (Section 6.2), CircuitGlue is compatible with most common components and modules.
(2) **Current rating**: CircuitGlue supports currents up to 3 amps. Breakout boards or components should require at most this much current to be compatible with CircuitGlue.
(3) **Digital communication interfaces**: Similar to other popular development boards such as Arduino [5] and micro:bit [26], CircuitGlue supports common digital communication protocols such as I2C, SPI, and UART, making it easy to interface with a wide variety of breakout boards and modules.
The connection between the microcontroller and digital programmable header pin is made via a logic level converter and an analog switch. The parasitic capacitance of the switch is up to 35pF when 'closed', but more significantly data throughput will be limited by the maximum operating speed of the TXS0104 logic level converter, reported in the datasheet as 24Mbps when used as push-pull and 2Mbps when open-drain (see also Section 6.3.1).
To verify the latency introduced by CircuitGlue in practice, we benchmarked the delay between pulling a programmable header pin high and reading the high voltage at the microcontroller. The test results are shown in Figure 19A. As the latency is only a handful of nanoseconds, CircuitGlue will not noticeably impact the overall performance of most prototypes.
The DC input and output impedance and leakage current during digital communications is dominated by the characteristics of the CircuitGlue microcontroller.
(4) **Analog communication interfaces**: CircuitGlue supports modules that communicate over analog signals of up to 12V. We measured the accuracy of analog readings by configuring a channel of our logic analyzer as an analog input, and connected it to a CircuitGlue programmable header pin configured as analog input. We then applied a voltage to this pin and compared measurements from the logic analyzer with those measured by the CircuitGlue microcontroller. We conducted these measurements twice, once with our voltage divider enabled (allowing input between 0 and 12V), and once with it disabled (allowing input between 0 and 3.3V). Results are shown in Figure 19B-C. We note that readings differ only by a few millivolts.

[21] https://platformio.org/

In addition to analog voltages, CircuitGlue also supports modules that use a variable resistance as output, such as a temperature or light sensor. As described in Section 6.3.2, CircuitGlue has an internal pull-down resistor which can be used to measure the resistance of a sensor on the module.
As with digital communications, the DC input and output impedance and leakage current during analog communications is dominated by the characteristics of the CircuitGlue microcontroller.

(5) **Pulse width modulation**: CircuitGlue can generate a pulse width modulation (PWM) signal in two ways: through a microcontroller-generated PWM signal or by rapidly switching between the Power and Ground functions of a CircuitGlue programmable pin. The best method to use depends on the requirements of the component or module being driven. In general, microcontroller-generated PWM signals are preferred for their flexibility and precision, while Power and Ground switching is used when non-standard voltages or higher currents are required. For example, an electronic speed controller (ESC) module will work well with a microcontroller-generated PWM signal to set the speed of a brushless motor, while RGB LEDs and DC motors use the Power and Ground functions as they require different voltages and/or currents.
When a microcontroller-generated PWM signal is used, the frequency and duty cycle is directly controlled by the nRF52840 microcontroller. CircuitGlue supports PWM frequencies ranging up to 16MHz, and the duty cycle can be adjusted with a resolution of 8 bits. As the logic level converter (Section 6.3.1) can change its state in less then 10ns (5ns ±3ns), it doesn't limit the frequency of the PWM signal. On the other hand, if the Power and Ground functions are used to control the PWM signal, the frequency is limited by the characteristics of the electronic components used to switch Ground and Power (Section 6.3.3 and Section 6.3.4). According to the datasheets of the relevant MOSFETs, shift register and decoder ICs (Section 6.3), switching to from Ground to Power takes around 40ns, while switching from Power to Ground takes 120nS. To provide adequate time for charging and discharging, we limit the PWM frequency to 1MHz.

(6) **Physical interface**: As described in Section 4.2, CircuitGlue uses a physical interface consisting of a single row of eight header pins that most common breakout boards can directly plug into. However, there may be situations where a module's pinout does not match this configuration, such as modules that expose two rows of pins or have a different style of connector. In such cases, an adapter board or a small breadboard can be used to connect the module to CircuitGlue.

(A) GPIO latency

logic level	latency	SD (n=10)	unit
3.3V	5,2	2,71	ns
5V	5	2,86	ns

(B) Analog accuracy (0-12V)

input	measured	SD (n=10)	unit
2,964	2,975	0,009	V
4,989	5,037	0,008	V
12,26	12,375	0,016	V

(C) Analog accuracy (0-3.3V)

input	measured	SD (n=10)	unit
1,789	1,781	0,005	V
2,959	2,969	0,003	V

(D) VCC accuracy

requested	output	SD (n=10)	unit
2,000	2,002	0,009	V
3,000	2,995	0,014	V
4,000	3,996	0,009	V
5,000	4,999	0,020	V
6,000	5,995	0,022	V
7,000	7,014	0,030	V
8,000	7,996	0,011	V
9,000	8,998	0,014	V
10,000	10,004	0,030	V
11,000	10,998	0,025	V
12,000	11,992	0,013	V

Fig. 19. Results of the technical evaluation of CircuitGlue.

10 PRELIMINARY USER EVALUATION

CircuitGlue combines the versatility and extensibility of Type 2 breakout boards with the ease of use of Type 3 integrated modular systems. To collect users' feedback and better understand the utility of CircuitGlue for electronic novices when building interactive prototypes, we conducted a preliminary user evaluation. This user evaluation aimed to better understand the differences when prototyping an electronic system using CircuitGlue versus a traditional approach, in which Type 2 breakout boards are wired to a development board using breadboards and jumper wires.

10.1 Participants

We recruited six participants from our research institution, all aged between 25 and 36. Before recruitment, we assessed candidates' experience level by asking them to elaborate on previous projects they had built. As CircuitGlue is specifically designed for novices, we recruited five participants who self-claimed having a rough to basic understanding of electronics prototyping and one participant (P5) with no prior experience building electronic circuits. Participants with a basic knowledge of electronics had only participated in small projects using cheap and common components such as LEDs, buttons and occasionally breakout boards. The study started with a short interview to further assess the participants' background and familiarity with prototyping workflows.

10.2 Procedure

The study design consists of two conditions, which we refer to as *CIRCUITGLUE* and *BREADBOARD*. In both conditions, participants were asked to prototype the smart desktop fan prototype, introduced in the Walkthrough (Section 2). Both conditions used the BBC micro:bit as development board, a temperature sensor breakout board (KY-015), a PIR motion sensor breakout board (HC-SR501) and a 12V PC fan consisting of a DC motor and built-in protection circuit. The temperature sensor and PIR motion sensor communicated using digital GPIO signals and required 3.3V and 5V, respectively, while the PC fan required a 12V PWM signal. In the *CIRCUITGLUE* condition, participants had access to three CircuitGlue boards with Jacdac cables for interconnecting them. Additionally, we handed participants a one-page printed guide displaying all three modules and their component names to ease identification. In the *BREADBOARD* condition, participants had access to breadboards, jumper wires, a DC-DC converter module (LM2596), and a DC motor driver (TB6612). Additionally, we gave participants access to a printout of the breadboard circuit diagram (Appendix B) for wiring the smart desktop fan. The study followed a within-subjects design where the order of conditions was alternated between participants.

Within the focus of this study, we concentrated on comparing the mechanics of building an interactive prototype through CircuitGlue versus a standard breadboard with additional conversion modules and associated wiring. We therefore considered the writing of application logic and the creation of translation code for the modules in both conditions to be out of scope. We believe this is the basis of a fair evaluation, because we are focused on the electronics aspects of prototyping, not the ease of writing code—which would be similar when using microcontroller boards like micro:bit and Arduino without the help of CircuitGlue

We preconfigured the BBC micro:bit with the necessary application logic written in MakeCode[4] and ensured all three modules were supported in both the *CIRCUITGLUE* and *BREADBOARD* conditions. The application logic for the *CIRCUITGLUE* condition uses Jacdac blocks to interact with the high-level Jacdac services announced by the temperature sensor, PIR motion sensor, and PC fan. In the *BREADBOARD* condition, the application logic uses default GPIO functions to interact with the PIR motion sensor and DC motor driver. In addition, an external library is used to drive the temperature sensor.

For both conditions, participants received an introduction during which we demonstrated how to connect the PIR motion sensor, which they then had to replicate. We asked participants to think out loud throughout the

study and describe their process. We concluded the evaluation with an interview asking participants about their experience with both methods. On average, the user evaluation lasted for 1 hour.

10.3 Results

While the results presented in this section are still very preliminary, they already show the usability of CircuitGlue for novices when building electronic prototypes.

On average, participants took 15 minutes (SD=2.6) to complete the breadboard prototype and 5.2 minutes (SD=1.1) when using CircuitGlue. In the *BREADBOARD* condition, participants expressed being concerned about damaging components by accidentally making faulty connections. All participants described their breadboard prototypes as a mess of wiring and did not feel confident about powering the prototype on before we verified it with them. Indeed, we noticed two participants made a mistake while wiring; after they thought their prototype was finished, we asked some follow-up questions about the connections wires to guide them to their mistake and self-correct it. We did not include this additional time in the measurements. In the *CRICUITGLUE* condition, P1, P2, P3, and P5 told us they were not very confident that the board had been configured correctly by the CircuitGlue software, and would have been reassured if they could see details of the current configuration. This contrasts with P4 and P6 who were under the impression that CircuitGlue would automatically verify important parameters and thereby prevent faults if they made a mistake during configuration.

During the post-study interview, we asked participants for each condition how comfortable and confident they felt when connecting modules. All participants reported being more comfortable and confident in the *CIRCUITGLUE* condition. All participants, however, had initial concerns about the size and additional cost that comes with CircuitGlue, especially for simple prototypes. When we explained that only a single CircuitGlue board is strictly necessary because CircuitGlue's Diagram Generator can be used after each component is tested and operational (Section 8.3), all participants agreed that this prototyping style would be practical.

As we gave participants a printout of the exact circuit diagram in the *BREADBOARD* condition to accommodate for their limited electronics expertise, this condition is actually easier than it would likely be in practice. We therefore asked participants if they think they would be able to build the prototype in the *BREADBOARD* condition also without the diagram. All participants thought this would be feasible given enough time and access to online resources, although P4, P5 and P6 added that they would only consider starting such a project if absolutely needed.

We finally asked participants which method they would prefer and which they would potentially adopt when prototyping in the future. All participants preferred CircuitGlue to quickly test different electronic modules before starting to prototype. While P4 and P5 would prefer CircuitGlue for any prototype, P1, P2, P3, and P6 reported that this would depend on the specifics of the prototype such as complexity of modules, application area and whether the purpose of the prototype is experimentation or building a specific device. P6 would only use CircuitGlue when boards are at hand. In addition, P6 also questioned if both methods could be used together and saw potential in a hybrid approach to offload complex components to CircuitGlue while still wiring simple components manually, similar to prototyping styles covered in Section 8. P5 especially liked that CircuitGlue allows for only concentrating on I/O components and does not require additional components for conversions.

We did not notice any effect on the order of conditions. Although both conditions instructed participants to build the same prototype, the process used for interconnecting components is different and thus requires different knowledge. While the *BREADBOARD* condition required users to interconnect modules using individual jumper wires according to a diagram, the *CIRCUITGLUE* condition involved plugging in components on a female header and selecting items from a drop-down menu.

11 DISCUSSION AND FUTURE WORK

As argued throughout this paper, we believe CircuitGlue combines the ease of use of integrated modular systems, known as Type 3 electronics [21], with the versatility and flexibility of breakout boards and modules, known as Type 2 electronics [21]. Although components and modules need to be configured before they can be used, we imagine an online repository of configurations that are re-used subsequently. CircuitGlue does not lock users into either the CircuitGlue or Jacdac eco-system as modules from any supplier can be used, overcoming a key disadvantage of modular electronics toolkits as previously reported in the literature. Additionally, the circuit diagram generator (Section 8.3) allows CircuitGlue boards to be removed at any time in favor of transitioning to a standard breadboard prototyping style. In this way CircuitGlue lowers the floor for making electronic system prototypes [34] without limiting what can be built and without increasing the ultimate cost and complexity of the final implementation. The initial experience of users in our preliminary evaluation indicates that CircuitGlue does indeed simplify working with electronics for beginners, speeding up the process nearly threefold and reducing errors.

Large-scale prototypes, involving many modules, are supported by our bus architecture but would require many CircuitGlue boards. However, even users who only have a single CircuitGlue board can build large prototypes with by iteratively moving from a CircuitGlue-based interface to a custom breadboard circuit replacement once a component or module is tested and operational (Section 8.2).

Modules and components are instantly operational with CircuitGlue and do not require users to understand all the characteristics or workings first. Although our approach initially hides aspects of the electronics in the 'black box' formed by the CircuitGlue board, we believe it offers opportunities for a new top-down learning experience, in which students first build prototypes they are highly interested in and later discover and learn about lower level details by conversion to a breadboard circuit (Section 8.3). Future versions of CircuitGlue could incorporate display elements like multi-color LEDs or an LCD screen next to the header pins to reassure users about the current configuration while also offering a learning experience. It would also be interesting to explore the potential benefits of a wireless CircuitGlue board which is battery-powered and exposes the module(s) or component(s) connected to its header pins over a wireless implementation of Jacdac based on the nRF52840 SoC's built-in radio.

Our experience with CircuitGlue and our technical evaluation of its performance both indicate that it is compatible with a large number of existing modules, as is likely future modules too. Nonetheless, future versions could support higher voltages, e.g. up to 24V, for driving larger inductive coils[22]. Another useful feature would be a constant current source to offer control over the current on universal header pins. Constant current enables driving components such as LEDs without a resistor and supports components and protocols that use the "4 to 20mA current loop", which is popular in industry. An digital-to-analog converter (DAC) would enable fine-tuned output voltages, used for audio processing for example. CircuitGlue could also be upgraded with additional programmable header pins; while most common electronic modules require less than eight pins to function, some components, such as parallel displays[23], require more.

While the CircuitGlue board has over-voltage and over-current protection sensors, the speed and efficacy of detection and mitigation depend on software. In our experience many components can tolerate voltage and current spikes long enough for the SoC to respond; however, incorporating additional hardware components that react faster, such as programmable fuses and back EMF protection diodes, could provide further protection to the CircuitGlue board and connected modules. Additionally, a thorough technical evaluation of the CircuitGlue board, including measurements of analog crosstalk, slew rate and characteristic impedances across different frequencies, could be insightful.

[22] https://www.adafruit.com/product/5141
[23] https://www.digikey.com/en/products/detail/futaba-corporation-of-america/ELF1101AA/14669494

Finally, we note that the CircuitGlue board itself could be made more compact by moving to a design with components on both sides of the circuit board, and by using smaller IC packages where available.

12 CONCLUSION

In this paper, we contribute CircuitGlue, an electronic converter that automates the process of interconnecting heterogeneous electronic components and modules by ensuring voltage levels, interface types, communication protocols and pinouts are compatible. Each pin in the programmable header of the CircuitGlue board can support either power, analog signals, or digital communication. We demonstrated several new prototyping styles created by CircuitGlue and evaluated its usability by conducting a preliminary user study with six participants. Their responses indicate that CircuitGlue significantly speeds up building electronic prototypes and is likely to accrue the greatest benefits in educational settings and for makers who quickly want to prototype something. We hope that others can build on the concepts and the implementation details we have shared in this paper to further evaluate and explore the design of CircuitGlue and its application in different domains and with different user bases.

ACKNOWLEDGMENTS

We would like to thank Danny Leen for his valuable input, proofreading and improving our figures. We also would like to thank all participants for their valuable insights in CircuitGlue.

REFERENCES

[1] Adafruit. 2022. Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055. Retrieved March 27, 2022 from https://www.adafruit.com/product/2472

[2] Ephrem Ryan Alphonsus and Mohammad Omar Abdullah. 2016. A review on the applications of programmable logic controllers (PLCs). *Renewable and Sustainable Energy Reviews* 60 (2016), 1185–1205. https://doi.org/10.1016/j.rser.2016.01.025

[3] Anadigm. 2022. Anadigm FPAA. Retrieved April 6, 2022 from https://www.anadigm.com/fpaa.asp

[4] 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. https://doi.org/10.1145/3126594.3126637

[5] Arduino. 2022. Arduino - Home. Retrieved March 27, 2022 from https://www.arduino.cc/

[6] Ayah Bdeir. 2009. Electronics as Material: LittleBits. 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, 397–400. https://doi.org/10.1145/1517664.1517743

[7] Paulo Blikstein et al. 2015. Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. *Found. Trends Hum. Comput. Interact.* 9, 1 (2015), 1–68.

[8] Circuito.io. 2022. Design Your Circuit with Circuito.io. Retrieved August 1, 2022 from https://www.circuito.io/

[9] James Devine, Michal Moskal, Peli de Halleux, Thomas Ball, Steve Hodges, Gabriele D'Amone, David Gakure, Joe Finney, Lorraine Underwood, Kobi Hartley, Paul Kos, and Matt Oppenheim. 2022. Plug-and-play Physical Computing with Jacdac. In *ACM Interact. Mob. Wearable Ubiquitous Technol.* Association for Computing Machinery, New York, NY, USA, 30 pages. https://doi.org/10.1145/3550317

[10] 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. https://doi.org/10.1145/2984511.2984566

[11] Espressif. 2022. A cost-effective and highly integrated Wi-Fi MCU for IoT applications. Retrieved March 27, 2022 from https://www.espressif.com/en/products/socs/esp8266

[12] Espressif. 2022. ESP32. Retrieved April 6, 2022 from https://www.espressif.com/en/products/socs/esp32

[13] Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (Orlando, Florida, 2001-11-11) *(UIST '01)*. Association for Computing Machinery, New York, NY, USA, 209–218. https://doi.org/10.1145/502348.502388

[14] Steve Hodges, James Scott, Sue Sentance, Colin Miller, Nicolas Villar, Scarlet Schwiderski-Grosche, Kerry Hammil, and Steven Johnston. 2013. .NET Gadgeteer: A New Platform for K-12 Computer Science Education. In *Proceeding of the 44th ACM Technical Symposium on*

Computer Science Education (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 391–396. https://doi.org/10.1145/2445196.2445315

[15] Steve Hodges, Nicolas Villar, James Scott, and Albrecht Schmidt. 2012. A New Era for Ubicomp Development. *IEEE Pervasive Computing* 11, 1 (2012), 5–9. https://doi.org/10.1109/MPRV.2012.1

[16] Infineon. 2022. 32-bit PSoC™ Arm® Cortex® Microcontroller. Retrieved April 6, 2022 from https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/

[17] 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 (sep 2020), 21 pages. https://doi.org/10.1145/3369838

[18] 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. https://doi.org/10.1145/3290605.3300407

[19] 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.* Association for Computing Machinery, New York, NY, USA, 987–998. https://doi.org/10.1145/3379337.3415887

[20] SAM Labs. 2022. SAM Labs homepage. Retrieved April 6, 2022 from https://samlabs.com/us/

[21] Mannu Lambrichts, Raf Ramakers, Steve Hodges, Sven Coppers, and James Devine. 2021. A Survey and Taxonomy of Electronics Toolkits for Interactive and Ubiquitous Device Prototyping. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2, Article 70 (jun 2021), 24 pages. https://doi.org/10.1145/3463523

[22] Mannu Lambrichts, Jose Maria Tijerina, and Raf Ramakers. 2020. SoftMod: A Soft Modular Plug-and-Play Kit for Prototyping Electronic Systems. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Sydney NSW, Australia, 2020-02-09) *(TEI '20)*. Association for Computing Machinery, New York, NY, USA, 287–298. https://doi.org/10.1145/3374920.3374950

[23] 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. https://doi.org/10.1145/3430524.3440623

[24] Lego. 2022. Lego Mindstoms EV3. Retrieved March 27, 2022 from https://www.lego.com/en-us/product/lego-mindstorms-ev3-31313

[25] David A. Mellis, Leah Buechley, Mitchel Resnick, and Björn Hartmann. 2016. Engaging Amateurs in the Design, Fabrication, and Assembly of Electronic Devices. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1270–1281. https://doi.org/10.1145/2901790.2901833

[26] BBC micro:bit. 2022. Micro:bit Educational Foundation. Retrieved March 27, 2022 from https://microbit.org/

[27] Raspberry Pi. 2022. RP2040. Retrieved April 6, 2022 from https://www.raspberrypi.com/products/rp2040/

[28] Raspberry Pi. 2022. Teach, Learn, and Make with Raspberry Pi. Retrieved March 27, 2022 from https://www.raspberrypi.org/

[29] Raspberry Pi. 2022. What is PIO? Retrieved April 6, 2022 from https://www.raspberrypi.com/news/what-is-pio/

[30] Raf Ramakers, Kashyap Todi, and Kris Luyten. 2015. PaperPulse: An Integrated Approach to Fabricating Interactive Paper. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 267–270. https://doi.org/10.1145/2702613.2725430

[31] David Romano. 2022. A Brief History of FPGA. Retrieved April 6, 2022 from https://makezine.com/2019/10/11/a-brief-history-of-fpga/

[32] Nordic Semiconductor. 2022. nRF52 Series. Retrieved April 6, 2022 from https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_nrf52%2Fstruct%2Fnrf52.html

[33] Nordic Semiconductor. 2022. PPI — Programmable peripheral interconnect. Retrieved April 6, 2022 from https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Fppi.html

[34] Ben Shneiderman, Gerhard Fischer, Mary Czerwinski, Mitch Resnick, Brad Myers, Linda Candy, Ernest Edmonds, Mike Eisenberg, Elisa Giaccardi, Tom Hewett, Pamela Jennings, Bill Kules, Kumiyo Nakakoji, Jay Nunamaker, Randy Pausch, Ted Selker, Elisabeth Sylvan, and Michael Terry. 2006. Creativity Support Tools: Report From a U.S. National Science Foundation Sponsored Workshop. *International Journal of Human–Computer Interaction* 20, 2 (2006), 61–77. https://doi.org/10.1207/s15327590ijhc2002_1 arXiv:https://doi.org/10.1207/s15327590ijhc2002_1

[35] 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. https://doi.org/10.1145/3126594.3126618

[36] 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. https://doi.org/10.1145/2984511.2984527

[37] 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. https://doi.org/10.1145/3332165.3347938

[38] 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. https://doi.org/10.1145/3126594.3126646

[39] R. Świerczyński, K. Urbański, and A. Wymysłowski. 2014. Methodology for supporting electronic system prototyping through semiautomatic component selection. In *2014 15th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*. Institute of Electrical and Electronics Engineers, New York, NY, USA, 1–4. https://doi.org/10.1109/EuroSimE.2014.6813792

A EXAMPLE OF TRANSLATION CODE

```
#include "Arduino.h"
#include "DHT.h"


DHT dht(1, DHT11);

uint32_t last_sample = 0;
uint32_t last_sample_interval = 0;

void initializer(void) {
    // start the temperature sensor
    dht.begin();
}

void updater(void) {
    // sample the temperature every two seconds
    if (jd_should_sample_delay(&last_sample_interval, 2000))
        last_sample = JD_FLOAT_TO_I22_10(dht.readTemperature());
}

uint32_t get_temperature(void) {
    // returns the last sample
    return last_sample;
}


const env_sensor_api_t custom_temperature_api = {
    .init = initializer,
    .process = updater,
    .get_reading = get_temperature,
};

void setup(void) {
    // initialize the temperature service providing our custom api
    temperature_init(&custom_temperature_api);
}

```

Fig. 20. Example of the translation code written to support the DHT11 temperature sensor module.

B CIRCUIT DIAGRAM USED FOR THE PRELIMINARY USER EVALUATION

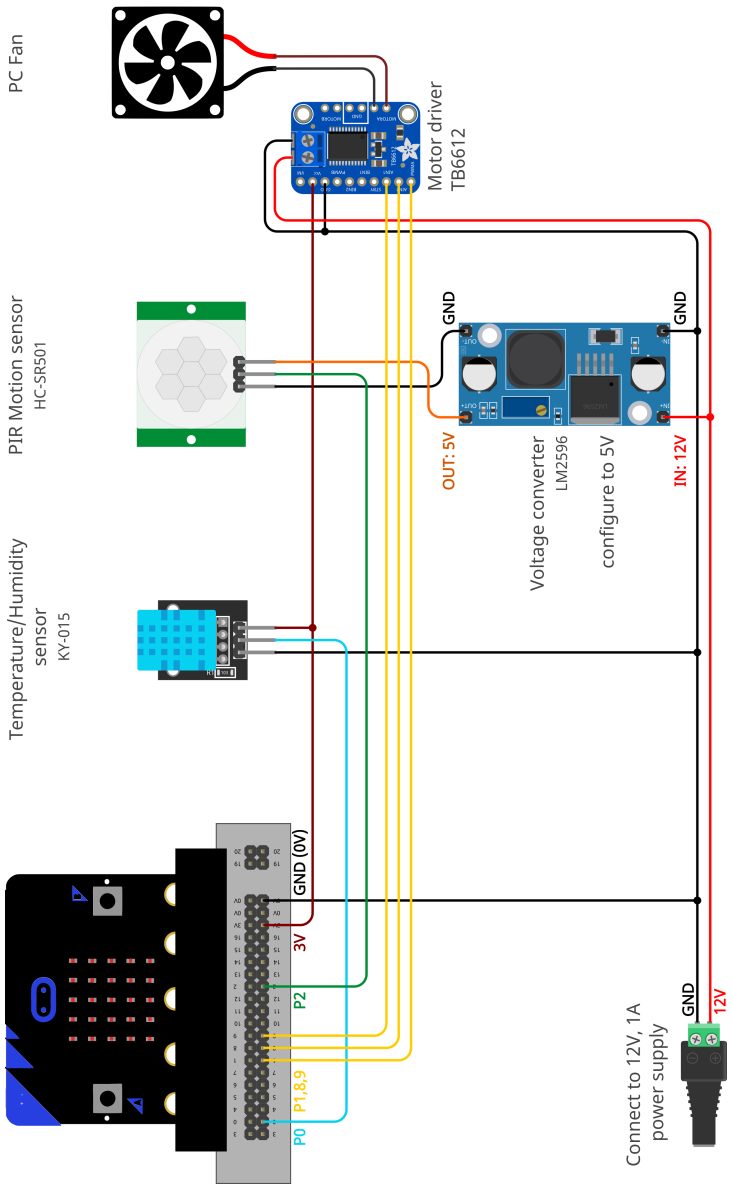

Fig. 21. Breadboard diagram demonstrating how participants of the user evaluation should interconnect all electronic modules.