ActuBoard: An Open Rapid Prototyping Platform to integrate Hardware Actuators in Remote Applications

Sebastian Günther
guenther@tk.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Florian Müller
mueller@tk.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Felix Hübner
huebner@tk.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Max Mühlhäuser
max@informatik.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Andrii Matviienko
matviienko@tk.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

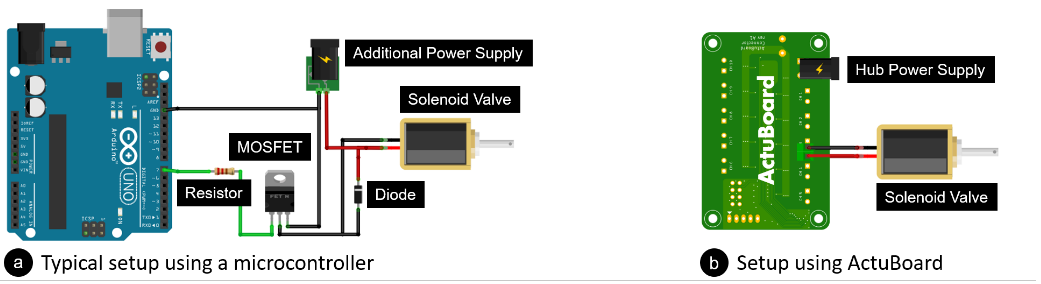

Figure 1: Example comparison of how to connect a solenoid valve a) using a typical method using an Arduino and necessary additional electronic components, and b) using our ActuBoard platform with the valve directly plugged into the Hub.

ABSTRACT

Prototyping is an essential step in developing tangible experiences and novel devices, ranging from haptic feedback to wearables. However, prototyping of actuated devices nowadays often requires repetitive and time-consuming steps, such as wiring, soldering, and programming basic communication, before HCI researchers and designers can focus on their primary interest: designing interaction. In this paper, we present ActuBoard, a prototyping platform to support 1) quick assembly, 2) less preparation work, and 3) the inclusion of non-tech-savvy users. With ActuBoard, users are not required to create complex circuitry, write a single line of firmware, or implementing communication protocols. Acknowledging existing systems, our platform combines the flexibility of low-level microcontrollers and ease-of-use of abstracted tinker platforms to control actuators from separate applications. As further contribution, we highlight the technical specifications and published the ActuBoard platform as Open Source.

CCS CONCEPTS

• **Human-centered computing** → **Interaction devices**; Haptic devices; • **Hardware** → *Communication hardware, interfaces and storage.*

KEYWORDS

haptics, actuators, rapid prototyping, tinkering, virtual reality, hardware, open source

ACM Reference Format:
Sebastian Günther, Florian Müller, Felix Hübner, Max Mühlhäuser, and Andrii Matviienko. 2021. ActuBoard: An Open Rapid Prototyping Platform to integrate Hardware Actuators in Remote Applications. In *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '21 Companion), June 8–11, 2021, Virtual Event, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3459926.3464757

1 INTRODUCTION

Prototyping in the field of Human-Computer interaction (HCI) is essential across a large spectrum of technologies where projects related to tangibles, IoT, wearables, or haptic feedback, have gained incredible momentum. Although the technology used behind these topics has advanced due to the interest of different domains within the community, its hardware-oriented nature continues to require basic knowledge of electrical engineering [5, 23].

Typically, today's prototyping starts by identifying the technical requirements, e.g., the voltage or resistors needed, followed by assembling each component on a breadboard or soldering parts

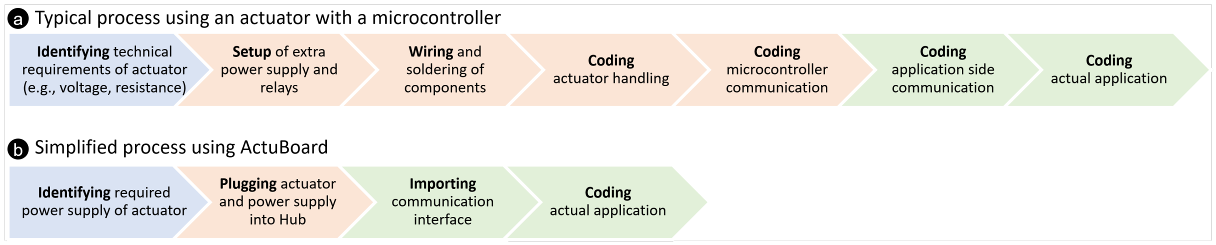
a Typical process using an actuator with a microcontroller
Identifying technical requirements of actuator (e.g., voltage, resistance)
Setup of extra power supply and relays
Wiring and soldering of components
Coding actuator handling
Coding microcontroller communication
Coding application side communication
Coding actual application
b Simplified process using ActuBoard
Identifying required power supply of actuator
Plugging actuator and power supply into Hub
Importing communication interface
Coding actual application

Figure 2: Comparison of the typical process of how to connect an actuator to a microcontroller until it is usable within a remote application. a) The typical workflow involves much more manual wiring, setup, and writing fundamental code compared to b) using an actuator with ActuBoard. Colors depict the preparation (blue), microcontroller (orange), and application (green) phases.

(Fig. 2 a). However, imagine a designer who only wants to investigate new interaction concepts and wants a basic prototype. While the designer may have done similar projects with a similar setup before, a new project often cannot fully reuse existing hard- and software. Hence, even similar actuators have to be controlled slightly differently and require to start again with an overhead of configuration, boilerplate code, and working on circuitry. While this has been eased by the emergence of tinker platforms, such as Arduino [16] or Teensy[1], tool-assisted circuitry designers [1, 21, 37], and DIY websites providing schematics, even experienced tinkers still need to perform those repetitive steps before they can use their prototype.

To address such circumstances, a large number of other prototyping platforms arose that reduce those error-prone processes, such as the Arduino Grove [2] or Microsoft's Gadgeteer [33]. However, while those are perfect for carrying out hardware tinkering, they rely on preconfigured modules rather than off-the-shelf components, or require breakout modules with similar complexity and repeating assembly steps as low-level platforms. And yet, even when all these hardware preparations are done, users still have to program low-level firmware or repetitive boilerplate code to control the hardware components from external applications, such as a VR-based application for haptic actuation. This includes the handling and addressing of the actuators on the microcontroller, as well as the communication interface between the prototype and the remote application until they have a smooth interplay. Prototyping is undoubtedly fun. However, these error-prone steps challenge inexperienced tinkers or students who may not come from hardware-related domains and even add up to a time-consuming effort for experienced users.

In this paper, we present ActuBoard, a prototyping platform to minimize initial and repetitive steps to support developers focusing on designing interaction. Acknowledging traditional approaches, ActuBoard combines the flexibility of low-level hardware platforms for off-the-shelf hardware components with the simplicity and convenience of high-level tinker toolkits. Our platform contributes 1) a quick assembly and addressing of off-the-shelf components, 2) a communication interface for external applications, and 3) the inclusion of non-tech-savvy users. Thereby, actuators can be used through a plug-and-play approach with off-the-shelf components and a fitting power supply that only need to be plugged into ActuBoard. Further, the addressing of them is done automatically without writing microcontroller firmware, allowing a user to focus on building high-level applications and investigating interaction concepts (Fig. 2 b). Summarizing, ActuBoard supports the following features:

Flexible: A high versatility for off-the-shelf components without limiting to pre-defined or proxy modules.

Plug-And-Play: Actuators with up to 24 V supported. No need to write a single line of firmware code on the microcontroller as ActuBoard takes care of every hardware-related addressing.

Application Support: ActuBoard provides an easy-to-use serial and C# interface for direct application communication.

Inclusion of non-tech-savvy users: Less preparation work and electrical engineering knowledge are necessary. Breadboarding or soldering of circuitry is not required.

Open Source: We released ActuBoard as Open Source[3].

1.1 Example scenario: Remotely watering a plant

Alice wants to automatically water her plant before going on vacation. Therefore, she designed an application on her PC that receives weather information and a valve that releases water for her plant connected to a microcontroller. With a traditional approach (Fig. 1a), she has to check the specifications of the valve, program the microcontroller, and add a power supply, transistor, and resistor for the valve. Also, a diode is useful to ensure that the components are protected. However, this requires a lot of wiring, soldering, and most importantly: time. Yet, the complete software for communication between the microcontroller and her application is still missing.

In contrast, when using ActuBoard (Fig. 1b), Alice only needs to find an appropriate valve and power supply, plug them into the ActuBoard, and import the communication interface in her application. Project done. Thus, no need to focus on hardware details while remaining in full control of her off-the-shelf components. Even if she wants to water more plants, she only needs to plug more valves into the ActuBoard instead of repeating the time-consuming circuitry.

2 REQUIREMENTS

Working with actuators often challenges to understand the underlying concepts, software, and hardware. However, we think most

[1]https://www.pjrc.com/teensy/, last accessed 20th May, 2021
[2]https://seeeddoc.github.io/Grove-Base_shield_v2/, last accessed 20th May, 2021
[3]https://git.tk.informatik.tu-darmstadt.de/sebastian.guenther/actuboard-public, last accessed 20th May, 2021

Table 1: Comparison of ActuBoard to a selection of commonly used hardware platforms.

Requirement	ActuBoard	Arduino	ESP32	Grove	Gadgeteer	Raspberry PI
R1: Off-the-Shelf components	✓	✓	✓	1	1	✓
R2: Plug-And-Play	✓	-	-	✓	✓	-
R3: No Firmware Coding	✓	-	-	-	✓	-
R4: Communication Interface	✓	-	-	-	✓	✓
R5: Small, Mobile, Wireless	✓	✓	✓	✓	✓	✓
R6: Low Cost and Affordable	2	✓	✓	3	-	✓
R7: Debugging	✓	✓	✓	✓	✓	✓

1 With breakout modules (but then conflicting R2). **2** No mass production, hence higher initial costs. **3** Pre-configured modules tend to be more expensive.

hardware-based prototypes often require similar steps, especially when wanting to communicate between a physical prototype and a separate application. Therefore, before we look into existing approaches, we first discuss a set of requirements for our prototyping platform based on past experiences and related work (e.g., [4–6, 22–24, 29, 32, 33]).

R1 Off-the-Shelf Components The first requirement is to support a large number of off-the-shelf actuators as well as comprehensive support for many differing specifications. Especially when working with components for haptic actuation, the designer should not be limited by pre-defined modules. This also includes actuators that need an additional power supply beyond the typical 3.3 or 5 V.

R2 Plug-and-Play Simplicity to free the developer from hardware details is desirable. For this, actuators should be able to be connected plug-and-play. Minimal hardware knowledge should be necessary, as well as minimized time spent on hardware circuitry. Developers should also have not to care about the protection of the circuitry, the microcontroller, and other components in the event of unexpected failures.

R3 No Firmware Coding The amount of firmware coding and setup should be minimal or not necessary to run plug-and-play components to reduce hardware-close implementation effort.

R4 Communication Interface An easy communication interface should exist that a developer does not need to write communication interfaces on the controller itself. At best, this is done abstracted, platform-independent, and flexible.

R5 Small, Mobile, Wireless A subsidiary requirement is a platform that is as compact, lightweight, and mobile as possible. This means that prototypes should not become superfluously large while at the same time keeping the hardware in the background - making it amenable to mobile and wearable applications. For mobile applications, a wireless connection, either native or through modules, is beneficial.

R6 Low Cost and Affordable Prototypes are far from commercially available products and tend to have various iterations throughout their development. Nevertheless, it should be cost-efficient in case of broken parts due to experimentation or unexpected failures.

R7 Debugging Although using the platform is supposed to be as straightforward as possible, a further requirement is a possibility for debugging as prototyping always involves an initial testing phase of the actuators, or individual components may become malfunctioning over time.

3 RELATED WORK AND PLATFORMS

In the following, we give an overview of existing prototyping platforms and prototyping methods in related research.

3.1 Prototyping Platforms

Due to the proliferation of a variety of tinkering and prototyping platforms over the last decades, particularly accelerated by the Arduino ecosystem and research (e.g., [19]), two main types of platforms have emerged: 1) low-level, and 2) component-based platforms.

Low-level platforms provide high flexibility, such as the popular Arduino, its forks, and derivates. Similar, other platforms appeared from small microcontrollers with accessible hardware interfaces, such as the ESP family[4], Teensy, or the Photon[5], to educational-focused platforms, such as Micro:bit[6] [3, 30] or TinkerForge[7].

Component-based or module-based systems provide a mix between flexibility and simplicity of pre-defined plug-and-play components, such as Grove or Gadgeteer [33]. However, while this makes prototyping easier, pre-defined components may not always provide a desired functionality or tend to be expensive. Available breakout modules are possible, but those require the same complex circuitry as low-level platforms. Further, these platforms focus on working on a microcontroller basis while controlling it from a remote application still requires the coding of a communication interface. Single-board computers, e.g., Raspberry Pi[8], or BeagleBoard[9], provide direct interfaces for I/O components within their operating systems, however, tend to be too powerful for most prototypes or are still not powerful enough to run demanding applications directly.

The listed examples are just a small subset, however, typically with trade-offs between flexibility and complexity. More flexible platforms, like Arduino, inevitably introduce more complexity for users. Less complex platforms, such as Grove, sacrifice flexibility for simpler usage. And yet, both require the coding of the firmware.

[4]https://www.espressif.com/en/products/devkits, last accessed 20th May, 2021
[5]https://docs.particle.io/photon/, last accessed 20th May, 2021
[6]https://microbit.org/, last accessed 20th May, 2021
[7]https://www.tinkerforge.com/, last accessed 20th May, 2021
[8]https://www.raspberrypi.org/, last accessed 20th May, 2021
[9]https://beagleboard.org/, last accessed 20th May, 2021

As such, those trade-offs often pose a hurdle to quickly design systems that do not focus on hardware tinkering but on exploring interaction concepts. Also, while component-based platforms are more accessible through pre-defined modules, they often tend to have a limited area of application, do not support off-the-shelf components without the low-level effort (e.g., using breakout boards), focus on low-voltage components, or only support a set of expensive modules. In contrast, with ActuBoard, we want to address these trade-offs by providing a platform situated in between with high flexibility and reduced complexity. Table 1 compares the presented requirements ActuBoard to a selection of common platforms.

3.2 Prototyping in Research

There exist various prototyping platforms and toolkits in the research community. While some provide virtual prototyping [2, 36], tool-based circuit designers [1, 21, 35], or AR- and vision-based support to faster create prototypes [14, 15], it requires actual hardware and physical actuators when it comes to tangible or haptic feedback. For this, some works use special tangibles or hardware that allow for the exploration of interaction concepts (e.g., [10, 17, 18]) or use proxy modules for low-fidelity prototypes [37]. However, those are limited to the features provided by the devices. To add more flexibility, research also proposed hardware platforms with a pre-defined set of actuators (e.g., [6, 19, 24, 29, 32]) which are similar to the Grove and Gadgeteer platform [33]. While this allows fast prototyping, they are limited to the given components or, again, do not support a plug-and-play approach for off-the-shelf actuators.

Other research aims to ease the software-side control or communication with actuation devices (e.g., [20]). There are several proposed tools to support the actuation of vibrotactile feedback [22, 25, 26], or other haptic feedback [8, 27, 31]. Hereby, those approaches are focusing on the creation of interaction patterns and mapping of actuators with certain events. Thus, they can be useful as an extension of ActuBoard, which focuses on the creation of the prototype itself.

Summarizing, the presented work and platforms provide powerful interfaces. However, to the best of our knowledge, there is no open platform that supports experienced and novice designers to quickly build and control a prototype using off-the-shelf actuators from remote applications without coding on a microcontroller.

4 ACTUBOARD PLATFORM

The ActuBoard hardware consists of two main components: 1) the ActuBoard Controller, and 2) the ActuBoard Hubs (Fig. 3 a). While the Controller is managing the actuators and serial communication, the Hubs are stackable extension boards that can each address up to ten actuators plus an additional power source that can be different for each Hub. Controller and Hubs interconnect and communicate through I^2C (Fig. 3 b). Further, we published the source code, schematics, and all necessary information to build and use ActuBoard in a public GIT repository[10].

4.1 ActuBoard Controller

The Controller is responsible for managing the actuators and establishing communication with an application. It is based on a standard ESP32 microcontroller with full Arduino compatibility, embedded on the printed circuit board (PCB), as it provides powerful processing capabilities and supports communication via USB, Bluetooth, and WiFi (Fig. 3 b). Using the I^2C interface of the ESP32 as an additional connection bus, the ActuBoard Hubs can be linked. This also includes supply pins, clock data, and an additional output enable channel to (optionally) toggle all actuators simultaneously. The default I/O pins of the ESP remain unassigned and are available for other purposes if necessary.

4.2 ActuBoard Hubs

The ActuBoard Hubs are stackable PCBs providing up to ten actuator ports each. Each Hub contains the full circuitry to address a broad range of actuators. Therefore, we use 16-channel LED drivers with a fixed PWM frequency of 97 kHz and 8-bit resolution (*PCA9635PW*) on each Hub. An additional "*group PWM*" with a frequency of 190 Hz is used to also provide an optional *blinking* pattern for actuators. To switch the actuators, we use high-side MOSFETs with a current of up to 11 A (*CSD17579Q3A*) and connected a diode to protect the hardware from potential inductive loads.

For addressing an actuator, we use the lower 4 bits of the I^2C bus' address pins of each Hub as base ID (*0-11*, settable with jumper pins) combined with the port number of a Hub's connectors (*0-9*). This results in up to 12 addressable Hubs for one ActuBoard Controller, or up to 10×12=120 connected actuators in total. As example, Fig. 3 c depicts an ActuBoard Controller with five attached Hubs. Further, each Hub provides a separate power supply port for actuators with up to 24 V. For a more convenient connection of the actuators and power supply, we use small connector plugs (Fig. 3 d).

4.3 Communication Interface

Communication with external applications is important when a prototype has to react to certain events, e.g. in VR. To reduce coding effort, ActuBoard also provides an easy-to-use communication interface supporting a direct serial communication and an optional C# library with full Unity support, as it is one of the most commonly used engines[11] (*Unity Full .NET 3.5 compatible*). Instruction commands are kept compact with only a few Bytes, but powerful enough to support various functionalities. This includes the setting and reading of single actuators or multiple channels at the same time with identical or varying values (*PWM* enabled). For example, the command for setting a value is 1 Byte for the command type (e.g., *set actuator value*), 2 Bytes command length (necessary when setting multiple channels at once), 2 Bytes for the actuator ID, and 2 Bytes for the value, totaling in 7 Bytes. Though, as handling serial commands is cumbersome, our optional C# library supports convenient wrapper methods. Other commands include a *blinking* for a periodic enable and disable to reduce software side flooding,

[10]https://git.tk.informatik.tu-darmstadt.de/sebastian.guenther/actuboard-public, last accessed 20th May, 2021

[11]based on an official Unity statement https://twitter.com/unity3d/status/1256256504098947080, last accessed 2021-01-10

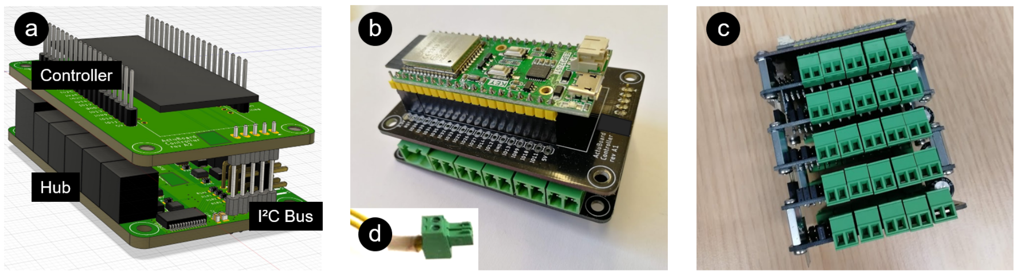

Figure 3: The ActuBoard shown as a) conceptual rendering, b) as final result with one Controller and one attached Hub, and c) with five attached Hubs. For connecting an actuator, we use d) small connector plugs.

a listing of all available Hubs with their IDs and addresses, a *kill-switch* to disable or enable all at once, and a *debug* toggle. A full list of all commands are available in the documentation of the GIT repository.

4.3.1 Debugging. For easy debugging, we provide two approaches. First, a direct serial connection to ActuBoard, and second, a standalone tool based on our C# library that provides a GUI control with an integrated terminal-like output. Both methods allow to directly execute all commands of the software interface.

4.4 Supported Types of Actuators

Currently, we consider two complementary types of actuators: 1) discrete-controlled, and 2) continuously-controlled. Discrete-controlled actuators mostly have only two or a few distinct states. This ranges from the enabled and disabled state of an actuator, e.g., an on/off property of a lamp, to setting discrete states, e.g., different levels of a lift. Continuously-controlled actuators are not limited to binary states but have a continuous range of different states. Those can be typical vibration actuators that are controlled in their frequency, the speed of motors, or the intensity of electrothermal modules.

5 APPLICATIONS USING ACTUBOARD

In the following, we showcase previous projects based on ActuBoard and give ideas for other use-cases.

5.1 Thermal Haptic Feedback

As virtual worlds are getting more realistic, researchers also further improve by applying thermal feedback. Often, this is achieved through Peltier elements (e.g., [7, 28]) that regulate their temperature based on the applied voltage. However, while ActuBoard could operate those elements as well, we investigated how we can use liquids as a medium [12].

As before, ActuBoard supported building the prototype and enabled us to focus on the investigation of the interaction in align with our simplified process (Fig. 2 b). First, we identified the required components and opted for an actuation using liquids that required a warm and cold water supply. While the cold supply directly came from the tap, we had to pre-heat water in a boiler and then pump it through our systems. Further, we needed to control the flow through solenoid valves. As both types of components, the pump, and valves, required 12 V, we could plug all of the components in one ActuBoard Hub, using a single power supply. This allowed us to regulate the flow and temperature of the liquids as we could also actuate the connected pump. In contrast on the application side, we only had to import the C# interface (Sec. 4.3) and were ready to work on our VR environment. Additionally, even though ActuBoard is currently not designed for input electronics, we used the free I/O ports of the embedded ESP32 to read out a flow and temperature sensor.

5.2 Vibrotactile Feedback for VR Sketching

Physical contact with the body is among the most common forms of haptic feedback and often realized through vibrotactile actuation. In this project, Elsayed et al. [9] investigated how vibrotactile feedback can support sketching in VR by embedding vibration motors into a 3D-printed pen. The vibration simulated the friction of a virtual canvas, while the pen was used as a physical proxy object. The vibration motors were directly connected to the ActuBoard and controlled using the Unity library. Depending on the initial settings and the virtually applied pressure of the pen onto the canvas, the VR application used our interface to regulate the intensity of the vibration to imitate realistic friction.

5.3 Pneumatic Actuation

Complementary to vibrotactile stimuli, haptic feedback also includes stronger expressions of touch, such as pressure. In these projects, we investigated how pressure [13] and kinesthetic feedback [11] can be replicated in VR using a pneumatic approach for air cushions and pneumatic actuated muscles. To supply these actuators with reasonable force, we connected them to an air compressor and actuated them with magnetic solenoid valves. Since those require 24 Volt, which cannot be provided by the usual 3.3 V or 5 V supply of a standard microcontroller, we normally would need an external power source and relays. However, for an increasing number of actuators, this substantially extends the development time and adds to the complexity. By using ActuBoard, we reduced this to a minimum, and all components were directly connected to our system which also provided the power supply without additional circuitry. Besides, cumbersome configuration and microcontroller coding was not necessary as we could address each via our C# interface.

5.4 Further Application Ideas

The modular design provides manifold opportunities to use different off-the-shelf actuators (see 4.4). While the previous projects mostly used ActuBoard for haptic feedback in VR, we can also think of similar applications in AR or in automotive-related projects. Thanks to the small form factor, ActuBoard could even be used in tangibles or IoT devices for the smart home. Also, we see the potential for educational purposes where students can learn how to work with actuators and investigate interaction concepts, before having to understand every technical detail, similar to [6, 17, 34].

6 LIMITATIONS AND FUTURE WORK

The main intention of ActuBoard is to reduce hardware efforts when working with actuators, however, there is still potential for further extensions. Therefore, ActuBoard only supports plug-and-play for output components, while input components, such as buttons, have to be connected to the free I/O pins of the microcontroller (see 4.1). Further, we plan to fully support complex actuators that have an internal logic, e.g., stepper motors with a separate driver, as they currently can not be connected to a Hub. However, likewise to input components, it is possible to connect them directly to the microcontroller with typical effort. Further, we plan to support the interconnection between multiple ActuBoards for IoT capabilities. As we currently only support communication with a remote application, a communication between ActuBoards would even allow for distributed systems.

7 CONCLUSION

In this paper, we contributed ActuBoard which supports 1) a quick assembly and addressing of off-the-shelf components, 2) a communication interface for external applications, and 3) the inclusion of non-tech-savvy users. The plug-and-play approach for actuators and the software interfaces allow addressing actuators without being concerned about underlying communication protocols, microcontroller firmware, or circuitry. Further, we presented applications built on top of ActuBoard to showcase its practicability and gave ideas for other use-cases. We also published ActuBoard as Open Source to invite anyone interested to use, reproduce, or improve the platform.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) under No. 211500647.

REFERENCES

[1] 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

[2] Susanna Aromaa, Simo-Pekka Leino, and Juhani Viitaniemi. 2014. *Virtual prototyping in human-machine interaction design.* Number 185 in VTT Technology. VTT Technical Research Centre of Finland, Finland. Project code: 71112.

[3] Jonny Austin, Howard Baker, Thomas Ball, James Devine, Joe Finney, Peli De Halleux, Steve Hodges, Michał Moskal, and Gareth Stockdale. 2020. The BBC Micro:Bit: From the U.K. to the World. *Commun. ACM* 63, 3 (Feb. 2020), 62–69. https://doi.org/10.1145/3368856

[4] Carlos Bermejo and Pan Hui. 2017. A survey on haptic technologies for mobile augmented reality. arXiv:1709.00698 [cs.HC]

[5] 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. https://doi.org/10.1145/2858036.2858533

[6] Leah Buechley and Benjamin Mako Hill. 2010. LilyPad in the Wild: How Hardware's Long Tail is Supporting New Engineering and Design Communities. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems* (Aarhus, Denmark) *(DIS '10)*. Association for Computing Machinery, New York, NY, USA, 199–207. https://doi.org/10.1145/1858171.1858206

[7] Zikun Chen, Wei Peng, Roshan Peiris, and Kouta Minamizawa. 2017. ThermoReality: Thermally Enriched Head Mounted Displays for Virtual Reality. In *ACM SIGGRAPH 2017 Posters* (Los Angeles, California) *(SIGGRAPH '17)*. ACM, New York, NY, USA, Article 32, 2 pages. https://doi.org/10.1145/3102163.3102222

[8] Alexandra Delazio, Ken Nakagaki, Scott E Hudson, Jill Fain Lehman, and Alanson P Sample. 2018. Force Jacket : Pneumatically-Actuated Jacket for Embodied Haptic Experiences. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI'18* (2018), 1–12. https://doi.org/10.1145/3173574.3173894

[9] Hesham Elsayed, Mayra Donaji Barrera Machuca, Christian Schaarschmidt, Karola Marky, Florian Müller, Jan Riemann, Andrii Matviienko, Martin Schmitz, Martin Weigel, and Max Mühlhäuser. 2020. VRSketchPen: Unconstrained Haptic Assistance for Sketching in Virtual 3D Environments. In *26th ACM Symposium on Virtual Reality Software and Technology* (Virtual Event, Canada) *(VRST '20)*. Association for Computing Machinery, New York, NY, USA, Article 3, 11 pages. https://doi.org/10.1145/3385956.3418953

[10] C. Gallacher, A. Mohtat, S. Ding, and J. Kövecses. 2016. Toward open-source portable haptic displays with visual-force-tactile feedback colocation. In *2016 IEEE Haptics Symposium (HAPTICS)*. 65–71. https://doi.org/10.1109/HAPTICS.2016.7463157

[11] Sebastian Günther, Mohit Makhija, Florian Müller, Dominik Schön, Max Mühlhäuser, and Markus Funk. 2019. PneumAct: Pneumatic Kinesthetic Actuation of Body Joints in Virtual Reality Environments. In *Proceedings of the 2019 on Designing Interactive Systems Conference* (San Diego, CA, USA) *(DIS '19)*. ACM, New York, NY, USA, 227–240. https://doi.org/10.1145/3322276.3322302

[12] Sebastian Günther, Florian Müller, Dominik Schön, Omar Elmoghazy, Martin Schmitz, and Max Mühlhäuser. 2020. Therminator: Understanding the Interdependency of Visual and On-Body Thermal Feedback in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. ACM, New York, NY, USA. https://doi.org/10.1145/3313831.3376195

[13] Sebastian Günther, Dominik Schön, Florian Müller, Max Mühlhäuser, and Martin Schmitz. 2020. PneumoVolley: Pressure-based Haptic Feedback on the Head through Pneumatic Actuation. In *Proceedings of the 2020 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. ACM, New York, NY, USA. https://doi.org/10.1145/3334480.3382916

[14] Annie Kelly, R. Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. 2018. ARcadia: A Rapid Prototyping Platform for Real-Time Tangible Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3173574.3173983

[15] 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. https://doi.org/10.1145/3379337.3415887

[16] David Kushner. 2011. The making of Arduino. *IEEE spectrum* 26 (2011). https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino

[17] Zuzanna Lechelt, Yvonne Rogers, Nicolai Marquardt, and Venus Shum. 2016. ConnectUs: A New Toolkit for Teaching about the Internet of Things. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI EA '16)*. Association for Computing Machinery, New York, NY, USA, 3711–3714. https://doi.org/10.1145/2851581.2890241

[18] David Ledo, Miguel A. Nacenta, Nicolai Marquardt, Sebastian Boring, and Saul Greenberg. 2012. The HapticTouch Toolkit: Enabling Exploration of Haptic Interactions. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (Kingston, Ontario, Canada) *(TEI '12)*. Association for Computing Machinery, New York, NY, USA, 115–122. https://doi.org/10.1145/2148131.2148157

[19] Johnny C. Lee, Daniel Avrahami, Scott E. Hudson, Jodi Forlizzi, Paul H. Dietz, and Darren Leigh. 2004. The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (Cambridge, MA, USA) *(DIS '04)*. Association for Computing Machinery, New York, NY, USA, 167–175. https://doi.org/10.1145/1013115.1013139

[20] Y. Lin, Y. Lin, M. Yang, and J. Lin. 2019. ArduTalk: An Arduino Network Application Development Platform Based on IoTtalk. *IEEE Systems Journal* 13, 1 (2019), 468–476. https://doi.org/10.1109/JSYST.2017.2773077

[21] 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. https://doi.org/10.1145/3290605.3300633

[22] Jonatan Martínez, Arturo S. García, Miguel Oliver, José P. Molina, and Pascual González. 2014. VITAKI: A Vibrotactile Prototyping Toolkit for Virtual Reality and Video Games. *International Journal of Human–Computer Interaction* 30, 11 (2014), 855–871. https://doi.org/10.1080/10447318.2014.941272 arXiv:https://doi.org/10.1080/10447318.2014.941272

[23] 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

[24] Kouta Minamizawa, Yasuaki Kakehi, Masashi Nakatani, Soichiro Mihara, and Susumu Tachi. 2012. TECHTILE Toolkit: A Prototyping Tool for Design and Education of Haptic Media. In *Proceedings of the 2012 Virtual Reality International Conference* (Laval, France) *(VRIC '12)*. Association for Computing Machinery, New York, NY, USA, Article 26, 2 pages. https://doi.org/10.1145/2331714.2331745

[25] Mathias Nordvall, Mattias Arvola, Emil Boström, Henrik Danielsson, and Timothy Overkamp. 2016. Vibed: A prototyping tool for haptic game interfaces. In *iConference 2016*. iSchools.

[26] Sabrina Panëels, Margarita Anastassova, and Lucie Brunet. 2013. TactiPEd: Easy Prototyping of Tactile Patterns. In *Human-Computer Interaction – INTERACT 2013*, Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 228–245.

[27] S. A. Panëels, J. C. Roberts, and P. J. Rodgers. 2010. HITPROTO: a tool for the rapid prototyping of haptic interactions for haptic data visualization. In *2010 IEEE Haptics Symposium*. 261–268. https://doi.org/10.1109/HAPTIC.2010.5444647

[28] Roshan Lalintha Peiris, Wei Peng, Zikun Chen, Liwei Chan, and Kouta Minamizawa. 2017. ThermoVR: Exploring Integrated Thermal Haptic Feedback with Head Mounted Displays. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 5452–5456. https://doi.org/10.1145/3025453.3025824

[29] J. Sarik and I. Kymissis. 2010. Lab kits using the Arduino prototyping platform. In *2010 IEEE Frontiers in Education Conference (FIE)*. T3C–1–T3C–5. https://doi.org/10.1109/FIE.2010.5673417

[30] Sue Sentance, Jane Waite, Steve Hodges, Emily MacLeod, and Lucy Yeomans. 2017. "Creating Cool Stuff": Pupils' Experience of the BBC Micro:Bit. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 531–536. https://doi.org/10.1145/3017680.3017749

[31] C. Swindells, S. Pietarinen, and A. Viitanen. 2014. Medium fidelity rapid prototyping of vibrotactile haptic, audio and video effects. In *2014 IEEE Haptics Symposium (HAPTICS)*. 515–521. https://doi.org/10.1109/HAPTICS.2014.6775509

[32] Xavier Vilajosana, Pere Tuset, Thomas Watteyne, and Kris Pister. 2015. OpenMote: Open-Source Prototyping Platform for the Industrial IoT. In *Ad Hoc Networks*, Nathalie Mitton, Melike Erol Kantarci, Antoine Gallais, and Symeon Papavassiliou (Eds.). Springer International Publishing, Cham, 211–222.

[33] Nicolas Villar, James Scott, and Steve Hodges. 2010. Prototyping with Microsoft .Net Gadgeteer. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction* (Funchal, Portugal) *(TEI '11)*. Association for Computing Machinery, New York, NY, USA, 377–380. https://doi.org/10.1145/1935701.1935790

[34] Torben Wallbaum, Swamy Ananthanarayan, Andrii Matviienko, and Susanne Boll. 2020. A Real-Time Distributed Toolkit to Ease Children's Exploration of IoT. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society* (Tallinn, Estonia) *(NordiCHI '20)*. Association for Computing Machinery, New York, NY, USA, Article 9, 9 pages. https://doi.org/10.1145/3419249.3420179

[35] 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

[36] G. Gary Wang. 2003. Definition and Review of Virtual Prototyping . *Journal of Computing and Information Science in Engineering* 2, 3 (01 2003), 232–236. https://doi.org/10.1115/1.1526508 arXiv:https://asmedigitalcollection.asme.org/computingengineering/article-pdf/2/3/232/5637643/232_1.pdf

[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