LearnIoTVR: An End-to-End Virtual Reality Environment Providing Authentic Learning Experiences for Internet of Things

Zhengzhe Zhu*
zhu714@purdue.edu
Purdue University

Ziyi Liu*
liu1362@purdue.edu
Purdue University

Youyou Zhang
zhan3264@purdue.edu
Purdue University

Lijun Zhu
zhu944@purdue.edu
Purdue University

Joey Huang
chujenh@uci.edu
University of California Irvine

Ana Villanueva
villana@purdue.edu
Purdue University

Xun Qian
qian85@purdue.edu
Purdue University

Kylie Peppler
kpeppler@uci.edu
University of California Irvine

Karthik Ramani
ramani@purdue.edu
Purdue University

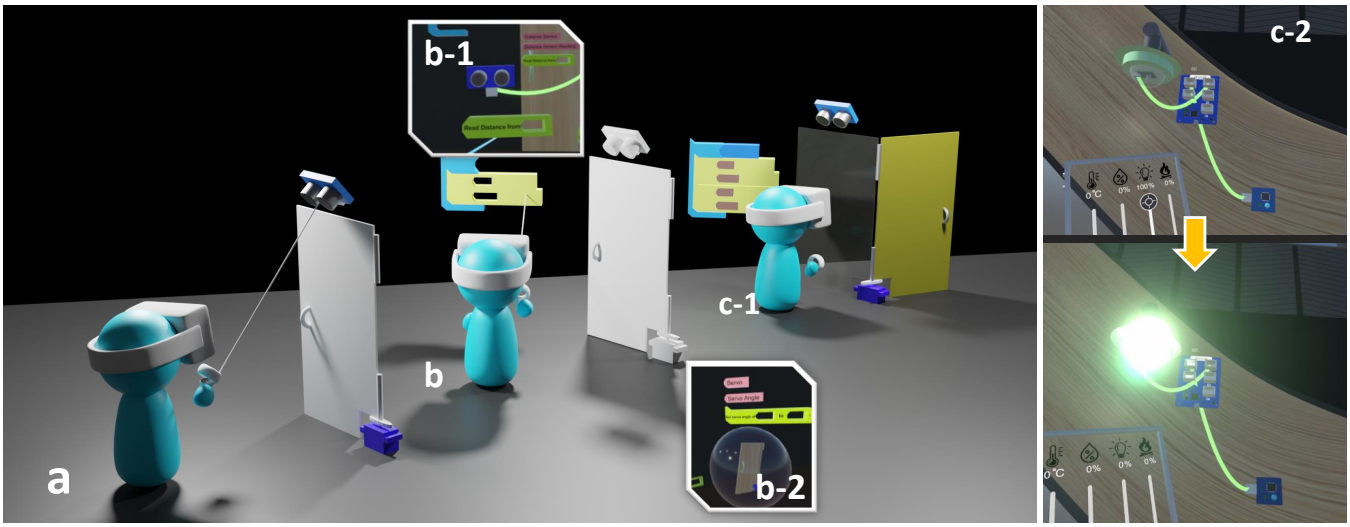

Figure 1: An overview of the end-to-end learning process in LearnIoTVR. (a) A student starts the learning process by installing the motion sensor above the door frame and the servo motor inside the door shaft. (b) The student then programs an automatic door that opens by itself when it detects a person approaching. The programming is performed by assembling the block-based language which is custom designed for the immersive environment. (b-1) Coding blocks are initially situated beside their relevant objects. (b-2) The "*container*" feature creates a proxy for objects from afar. It allows users to program the interaction to open a door and the light in another room in one place. (c) Two exploration mechanisms supported by LearnIoTVR: (c-1) The student approaches the door in person and checks if the door would open within the distance set in the program. (c-2) The student arbitrarily changes the darkness of a room through a control panel and then observes if the light can be automatically turned on.

ABSTRACT

The rapid growth of Internet-of-Things (IoT) applications has generated interest from many industries and a need for graduates with relevant knowledge. An IoT system is comprised of spatially distributed interactions between humans and various interconnected IoT components. These interactions are contextualized within their ambient environment, thus impeding educators from recreating authentic tasks for hands-on IoT learning. We propose LearnIoTVR, an end-to-end virtual reality (VR) learning environment which

*Both authors contributed equally to this research.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581396

helps students to acquire IoT knowledge through immersive design, programming, and exploration of real-world environments empowered by IoT (e.g., a smart house). The students start the learning process by installing virtual IoT components we created in different locations inside the VR environment so that the learning will be situated in the same context where the IoT is applied. With our custom-designed 3D block-based language, students can program IoT behaviors directly within VR and get immediate feedback on their programming outcome. In the user study, we evaluated the learning outcomes among students using LearnIoTVR with a pre- and post-test to understand to what extent does engagement in LearnIoTVR lead to gains in learning programming skills and IoT competencies. Additionally, we examined what aspects of LearnIoTVR support usability and learning of programming skills compared to a traditional desktop-based learning environment. The results from these studies were promising. We also acquired insightful user feedback which provides inspiration for further expansions of this system.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**.

KEYWORDS

Virtual Reality, IoT, Block-based Programming, Project-based Learning, Immersive Programming, Embodied Interaction

ACM Reference Format:
Zhengzhe Zhu, Ziyi Liu, Youyou Zhang, Lijun Zhu, Joey Huang, Ana Villanueva, Xun Qian, Kylie Peppler, and Karthik Ramani. 2023. LearnIoTVR: An End-to-End Virtual Reality Environment Providing Authentic Learning Experiences for Internet of Things. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3544548.3581396

1 INTRODUCTION

The Internet of Things (IoT) refers to the interconnected system of physical objects — "things" — that are embedded with sensors, actuators, and software [73, 103]. IoT technologies enable everyday objects to sense the environment and automatically perform tasks without human intervention [48, 91]. IoT has a massive potential to automate conventional workflows and has received ubiquitous adoption in numerous industries such as healthcare [1], transportation [10], manufacturing [8], and agriculture [10] with an estimated economic impact of up to $11 trillion by 2025 [9]. This emerging market creates a significant demand for graduates with IoT knowledge, which highlights the need to broaden the reach of IoT education [68, 69]. As a step in this direction, we explore an entry-level IoT learning environment for novice learners. Specifically, we target middle school through college-age learners who either 1) have no prior IoT knowledge, or 2) only know the basics of IoT while having no experience of programming IoT applications. The desired learning outcome for IoT may vary under different circumstances [80, 123]. In this work, we aim for the learning goals defined by Lechelt [68] which encompasses: 1) understanding the basics of IoT components (e.g., sensors and actuators), 2) understanding how these components interact with each other in an overall system, and 3) employing basic programming skills to create IoT applications.

With its components scattered around and embedded within the environment, a real-world IoT system is difficult to be recreated as a whole for hands-on and authentic learning experiences [21, 59, 75]. To bypass this challenge, some educators have chosen to decouple IoT components from their environment. For instance, the servo, the motor, and the microcontroller can be taken out and placed on the student's table, as opposed to being installed on the door (Figure 1 a). However, this approach risks decontextualized learning since the IoT components are presented in isolation from their real-world application environment [111]. For this reason, some educators have gone to great lengths to set up dedicated laboratories where realistic IoT applications are recreated [35, 50]. However, this approach has limited scalability due to significant cost and maintenance effort. On the other hand, virtual reality (VR) can simulate IoT components and their surrounding environment in a low-cost and flexible manner [33, 88, 117]. Furthermore, VR can be utilized to reenact events that would otherwise be difficult or dangerous to access in real life (e.g., an IoT system putting out a fire). Thus, we envision that simulating an integrated IoT system in VR presents a preferable alternative to provide real-world learning in educational settings.

Although virtual reality has been applied to provide authentic learning experiences for different subjects (e.g., chemistry [39, 83], art & cultural learning [55, 56], language learning [34, 84]), a full-fledged VR environment for IoT learning has yet to be developed. The main effort is to design a programming interface as programming IoT applications is an integral part of the learning process [68, 123]. The traditional 2D programming interface is inherently incompatible with the 3D environment. To address this issue, we adopt the immersive programming paradigm which directly integrates the programming interface within VR. In this way, students can constantly refer to spatial information surrounding them while programming while getting immediate feedback on the programs' executions. Previously, immersive programming interfaces have been primarily targeted at end-users who want to quickly prototype interactions and have rarely been adopted for educational purposes. To close this gap, we adapt the block-based programming language for use in VR. This language, which was initially designed for programming education and has since received widespread adoption, has proven to provide sufficient scaffolds for novice learners, who are our target users [126, 128]. By designing a 3D block-based programming interface, we try to open up space for the application of immersive programming in an educational context.

We propose LearnIoTVR, an innovative learning environment where instructors can bring project-based IoT education to students. To provide students with a deeper knowledge of the hardware, the learning process starts with installing IoT components to respective objects in the environment. Then, the students enter the programming stage where they define interactive behaviors of IoT with immersive block-based programming. Creating the outcome of a program requires assembling relevant blocks together and entering intended parameters. Meanwhile, the student can simultaneously program several devices from afar thanks to the *"container"* feature. During the exploration stage, students can freely alter the environmental context while visualizing the corresponding effects on the

IoT system in-situ and in real-time. Our system supports students to alter environmental context through first-person actions or by directly adjusting parameters on a control panel.

We propose the following contributions:

- An end-to-end IoT learning framework that supports installing, programming, and exploring authentic IoT systems in virtual reality.
- An immersive programming interface along with a custom-designed 3D block-based programming language.
- A flexible exploration mechanism that allows students to alter environmental context through first-person actions or direct parameter adjustments.
- One user study to evaluate the educational efficacy and usability of the system, and another one to compare our system with a traditional desktop-based virtual learning environment.

2 RELATED WORK

2.1 Pedagogical Practices of IoT

IoT represents the interconnection of physical objects that are embedded with sensors, actuators, microcontrollers, and other technologies [67, 71, 75]. It is used for making an environment intelligent and supportive of any human activity [27]. As one of the most important technologies in the 21st century, it is revolutionizing many industries [1, 7, 8, 10]. This trend has generated a high, unmet demand for IoT-knowledgeable graduates [30]. Therefore, it is crucial for the HCI community to provide context-aware and immersive applications that can entice students from different backgrounds to IoT engineering. In this direction, recent works have proposed learning tools with low entry barriers catering to those with little or no IoT background [69, 123]. The desired learning outcome for IoT may vary under different circumstances [80, 123]. In this paper, we aim for a comprehensive learning goal defined by Lechelt [68] which encompasses: 1) understanding the basics of IoT components (e.g., sensors and actuators), 2) understanding how these components interact with each other in an overall system, and 3) employing basic programming skills to program IoT applications.

Based on past findings from IoT education research [27, 29], effective learning of IoT generally takes place during students' interactions with real-world issues and applications. Real-world projects naturally demand real resources but real IoT devices are typically too complex, expensive, and often privacy-sensitive to be easily brought into the classroom [107]. Luckily, thanks to affordable microcontrollers like Arduino [43] with a wealth of compatible sensors and actuators [4], it is possible to develop various systems with less investment. Meanwhile, some modularized IoT toolkits have been developed in an effort to further lower the entry barrier [69, 102, 118, 123]. Nevertheless, these hardware components introduced for educational purposes are too simplified and have too many constraints [27], which could restrict the creativity of students. For instance, the servo motor used for education does not have enough torque to open a door, whereas a virtual servo can be freely configured to take up this task. In light of this limitation, we propose to virtualize IoT components to make learning experiences more unconstrained.

Furthermore, electronic components in an IoT system need to be embedded with spatially distributed objects, which is often time-consuming and may require special expertise [20, 50]. To simplify this process for classroom settings, some educators choose to introduce a mini version of an IoT system that is less constrained by physical environment. For example, Zhong et al. [140] presented three educational projects: *Stock Ticker*, *Water Leak Detector*, and *Lock Checker*, with each involving only one small physical object embedded with a handful of electronic devices. These projects are usually designed in an ad-hoc manner with limited room to expand, thus missing the opportunity to promote a holistic and system-level understanding of the IoT [107]. To address this issue, some researchers have tried to build an entire laboratory dedicated to hands-on IoT learning [35, 50]. These laboratories are designed using real-world smart environments (e.g., smart public building [50] and smart home [35]) as templates so that they are sophisticated enough to support various learning objectives. However, the substantial construction and maintenance costs involved would inevitably impede wider adoption.

To support a flexible and accessible project-based learning experience, we propose an end-to-end learning framework where students acquire comprehensive IoT knowledge through installing, programming, and exploring various IoT systems while being situated within a virtual smart environment. Taking advantage of VR technology, we simulate a wide range of real-world IoT applications to accommodate diverse learning experiences. Same with the aforementioned project-based learning approaches, students explore these applications with the guidance of instructors.

2.2 Applying VR in Education

In recent years, VR technology has seen numerous applications in the education domain [24, 32, 82, 99]. The availability of consumer-grade head-mounted displays (HMDs) [14, 17] allows for the creation of immersive learning experiences at a reasonable cost [46], paving the way for its mainstream adoption. Meanwhile, many researchers have set out to investigate the unique benefits of educational VR technology [22, 22, 95, 135]. Besides bringing more engagement, self-efficacy, and motivation among students [36, 52, 89], VR has two notable advantages compared to traditional learning mediums. Firstly, VR can simulate situations that would otherwise be difficult or impossible to access in real life. For instance, prior works have created VR environments for surgery training [72], aviation training [88], and chemistry experiment [74]. Secondly, VR can support a better understanding of spatial context by immersing students in the same environment [60, 87, 115]. It is particularly helpful in teaching abstract concepts that are rich in spatial information (e.g., molecular structures [83, 133], astronomical objects [25], and sorting algorithms [95]) which would not be effectively visualized in a traditional 2D interface [87].

We anticipate that these two benefits of VR could also apply to IoT education. On the one hand, an IoT system takes a lot of effort to create and maintain [20, 59]. The vast diversity of IoT applications may further add to this difficulty [91]. On the other hand, obtaining spatial information (i.e., environment layout and components distribution) is crucial for understanding and programming interactions between IoT [51, 108]. For instance, a motion

sensor's detection range should be configured depending on where it is installed, while an LED's luminance level should be determined by the size of the area it lights up. Leveraging these benefits, we adopt VR technology to simulate the IoT system as our learning environment. To the best of our knowledge, this is the first VR environment specifically designed for comprehensive IoT learning.

2.3 Programming Interface for VR

2.3.1 2D Programming & Immersive Programming. Currently, IoT programs are created on 2D interfaces. As a way to lower the barrier, many works have proposed visual programming interfaces where users program IoT behaviors by manipulating program elements graphically instead of writing lines of code [57]. For example, Node-RED [12] adopted the flow-based programming paradigm where users program IoT behaviors by wiring together configured IoT devices. In comparison, Xi et al. [134] adopted the block-based programming paradigm where users program IoT behaviors by assembling language blocks. Similarly, Smart Block [23] focused on increasing expressiveness when creating large IoT applications with its custom-designed visual block language.

However, these 2D programming interfaces are inherently incompatible with the VR's 3D environment. To address this issue, we adopt the immersive programming paradigm, which refers to programming directly within an immersive environment (e.g., AR, VR, MR) [76, 100, 113, 141]. Steed et al. [112] were among the earliest to introduce this concept by allowing users to define interaction with objects and their behaviors while immersed within the system itself. Following their lead, researchers have proposed authoring tools from which developers can program VR applications while situated in the VR world [137, 138]. Similarly, some other immersive programming interfaces have been developed for robot task planning [31, 45, 58] and real-time intelligent environment management [40, 53, 124].

Based on these works, we summarized two benefits of immersive programming and apply them to the IoT learning context. Firstly, it offers students with sufficient spatial information, context, and functionality of the IoT system while programming. Secondly, it allows students to get immediate feedback on their programs' executions since the programming interface is located beside the targeted IoT. The streamlined code-test cycle encourages students' active exploration which is essential for discovery learning [38, 86].

2.3.2 Block-Based Programming. In previous works, the development of immersive programming interfaces has been primarily focused on helping end-users (e.g, application developers, and engineers) to achieve higher productivity. Thus, they employed simplified programming primitives such as icons, arrows, or buttons to bypass some low-level coding structures. Those programming primitives, although easy to use, generally have a low ceiling of expressiveness which limits the room for students' exploration [78, 138]. They also bypass some low-level coding structures which are essential for scaffolding the learning process [116]. On the other hand, block-based programming language, which is initially designed for educational purposes, has proven to provide sufficient scaffolds for novice learners (low threshold) who are our target users [126, 128]. Led by the success of Scratch [101], Blockly [44], and Code.org [5], block-based programming is now an established part of the computer science education landscape. It has been shown to reduce the cognitive load by chunking code into a smaller number of meaningful elements [26]. Its elimination of syntax errors also allows users to focus on understanding the core computer science concepts and the structure of a program [129]. Users programming in block-based languages were found to spend less time off task and complete more of the activity's goals when compared to those using text-based languages [127].

There are different variations of block-based programming languages which caters to a diverse audience. Some are designed to be easily understandable to pre-teens children [98, 130] while others can support sophisticated programs taught at university classes [37, 64]. The three major styles of block languages are: (a) icons blocks, which rely on images instead of text to convey what the blocks do; (b) natural language blocks, which use standard written sentences; and (c) computer language blocks, which look just like an existing text-based language [90]. Several recent works contribute to the design of block-based programming languages for the VR environment that fall into the first two categories [54, 105, 122]. Although intuitive and easier to understand, their block-based languages can only define a limited set of logic and thus are not generalizable for more sophisticated programming tasks.

To close this gap, we offer a more flexible design in which a block could dynamically mutate its shape to incorporate different input parameters. These parameters can be freely adjusted throughout the process without affecting the overall block structure. As a result, more complex and advanced programming constructs (e.g., function calls) are supported.

3 THEORETICAL FRAMEWORK

The design of LearnIoTVR is guided by various established learning theories. In this section, we analyze prior literature and explain how their findings inspire us to develop unique features in LearnIoTVR.

3.1 Cognitive Theory of Multimedia Learning

Richard Mayer's seminal book Multimedia Learning [77] details his extensive research on how to structure multimedia materials effectively to maximize learning. In particular, two of the principles he distilled from his finding inspire the design of our system.

The Spatial Contiguity Principle states that "Students learn better when corresponding words and pictures are presented near rather than far from each other on the page or screen" (p. 135). In essence, spatial proximity can help students draw mental connections between related elements. Following this principle, we place function blocks (Figure 5 a-3) near their associated components, display sensor readings on top of the sensor, and implement the *container* feature which can teleport remote IoT components to students' local space.

The Temporal Contiguity Principle dictates that different forms of learning media (e.g., text, and pictures) should be delivered concurrently. Based on this principle, LearnIoTVR displays the sensor readings in real-time as students continuously alter the environmental context.

3.2 Cognitive Load Theory

This theory defines Extraneous Load (EL) as the mental resources devoted to elements that do not contribute to learning [114]. It has to be kept as low as possible to leave an adequate amount of mental resources for learning. To this end, LearnIoTVR selectively displays content to avoid overwhelming students with irrelevant information.

For instance, only finished programs while unused programming blocks are hidden during the exploration stage. Meanwhile, the control panel which allows students to adjust environmental context only displays parameters that are associated with current sensors in the scene.

3.3 Embodied Learning

Embodied cognition theories emphasize the relationship between physical activity and cognitive processes and are supported by a growing body of evidence from psychology and neurobiology [131, 132]. Thus, educational research incorporating students' bodily experiences, often referred to as embodied learning, is receiving considerable attention in the research community. In particular, some researchers have sought to exploit body interactions in immersive environments (e.g., VR [85, 109], AR [97], immersive whiteboard [47, 49]) to promote learning.

Following their lead, LearnIoTVR integrates the concept of embodied learning into the exploration interface, where students can alter the environmental context through embodied activities. For example, they can test the automatic door by walking towards it or change the smoke level by turning on the stove. This exploration mechanism can be more intuitive as it resembles the activities that an end-user would conduct in real life.

4 SYSTEM OVERVIEW

4.1 Virtual IoT Components

A generic IoT system in the real world consists of sensors, actuators, microcontrollers, and communication modules [106]. We compile a list of the most commonly used IoT components and recreated their corresponding virtual replicas (Figure 2). This list is derived after surveying popular online stores which sell electronic components for education [2, 4]. We do not include dedicated communication modules which are assumed to be already integrated within the microcontrollers. To avoid overwhelming students with nuanced details of the components, we choose to overlook the difference between different models and unify them under one virtual representation. In another word, the IoT components we are using are generic. These virtual IoT components inherit the capabilities as well as the overall appearance of their real-world counterparts. For instance, the microcontroller contains the exact same pins as the real one. Students are required to route the wires to the correct pins in order for this microcontroller to be functional.

4.2 3D Block-based Programming Language

Recently, some researchers have explored designing 3D block-based languages for VR environments [54, 105, 122]. Each of their programming blocks is represented by a fixed-sized cube which represents one predefined operation. Due to this lack of flexibility,

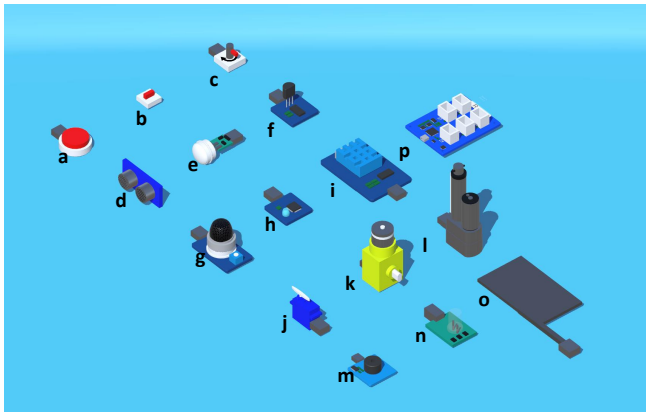

Figure 2: IoT components library: (a) Button (b) switch (c) potentiometer (d) ultrasonic distance sensor (e) motion sensor (f) temperature sensor (g) gas sensor (h)light sensor (i)humidity sensor (j) servo motor (k) DC motor (l) linear actuator (m) buzzer (n) LED (o) heater (p) microcontroller board.

the proposed block-based languages have had a low ceiling in the past. They are constrained to be used for programming primitive behaviors such as navigating a virtual character around the scene.

To support IoT programming which involves more complex logic, we borrow the *inner block* concept from Blockly–a popular block-based language library available for web platforms. In this paradigm, blocks can be nested within each other both horizontally (Figure 3 b-1) and vertically (Figure 3 b-2) rather than merely being stacked together. The shape of the parent block can adjust accordingly when incorporating different sizes and numbers of child blocks. This mechanism is particularly helpful when users want to dynamically configure the input parameters or code structures inside an existing block. The increased flexibility enables our 3D block-based language to support more sophisticated programs like IoT applications.

We classify our programming blocks into two categories — universal blocks (Figure 3 a-1) and IoT-specific blocks (Figure 3 a-2). The universal blocks are those representing universal programming constructs (e.g., while, if-else) across different components while the IoT-specific blocks encapsulate modularized functions (e.g., SetPin ()) for specific sensors/actuators. In a traditional programming environment, these functions are generally provided in certain component libraries (e.g., servo.h).

4.3 Installation Interface

The hands-on learning experience starts with installing IoT components to their respective physical objects in different places. For example, a distance sensor can be attached to the door frame to detect if a person is approaching, and a servo motor can be installed into the door shaft to open the door at a certain angle. After this process, an ordinary door is transformed into an automatic door.

In this interface, we designed a platform to display available components. Students can press the arrow button to browse through all the components in the library (Figure 4 a). Once they find the desired component, they can directly grab it from the platform and

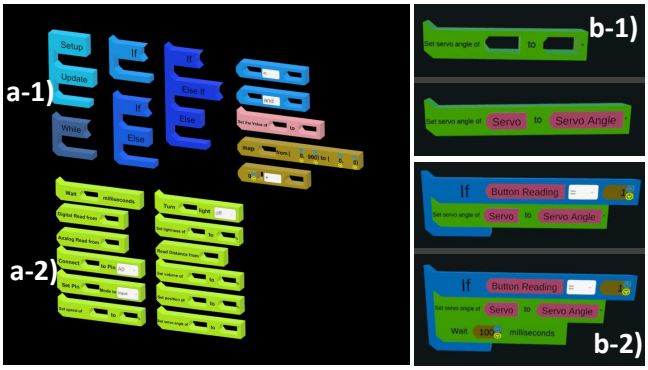

Figure 3: (a) The library of custom-designed programming blocks that includes (a-1) universal blocks and (a-2) IoT specific blocks. (b) The mesh of a parent block when rescaled (b-1) horizontally and (b-2) vertically.

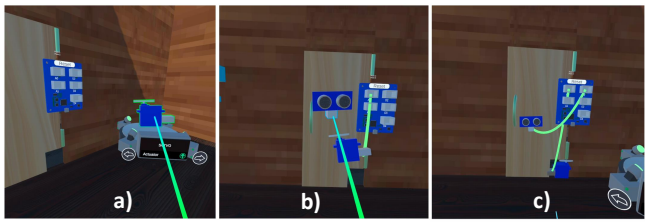

Figure 4: The installation interface. (a) Choose the component from the library. (b) Route the wires from the component to the correct pins on the board. (c) Place them onto appropriate locations

route the wires from sensors/actuators (assuming the microcontroller is pre-selected) to the designated pins on the microcontroller (Figure 4 b). Lastly, they need to put the components to the appropriate locations (Figure 4 c). It is worth noting that the installation location and angle would affect the components' behaviors. For instance, if the motion sensor is not facing down, it cannot detect the students' presence during exploration even if they programmed it correctly.

4.4 Programming Interface

Once the installation process is complete, students enter the programming interface. In the middle, there is the central workspace where students put together blocks to construct programs (Figure 5 a-2). If certain function blocks or variable blocks (e.g., Figure 5 a-3) are specifically applicable to an IoT component (e.g., LED), we display them directly above that component (Figure 5 a-3). This positioning enables users to locate and select blocks with minimal mental effort. The rest of the general-purpose blocks are organized and listed on a toolbox panel (Figure 5 a-1). For quick access, we further organize the general-purpose blocks into four categories — Math, Logic, Variable, and Function (Figure 5 a-3). All the blocks in the scene are configured to always directly face the students so they do not have to constantly adjust their body posture to see the block clearly (Figure 5 b). Throughout the programming period,

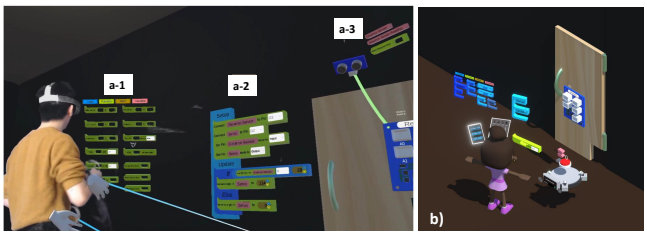

Figure 5: (a) The single-device programming interface. (a-1) The general-purpose blocks. (a-2) The central workspace. (a-3) The component-specific blocks. (b) The top-down perspective of the scene during programming.

the associated IoT components always appear in the background to give students more context information.

4.4.1 Single Device Programming. A unit of an IoT system is a single device equipped with several components. The automatic door shown in Figure 5 is an example. The distance sensor on the door frame activates the servo motor in the door shaft when it senses a person is approaching. In such cases, programming IoT behaviors is straightforward as all the components and their associated blocks are located in the vicinity.

4.4.2 Multi Device Programming. A holistic IoT system has multiple devices that interact with each other from different locations. For instance, the user may also want the light in another room to be turned on simultaneously when the door is opened. However, it is hard for users to have an overarching view of all these spatially distributed devices while programming. Inspired by an earlier work that studied distant interactions in VR [96], we introduce an interaction feature called *container*, which allows users to create a ghost replica of the remote device. Represented as a transparent sphere, the container stores a proxy of the selected device and can be brought to wherever students need it. Just like a normal device, the replicated device inside the container has its components' relevant blocks displayed above the container. Inspired by the concept of *remote function call* [16], we allow students to program interactions between multiple devices by directly embedding the function block of the remote device (e.g., TurnLEDLight (On))into the main program.

In the previous case, users can navigate to the room where the light is located through teleportation or continuous locomotion, create a *container* for the light (Figure 6 a), go back and place it near the door for reference (Figure 6 b-1). As an IoT device, the light has an LED as its component. Therefore, the function blocks and variable blocks of the LED are visualized above the container (Figure 6 b-2), which makes them easier to be selected and added to the main program.

4.5 Exploration Interface

Once the programming is complete, students enter the exploration interface where they actively change the environmental context and observe the corresponding effects on the IoT system. In this interface, we only display the finished program in front of each device while hiding the unused blocks and components to give

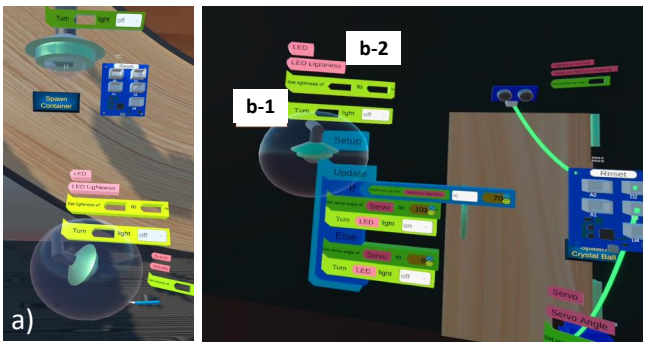

Figure 6: (a) The multi-device programming interface. (a) Create a *container* for the remote device. (b-1) Light inside *container* carried to the local space. (b-2) Component-specific blocks are shown above the *container*.

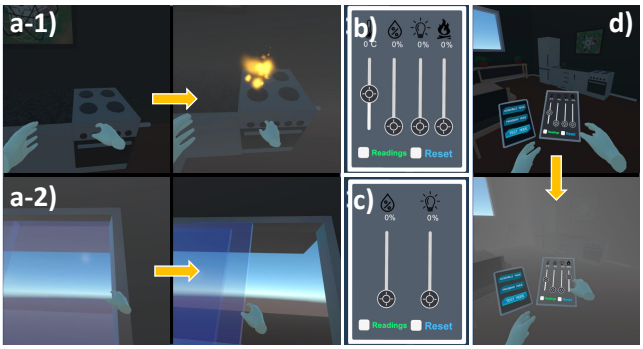

Figure 7: (a) Alter the environment through embodied actions. (a-1) turn on the stove to increase the smoke level. (a-2) Open the window to disperse the smoke. (b)(c) Adaptive environment control panel. (d) The environment change reflected in VR after the smoke level is tuned up.

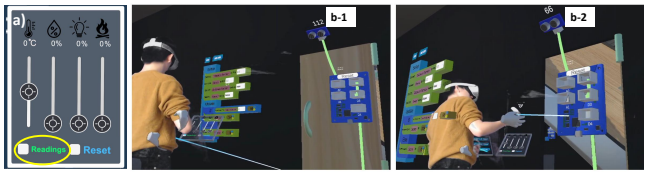

Figure 8: (a) The *display readings* option. (b) The user controls his avatar to approach the automatic door while checking the variations of the sensor readings (b-1),(b-2).

users a non-obstructive view during exploration. Leveraging VR as an immersive technology, we design two exploration mechanisms.

Firstly, students can utilize their body movement to alter the environmental context. For instance, they can manually turn on the stove to increase the smoke level (Figure 7 a-1) or open the window to disperse the smoke (Figure 7 a-2). These processes are intuitive and realistic as they resemble real-life scenarios. However, it is hard for students to fine-tune the environmental parameters using the first method which might be needed in some cases.

To complement the first mechanism, we also allow students to arbitrarily set environmental parameters (e.g., temperature, humidity, lightness) to a specific value through a control panel (Figure 7 b). To test an automatic light, students decrease the illumination parameters and see if the LED turns on (Figure 7 b-3). To test a smoke detector, students increase the smoke level (Figure 7 b-4) until the buzzer rings. The display of parameters on the panel is dynamic, as it is related to the kinds of sensors included in the scene. For instance, if there is only a humidity sensor and a lightning sensor in the scene, the panel will only display humidity (Figure 7 c-1) and illumination parameters (Figure 7 c-2) for students to adjust. Each environment change is reflected through a unique visualization so that it can be realistically perceived by the users (Figure 7 d).

Sometimes there are issues in a program that prevent devices from functioning as expected. To help users quickly identify these issues, we provide a *display readings* option (Figure 8 a). Once the option is enabled, every sensor and actuator in the scene displays their real-time internal value on top (Figure 8 b-1, b-2). For instance, a distance sensor displays its current distance from the person, and a servo motor displays its current opening angle. This extra level of data visualization enables users to closely monitor the execution of the program, which leads to more efficient debugging.

Throughout this process, users can spontaneously return to the programming interface and make changes to the blocks in the code, run it again, and test whether this debugging process was successful. By displaying the coding and testing process, we encourage students to learn through frequent trial-and-error exploration.

4.6 System Summary: A Detailed Use Scenario

We summarize the hands-on learning experience offered by LearnIoTVR with the following example. Bob is a student who wants to learn about IoT systems, he wants to build an automatic door that opens by itself upon sensing his presence. After correctly installing the IoT components (Figure 9 a), he creates and organizes the code blocks for them to function logically (Figure 9 b). However, after he starts testing the initial code, he discovers that the door does not open as expected when he approaches it (Figure 9 c). To locate the problem in the code, he enables the *Display Readings* option (Figure 9 d). By comparing the internal value of the distance sensor (i.e., raw value readings) with the threshold value for the distance he set in the program, he soon realizes that this threshold is too small. After resolving this issue by changing the threshold (Figure 9 e f), he wants to expand this application by adding a light that would simultaneously turn on by itself when the door opens. To do so, he navigates to another room where the light is located and uses the *container* to teleport it to the vicinity of the door (Figure 9 g). After adding the light's function block (i.e., TurnLEDLight (On)) to the main program (Figure 9 h), he can enter the exploration interface to experience the final IoT system (Figure 9 i).

4.7 Implementation

The BlocklyVR interface is developed in Unity3D (2020.3.0.f1) and compiled into an Oculus Quest headset [14]. The 3D IoT component

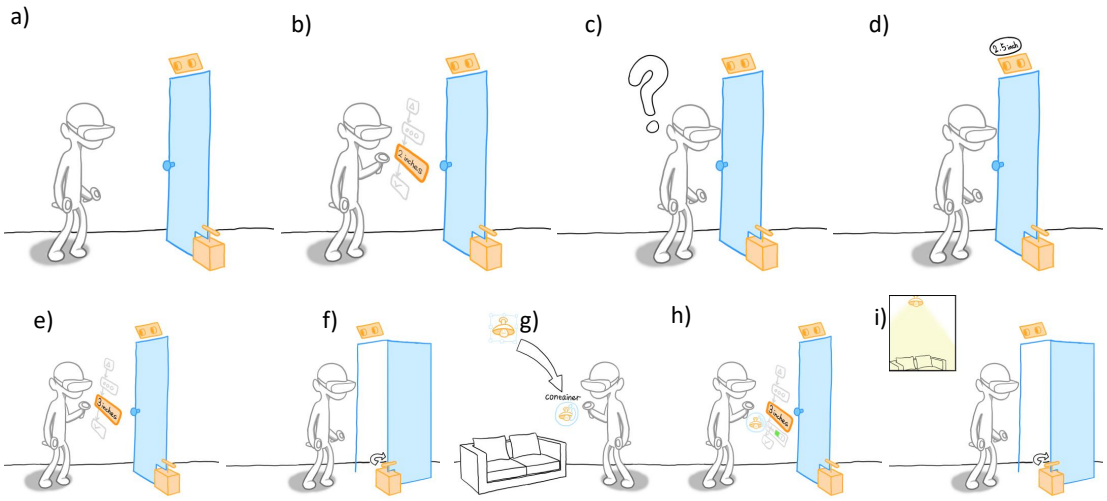

Figure 9: Step-by-step example of a learning process.

models and 3D programming blocks were made in Blender [3], and then exported into Unity3D. To dynamically change the shape of a parent block once it takes in child blocks, we performed mesh recalculation of the parent block based on the vertices of child blocks previously stored in arrays. This process is done in Blender with the help of its built-in mathematical functions.

5 USER STUDY EVALUATION

We conducted a two-session user study to evaluate our system. The first session aims to evaluate how effective our system is in helping students acquire IoT knowledge. We also evaluated the usability of the system using a System Usability Scale (SUS).

In the second user study, we compared LearnIoTVR with a traditional 2D screen-based virtual learning environment. In the second user study, we went in depth on what students like and dislike about this system compared to its traditional replacement. The second user study is designed to evaluate user experience. Therefore, we did not measure the participants' learning outcomes in the second session.

5.1 Session One: Educational Efficacy and Usability

5.1.1 Setup. For this session, we recruited 24 participants (15 male, 9 female) ranging from 18 to 25 years old (M=19.7, SD=2.29). 59% of our participants were freshmen who just entered university this year. 71% of our participants did not have programming experience in general. 92% of our participants did not have any prior knowledge of IoT, and 83% of our participants had no previous experience with programming electronic devices. 17% of our participants had previous VR experience, but it was limited to a few gaming experiences. None of the users have programmed an IoT application before. In general, the participants met our definition of target users.

The entire study took around 3.5 hours including the 2 hours for learning in VR and the rest for completing the pre-test and post-test. Considering that fatigue and motion sickness that are common in VR experiences, we allocate 5 min of resting time every 20 min. Each user was paid 40 dollars.

At the start of the user study, each participant received the context of the experiment and was requested to complete a pre-test on paper. The pre-test assessed previous knowledge of IoT and coding, as well as presented several exercises on organizing, writing, reading, and understanding block-based code. Then, participants wore the headset and were asked to explore the VR environment and adjust to the controllers. The VR environment is set in a "Smart Home", meaning that all IoT components are spatially distributed to where they correspond inside the house. This process took approximately 30-40 minutes.

After this preparation, participants began to explore the two tasks we provided. The first task requires students to program several single devices while the second task deals with the programming of a holistic IoT system involving two devices. The tasks are designed to increase in complexity so that participants would not get overwhelmed at the beginning. After reaching the designated goal of each step, participants were encouraged to further explore their IoT applications by adjusting some parameters and observing its outcomes. The descriptions of the tasks are as follow:

Task 1

Step 1: Programming the door (servo motor, ultrasonic distance sensor, push-button)

- When the push-button is pressed, the door opens at a 90-degree angle.

- Replace the push button with an ultrasonic distance sensor. When the ultrasonic distance sensor detects human is within a certain range, the door opens at a 90-degree angle.
- Use the *while ()* loop and *wait ()* function in conjunction to control the speed to open the door. The longer the wait time in each iteration, the slower the door opens.

Step 2: Programming the window (linear actuator, switch, potentiometer)

- When the switch is raised, the linear actuator opens the window completely. When the switch is lowered the window is closed completely.
- Use a potentiometer as a knob to linearly control the linear actuator to partially open the window.

Step 3: Programming the smoke alarm (smoke sensor, buzzer)

- Set the smoke sensor in digital mode (i.e., return 0 or 1) and set the volume of the buzzer to a fixed value. If the smoke sensor detects that the smoke level is above a threshold, then the buzzer turns on.
- Set the smoke sensor in analog mode (i.e., return range of values) and set the volume of the buzzer adaptively. The higher the smoke level, the louder the buzzer rings.

Task 2: Integration of step 2 and step 3 of task 1 (linear actuator, switch, potentiometer, smoke sensor, buzzer)
If the smoke sensor value reaches a certain threshold, it not only turns on the buzzer but also opens the window.

As mentioned in the related work, LearnIoTVR requires the involvement of instructors to guide students. During this user study, two of our researchers who have served as teaching assistants for IoT-related classes took the role of instructors. This user study was a one-on-one procedure while each researcher was responsible for half of the users. Each instructor provided instructions from outside the VR environment but in the same physical room. In order for the instructor to be aware of what the student was working on inside the VR system, we duplicated and projected the participants' VR display to an external monitor.

In the beginning, the instructor guided participants through the Oculus official tutorial [15] to learn how to navigate and manipulate objects in the VR environment. Then the instructor proceeded to guide participants through each task. During this process, the instructor first explained the background and requirements of the task. Then the instructor explained the functionality of the IoT components and the programming concept associated with the task using respective virtual components and code blocks in VR as reference. When programming IoT behaviors, the participant can refer to a reference picture in VR showing how the code blocks should be assembled to construct the program. This form of instruction is similar to those adopted by other educational block-based coding platforms [11, 13]. The instructor remained by the side, ready to answer any questions and offer further explanations if participants failed to understand the reference or got confused about certain concepts. After finishing programming, the participants entered the exploration stage where participants actively explored their IoT applications without the instructor's intervention. Exploration behaviors involve 1) altering the environmental context, 2) changing parameters in the program, and 3) reconstructing some or all of the code blocks themselves. The inclusion of the exploration process aims to make students active agents in the learning process instead of simply doing tasks as instructed.

The second task is designed to be the integration of the last two steps in the first task. In this task, the instructor only described the task objective to the participant at the beginning. No reference pictures are shown by default and the instructor would not step in unless the students specifically requested it.

After the learning session in VR, the participants were asked to complete a post-test with the same questions that appeared in the pre-test. They also completed a standard System Usability Scale (SUS) questionnaire.

5.1.2 Evaluation Metrics & Results. During the second task, we observed that 14 of the 24 users could complete the task independently, which indicated their ability to reconstruct the IoT applications they learned in the first task. 8 users needed the guide from the reference picture. In comparison, all users in task 1 needed to refer to these pictures. In addition, only 2 users in task 2 required additional help from the instructor. As previously mentioned, we adopted the learning goal defined by Lechelt [68] which encompasses: 1) understanding the basics of IoT components, 2) employing basic programming skills to create IoT applications, and 3) understanding how these components interact with each other in an overall system. Thus, the test given to the students measures their learning outcome of these three aspects.

For better clarity, we classify the hardware-related knowledge into three key competencies: *component type identification and PIN mode setting*: knowing the functionality of a sensor (reads values from the environment) or an actuator (produces an action based on those readings) and being able to connect to its corresponding port on the board. the PIN for a sensor is set to input mode, while the PIN for an actuator is set to output mode; *digital vs. analog*: Understand the difference between a digital mode (i.e., binary reading) and analog mode (i.e., range of multiple values) in a sensor; and *understanding component function*: understanding the functionality of an IoT component and its application scenarios.

Likewise, we measured participants' basic programming skills based on five key competencies. *conditionals*: the ability to understand if-else relationships; *loops*: understanding the concept of repeating pieces of code until a condition is met; *variables*: understanding that values can be stored and changed; *numeric calculations*: knowing how to use mathematical operators to manipulate numeric values; *functions*: understanding that modules of code can accomplish a specific task and that altering parameters changes the output of this code.

The pre- and post-test share the same questions(Figure 10). Each answer in the test was scored with a 0 if incorrect, +0.5 if the answer had some substance, or a +1 point if the answer was correct. Since a question in the test covers one or several of these aforementioned key competencies, we add the score to the overall sum of each corresponding competency. At last, the total points were normalized to fit into a 1-point scale for each category. This grading scheme is inspired by past work [92, 93]. All tests were graded by one primary grader. Inter-rater reliability on both the pre-test and post-test was validated by having a second person score over 25% of the data. From our rubric, two researchers in charge of grading

Type	Key Competency	Time	M	SD	Sig.
IOT Specific Knowledge	K1:Component Type Identify and set a PIN mode	Pre-Test	0.2708	0.25449	$Z = -4.141$ $p < 0.001$
		Post-Test	0.9021	0.21543	
		Gain	**0.633014202**		
	K2: Digital vs Analog	Pre-Test	0.6458	0.42934	$Z = -2.743$ $p = 0.006$
		Post-Test	0.9583	0.14116	
		Gain	**0.314531243**		
	K3: Components function and capability understanding	Pre-Test	0.2925	0.24825	$Z = -4.289$ $p < 0.001$
		Post-Test	0.8517	0.16032	
		Gain	**0.560840458**		
Computational Thinking	K4: Conditional(If If-else)	Pre-Test	0.3096	0.3454	$Z = -4.022$ $p < 0.001$
		Post-Test	0.8483	0.19437	
		Gain	**0.540372995**		
	K5: Loop(Update, while)	Pre-Test	0.3542	0.3753	$Z = -3.466$ $p < 0.001$
		Post-Test	0.7771	0.23819	
		Gain	**0.424403236**		
	K6: variable(assignment)	Pre-Test	0.2563	0.33124	$Z = -4.018$ $p < 0.001$
		Post-Test	0.8033	0.20967	
		Gain	**0.548405563**		
	K7: Numerical calculation(Map value map wrong or +,-)	Pre-Test	0.2333	0.34317	$Z = -3.932$ $p < 0.001$
		Post-Test	0.8154	0.23498	
		Gain	**0.583461215**		
	K8: Function(wait, map,AnalogRead)	Pre-Test	0.3229	0.37935	$Z = -3.789$ $p < 0.001$
		Post-Test	0.8292	0.26862	
		Gain	**0.507940139**		

Figure 10: Pre- and post-test results of key competencies assessment.

had a Cohen's Kappa of 0.71. The results of the pre-test, test, and post-tests are summarized in Figure 10. The learning gains of all the key competencies are statistically significant ($p<0.005$).

We also gave users a standard System Usability Scale (SUS) questionnaire, which returns a score of 75.4/100 with an SD of 14.86. During the semi-structured interview, the overall sentiment was that the participants enjoyed the learning experience, but would appreciate a longer period of time to become accustomed to VR technology more generally. In addition, most participants (20 out of 24) agreed that the exploration stage, where they can freely change program and environment parameters while directly experiencing the outcome, enhanced and reinforced their understanding of IoT. This finding conforms to the active learning theory which advocates for students to engage in the learning process rather than simply being the passive recipients of knowledge [28]. Meanwhile, all participants recognized the presence of the instructor was crucial during their learning process. This finding is consistent with the principle of project-based learning theory which emphasizes the importance of instructor involvement [63, 110].

5.2 Session Two: A Qualitative Study on User Experience

5.2.1 Setup. In this study, we compared our system to a desktop-based system featuring a WIMP (windows, icons, menus, pointer) interface on a website. The rationale behind designing this study is that participants could have engaged in similar interactive experiences in a 3D environment displayed on a 2D screen. Further, the typical use of block-based programming takes place on a 2D screen (e.g., Scratch). The desktop-based learning environment is a 2D equivalent of our VR environment in terms of functionality. The interface design was partially inspired by existing virtual learning websites in other fields [6, 13], which contain the virtual 3D environment displayed in 2D (Figure 11 a), the programming interface, and the tutorials. Similarly to the VR environment, the desktop-based environment facilitates the installation, programming, and exploration of the IoT system. Meanwhile, programming outcomes can be simulated in the virtual environment in real time. Students can zoom in/out and change the viewing angle through mouse operations including clicking, scrolling, and dragging (Figure 11 b).

For this session, we aimed toward a qualitative study of user experience, which analyzes users' preferences between these two learning environments. We didn't measure the learning outcome differences between them, the reason for which will be explained in Section 7.4. Therefore, no learning goals were set and users only need to report their feedback after experiencing tasks with both LearnIoTVR and the 2D interface. We provided necessary aid to ensure that users complete all the tasks. These aids included tutorials at the beginning and answering questions during the study. Ensuring that all users can complete the tasks is essential for a comprehensive evaluation as skipping a task could prevent users from experiencing certain features of LearnIoTVR. We recruited 6 participants (5 male, 1 female) ranging from 19 to 26 years old. To gather more in-depth feedback, we only recruited participants who have electronic programming experiences that include hands-on configuring of electronic devices. Four of them have been teaching assistants in related classes. The study was counterbalanced which means if one participant experiences the desktop-based system first, the second participant then used the LearnIoTVR system first. We prepared two tasks for both learning environments. The first task is the same as the one that is described in Section 4.6 (Figure 6). The second task is the same one as Task 2 from session one of the user study. The entire study lasted about 2 hours including interview time and each participant was paid 20 dollars. After completing two tasks in both environments, we held a semi-structured interview with participants to hear about their preferences for these learning environments as well as the reasoning behind them.

5.2.2 Results & Analysis. All of our participants reported fatigue and dizziness after a long period of use in VR, which is consistent with the feedback from the first session. We grouped the rest of their feedback into three categories which correspond to installation, programming, and exploration process.

Installation In terms of the installation process, participants favored the VR environment across the board. They complimented the realism of the installation process in VR as opposed to the *"detached (P2)"* experience in the desktop environment. *"When I was installing the motion sensor to the door, it was just like the other day [when] I was hanging a picture on the door (P1)". "Me grabbing and releasing the components onto different places feels more natural than clicking and dragging them through mouse (P3)".* The positive relationship between realism and immersion was also found in prior research which conducted a comparison between VR and desktops in other educational contexts. [62, 139].

In addition, the constraints of the desktop interface limit some nuanced adjustments of the components installed in the virtual space. The constraints were mainly imposed by the limited display area on the desktop and the inherent efficiency of the mouse when manipulating 3D content [104]. *"In order to see the objects [to which the components are attached] clearly, I need to zoom in and adjust the viewing angle properly. In VR, it is much simpler just to walk into that place and turn my head around (P5)". "(In VR) I can easily adjust*

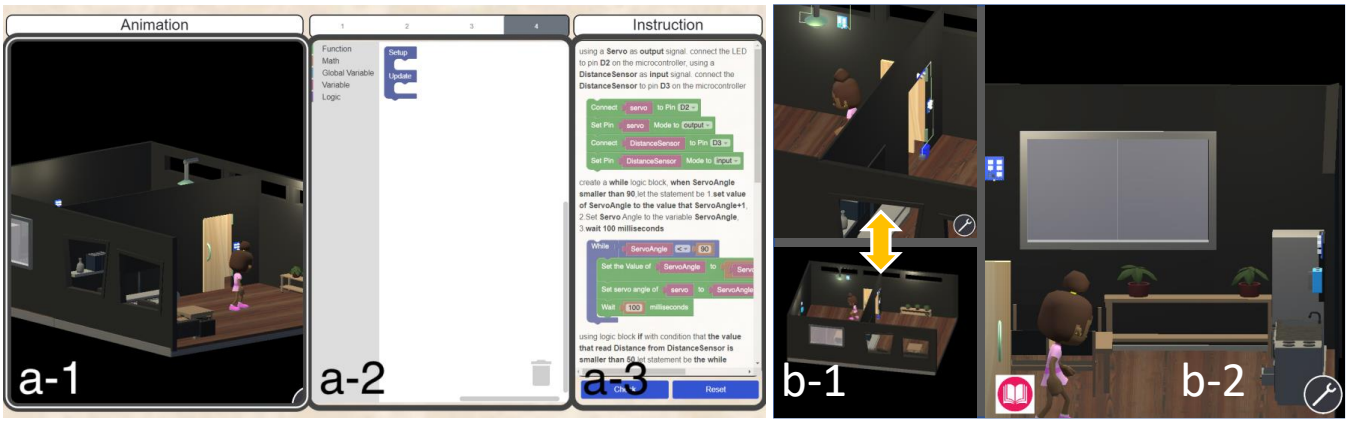

Figure 11: (a) Desktop learning environment. (a-1) Simulation environment. (a-2) Programming interface (a-3) Tutorials. (b) Navigating the simulation environment through (b-1) zooming in/out (b-2) changing perspective.

the angle of the motion sensor by grabbing two ends of the sensor and rotating it. (P3)".

Programming

The participants' opinions on the programming processes were mixed. On the one hand, most participants thought assembling the code blocks on the desktop interface is more intuitive. "Dragging and dropping a 2D block on a surface is much easier than reaching out to a 3D block in space (P2)". "Sometimes it is troublesome if the blocks (in VR) are far away from me so that I can not directly grab it (P5)". This result may be partially attributed to the *legacy bias* [81], by which users favor well-known interaction styles for familiar tasks.

On the other hand, they also mentioned several advantages of programming in the 3D environment. Firstly, embedding 3D coding blocks within the 3D virtual allows us to place blocks close to the relevant components, which helps participants *"easily locate the blocks needed (P6)".* Meanwhile since the 3D coding blocks are situated side-by-side with the programming target, participants can *"easily access environment information while programming (P5)".* Lastly, the VR hand controller provides vibration feedback whenever a virtual block is selected and some of our participants appreciated this haptic sensation. *"The programming blocks in VR seem to be more tangible and I prefer it that way (P1)."* This finding aligns with prior works which introduced tangible coding blocks to replace virtual ones and noticed users' preference for tangible feedback while programming [42, 79].

Exploration

All of our participants prefer the exploration inside the 3D world as opposed to on the 2D screen. The embodied interaction that is exclusive to immersive environments (e.g., VR) [47, 97] is the key feature that contributes to their preference for the 3D environment. *"The fact that I can use my own body to affect and interact with the environment makes me feel more empowered that I have an actual role to play in the system (P3)".*

In addition, participants appreciated the fact that they could instantly visualize the programming outcome *"directly behind the coding blocks (P1)"* in VR. On contrary, in the desktop interface, they have to *"direct their attention to separated areas (P1)"* where codes and simulations are located.

More interestingly, we noticed that participants exploring the VR world are more likely to have active and independent thinking by referring to their everyday experiences. For instance, in the second task, two of the participants who first experienced the VR environment first thought of programming a window that can open by themselves before they were briefed about this task. *"After I programmed the door that can automatically open to disperse the smoke, I instantly think I can let the window open as well which is exactly what I do every time I burned my steak (P2)".* This result could be explained by the situated learning theory which claims that the level of realism and immersion could affect students' ability to access their own everyday knowledge related to the context [66, 136].

However, two of our participants reported that sometimes it feels more intuitive to explore the environment from an overarching view as in the desktop-based interface. "Having a top-down view of entire the environment allows me to quickly switch between different places without traversing through the space (P6)". After reflecting on this feedback, we foresee that the drawbacks of the VR environment could be mitigated by implementing a *bird's-eye view* feature. Once this feature is enabled, the students can instantly have a third-person overview of the entire environment. Wang et al. [125] proposed a similar feature on their system to help users to navigate through the virtual remote space.

In summary, despite some shortcomings of the VR environment, the preference for it compared to a traditional desktop-based environment was quite clear among our participants.

6 EXAMPLE IOT APPLICATIONS

To demonstrate the diversity of learning experiences supported by LearnIoTVR, we showcase three more IoT applications that students can create and explore.

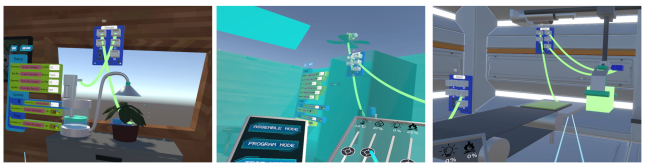

Figure 12: Example IoT Applications (a) Plant watering system. (b) Smart bathroom. (c) Automated factory.

6.1 Plant Watering System

In this use case, we present a system that automatically waters a plant. If the *humidity sensor* on the ground senses the soil is too dry, the *linear actuator* presses the piston and pumps the water out of the bucket to water the plant (Figure 12 a).

6.2 Smart Bathroom

Adjusting the temperature of an environment is important for comfort. In this use case, the temperature of the bathroom is automatically raised to be comfortably warm. Meanwhile, we also do not want too much water vapor from a hot temperature to obstruct our view. In a smart bathroom, if the *motion sensor* detects the person entering the bathroom, the *humidity sensor* and *temperature sensor* start monitoring the environment. If the temperature is lower than a certain degree threshold, the *heater* turns on to warm up the room. Meanwhile, if the humidity is higher than a certain degree threshold when the person is showering, the *motor* in the ventilator turns on to keep out the moisture. When the person leaves the room, the *motor* and *heater* automatically turn off to save electricity (Figure 12 b).

6.3 Automated Factory

In an automated factory, a high volume of industrial materials is constantly transported without human intervention. The *pressure sensor* installed under the conveyor belt detects if an object is placed on top of it. In that case, it activates the *motor* in the conveyor belt and the *linear actuator* on the sliding rail in sequence, in order to deliver the object to the designated location (Figure 12 c).

7 DISCUSSION & FUTURE WORK

7.1 Educational Benefit of Immersive Programming

Although the concept of immersive programming has been proposed for some time, it has been rarely applied in the context of education. In particular, some systems have been using immersive programming to manage real-life IoT systems. The priority of these interfaces has been simplification for end-users so they could carry out complex tasks. To this end, they often adopted simplified visual programming primitives with a lower ceiling for dynamic exploration. On the other hand, effective discovery learning requires enough room for exploration [38]. To this end, we designed our 3D block-based language for immersive programming. The two benefits of immersive programming (i.e. immediate feedback and rich contextual information) we hypothesized for IoT learning context have been corroborated by the feedback from our users. Furthermore, we envision that such benefits can also apply to other education areas that are inherently rich in spatial information. For instance, it could be used for teaching students to program the geometry of a 3D model. A desktop-based educational [6] programming environment for 3D models is already publicly available and it would be interesting to compare it to an immersive environment. Another area where immersive programming may be beneficial is storytelling, where students can concurrently develop and experience their storylines inside the virtual environment. Through facilitating quick and unconstrained idea iteration, immersive programming can allow students' imaginations to run free.

7.2 Symbiosis Between Virtual and Physical Learning

In the user study, we did not compare LearnIoTVR to traditional learning approaches where students learn IoT through programming physical components (e.g., Arduinos). The reason is that we do not view LearnIoTVR, which is a virtual learning tool, as a replacement but rather a supplement for traditional physical learning media.

LearnIoTVR fills in several missing pieces of the traditional physical learning approach. Firstly, off-the-shelf electronic kits for educational purposes are too simplified and have too many constraints for recreating many real-world systems (e.g., the automatic door), which limits the scope of applications. Meanwhile, LearnIoTVR has the potential to make learning IoT accessible for a diversity of students, rather than those traditionally best positioned to engage with physical computing. For those from low-income communities, the cost to procure physical electronic kits for different projects maybe be a burden. Some electronics can be easily damaged which adds up to the maintenance cost for those students. In contrast, VR headsets have become more affordable in recent years. For instance, an Oculus Quest [14] which our system runs on costs below $400. More importantly, the incremental cost of using VR is lower because the headset can be reused for multiple project-based learning sessions, whereas electronics for educational purposes are prone to malfunction and must be replaced regularly [19]. Inspired by a previous report showing that VR is a cost-effective solution for students from low-income families [19], we believe LearnIoTVR can offer an affordable alternative for them to acquire IoT knowledge. Lastly, the VR environment allows for more flexible exploration where students can freely simulate different environmental contexts (e.g., temperature) to test the system. This exploration mechanism would be difficult to facilitate in the physical world.

Therefore, we believe that LearnIoTVR can work in conjunction with the existing physical learning approach to deliver a more diverse and flexible experience. For example, educators can first recreate a digital twin of the physical IoT environment in VR, where students can freely test out different components and programs. Then based on the IoT applications finalized by the students, the VR environment automatically generates text-based codes that can be directly uploaded to a physical microcontroller (e.g., Arduino). Then, students can connect the microcontroller to associated components and deploy them to the physical environment. In this paradigm,

the VR environment provides flexibility as users can freely simulate different scenarios. Meanwhile, the physical learning process provides tangible experiences that have been demonstrated to be crucial for learning [41, 70].

7.3 Collaborative IoT Learning Experiences

Currently, LearnIoTVR environment can only host one user at a time and thus is not able to support collaborative learning. The benefits of collaborative learning such as developing higher level thinking and increasing retention have been increasingly emphasized within the education community [65, 120, 123]. In the meantime, VR has shown the potential to better support interpersonal communication in terms of social presence, rich non-verbal communication, and immersive realistic interactions [94]. In the future, we plan to improve LearnIoTVR by supporting multiple users to co-present as virtual avatars and collaborate in a shared VR environment. Users can take different roles in the IoT project. For instance, some can be responsible for programming while others are responsible for testing. The immersive, realistic interactions enabled by social VR can provide seamless communication between users throughout the learning process.

7.4 Continuation of the User Study

In the first session of the user study, we focused primarily on determining whether learning had occurred by comparing pre- and post-testing scores. However, we did not delve into how learning occurs. It remains to be determined which factor(s) contributed to successful learning outcomes for which knowledge concept. As a promising future research direction, we plan on running a comparative study where individual features (e.g., embodied exploration) of the system are examined in isolation. By analyzing the difference in learning outcomes between groups with enabled/disabled features, we can investigate in depth how the system can support IoT learning and beyond.

In the second session of our user study, we did not assess the learning differences between students using desktop-based environments and VR as we consider it outside the scope of this paper. This decision is made because of the significant scale and time frame required to derive meaningful results on learning differences. For instance, Nersesian et al. [83] once conducted a semester-long (18 weeks) study with 103 students to compare the academic differences in chemistry learning between VR and desktop environments. The promising results in this study indicate the advantages VR has over traditional desktop environments in promoting learning. It inspires us to further explore the educational efficacy of LearnIoTVR compared to traditional environments. In the future, we plan to design and conduct larger-scale user studies with appropriate tests to assess the learning differences.

7.5 Scalability of the Programming Interface

Currently, LearnIoTVR only focuses on programming a limited number of components at a time. When more components are involved, the number of blocks in the central workspace would increase accordingly, which inevitably crowds the interface and makes it difficult for users to program. One possible solution to this issue would be to display this content adaptively. For instance, we can decrease the opacity of the coding blocks that are not currently been selected by the user. Instead of displaying blocks in a single plane, we can also organize the coding blocks in different layers which are from close to far away from the user. Those blocks associated with the components currently being programmed are brought to the front for easier access.

7.6 Making LearnIoTVR Accessible to the General Public

As a research prototype, LearnIoTVR's mass rollout to general educators and learners is impeded by two factors. Firstly, the creation of learning materials (e.g., virtual IoT component) required expertise in VR development. Secondly, LearnIoTVR requires the constant presence of instructors to guide students through the learning process. However, we believe these limitations could be addressed in the future.

To make the learning materials accessible, we intend to build an online sharing platform where developed artifacts (e.g., IoT component) can be downloaded and shared. This is a promising direction because a similar content-sharing platform, Unity Asset Store [18], has been shown to significantly streamline the game development process. Meanwhile, novice-friendly tools for authoring virtual content [61, 119] can be adopted to make the creation of learning materials more accessible.

To alleviate the instructional burden during each session, we intend to enable instructors to pre-record a variety of instructional formats (e.g., verbal instruction, visual cue, embodied demonstration) and later display them to students. Meanwhile, LearnIoTVR can utilize AI technologies to support instructors. By adopting natural language processing technologies, RobotAR [121] demonstrated the potential of implementing intelligent tutors able to respond to frequently asked questions. It motivates us to develop intelligent tutors for LearnIoTVR along a similar path.

8 LIMITATIONS

During the user study, some participants complained that the VR headset is too heavy and that it caused some strain on their necks after some time. Some cases of minor dizziness were also reported. We regard this limitation as arising from the lack of maturity of current VR technologies in supporting desired interactions. We believe this issue will be gradually resolved as VR technology advances at the current pace.

As mentioned before, the virtual learning environment can only host one user at a time, which means the instructors have to deliver guidance from outside. Despite the fact that instructors can view students' virtual environment through display projection and scaffold learning through verbal instruction, they cannot utilize the diverse forms of non-verbal communication such as gestures, gaze, and facial expressions. During the user study, we also noticed that in some cases it is useful if instructors can *"point to the location while explaining associated questions (P3 in session two)."* This combined with the potential benefit of collaborative learning highlight the significance of developing a multi-user VR learning environment in the future.

9 CONCLUSION

In this paper, we presented LearnIoTVR, an end-to-end VR learning environment that enables installing, coding, and exploring IoT systems and promotes IoT learning. We introduce a library of common IoT components and their compatible 3D block-based programming language to allow students to freely create and explore various IoT systems. We performed a quantitative user study with 24 participants to evaluate the educational efficacy of the system. Our result demonstrated an improvement in several key competencies from our users (learning gains of up to 50%). Also, we compared LearnIoTVR with a traditional desktop-based learning environment hosted on a website and gathered/analyzed participants' in-depth feedback on user experience. In this work, we advance our understanding of IoT education in a VR environment, by removing the constraints of a physical environment and enabling immersive programming, which is becoming increasingly important in our current society.

ACKNOWLEDGMENTS

We wish to give a special thanks to the reviewers for their invaluable feedback. This work is partially supported by the NSF under the Future of Work at the Human Technology Frontier (FW-HTF) 1839971. We also acknowledge the Feddersen Distinguished Professorship Funds. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency. We sincerely thank Yikuan Liu for drawing the sketches.

REFERENCES

[1] 2021. What can IoT do for healthcare. https://www.wipro.com/business-process/what-can-iot-do-for-healthcare-/.
[2] 2022. Arduino Educational Kit. https://store-usa.arduino.cc/collections/edu-family.
[3] 2022. Blender Open source 3D creation. Free to use for any purpose, forever. https://www.blender.org/.
[4] 2022. Electronics products and services for Makers to Engineers. https://www.seeedstudio.com/.
[5] 2022. Every student in every school should have the opportunity to learn computer science. https://code.org/.
[6] 2022. Every student in every school should have the opportunity to learn computer science. https://www.blockscad3d.com/educators.
[7] 2022. How Is IoT Improving Transportation? https://www.iotforall.com/how-is-iot-improving-transportation.
[8] 2022. Industry 4.0 and the Smart Factory: Where Does IoT Fit? https://www.iotsworldcongress.com/iot-transforming-the-future-of-agriculture/.
[9] 2022. Internet of Things-We help clients unlock value by digitizing the physical world. https://www.mckinsey.com/featured-insights/internet-of-things/how-we-help-clients.
[10] 2022. IOT TRANSFORMING THE FUTURE OF AGRICULTURE. https://www.cassianetworks.com/blog/iotforsmartfarming/.
[11] 2022. Learneroo:Beginning Blockly. https://www.learneroo.com/modules/139/nodes/727.
[12] 2022. Low-code programming for event-driven applications. https://nodered.org/.
[13] 2022. Micro:bit. https://makecode.microbit.org/.
[14] 2022. Oculus 2 See it in action. https://www.oculus.com/quest-2/.
[15] 2022. Oculus Quest Introduction Tutorial. https://www.youtube.com/watch?v=PQVl8Xly4r0.
[16] 2022. Remote Function Call. https://en.wikipedia.org/wiki/Remote_Function_Call.
[17] 2022. SONY PlayStation VR. https://www.playstation.com/en-us/ps-vr/bundles/.
[18] 2022. Unity Asset Store: Community-powered creator solutions. https://assetstore.unity.com/.
[19] 2022. VR Is A Cost-Effective Solution For Title I Schools. http://www.xennialdigital.com/blog/vr-cost-effective-solution-title-I-schools/.
[20] Mostafa Al-Emran, Sohail Iqbal Malik, and Mohammed N Al-Kabi. 2020. A survey of internet of things (IoT) in education: opportunities and challenges. *Toward social internet of things (SIoT): Enabling technologies, architectures and applications* (2020), 197–209.
[21] Hanan Aldowah, Shafiq Ul Rehman, Samar Ghazal, and Irfan Naufal Umar. 2017. Internet of Things in higher education: a study on future learning. In *Journal of Physics: Conference Series*, Vol. 892. IOP Publishing, 012017.
[22] Sarune Baceviciute, Aske Mottelson, Thomas Terkildsen, and Guido Makransky. 2020. Investigating representation of text and audio in educational VR using learning outcomes and EEG. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.
[23] Nayeon Bak, Byeong-Mo Chang, and Kwanghoon Choi. 2020. Smart Block: A visual block language and its programming environment for IoT. *Journal of Computer Languages* 60 (2020), 100999.
[24] Catherine Ball and Kyle Johnsen. 2016. An accessible platform for everyday educational virtual reality. In *2016 IEEE 2nd Workshop on Everyday Virtual Reality (WEVR)*. IEEE, 26–31.
[25] Michael Barnett. 2005. Using virtual reality computer models to support student understanding of astronomical concepts. *Journal of computers in Mathematics and Science Teaching* 24, 4 (2005), 333–356.
[26] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. 2017. Learnable programming: blocks and beyond. *Commun. ACM* 60, 6 (2017), 72–80.
[27] Zorica Bogdanovic, Konstantin Simic, Miloš Milutinovic, Božidar Radenkovic, and Marijana Despotovic-Zrakic. 2014. *A Platform for Learning Internet of Things*. ERIC.
[28] Charles C Bonwell and James A Eison. 1991. *Active learning: Creating excitement in the classroom. 1991 ASHE-ERIC higher education reports*. ERIC.
[29] Barry Burd, Lecia Barker, Monica Divitini, Felix Armando Fermin Perez, Ingrid Russell, Bill Siever, and Liviana Tudor. 2018. Courses, content, and tools for internet of things in computer science education. In *Proceedings of the 2017 ITiCSE conference on working group reports*. 125–139.
[30] Barry Burd, Lecia Barker, Félix Armando Fermín Pérez, Ingrid Russell, Bill Siever, Liviana Tudor, Michael McCarthy, and Ian Pollock. 2018. The internet of things in undergraduate computer and information science education: exploring curricula and pedagogy. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 200–216.
[31] Yuanzhi Cao, Zhuangying Xu, Fan Li, Wentao Zhong, Ke Huo, and Karthik Ramani. 2019. V. Ra: An in-situ visual authoring system for robot-IoT task planning with augmented reality. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. 1059–1070.
[32] Andrea Casu, Lucio Davide Spano, Fabio Sorrentino, and Riccardo Scateni. 2015. RiftArt: Bringing Masterpieces in the Classroom through Immersive Virtual Reality.. In *STAG*. 77–84.
[33] Terrence L Chambers, Amit Aglawe, Dirk Reiners, Steven White, Christoph W Borst, Mores Prachyabrued, and Abhishek Bajpayee. 2012. Real-time simulation for a virtual reality-based MIG welding training system. *Virtual Reality* 16, 1 (2012), 45–55.
[34] Alan Cheng, Lei Yang, and Erik Andersen. 2017. Teaching language and culture with a virtual reality game. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 541–549.
[35] Jeannette Chin and Vic Callaghan. 2013. Educational living labs: a novel internet-of-things based approach to teaching and research. In *2013 9th International Conference on Intelligent Environments*. IEEE, 92–99.
[36] Athanasios Christopoulos, Nikolaos Pellas, and Mikko-Jussi Laakso. 2020. A learning analytics theoretical framework for STEM education virtual reality applications. *Education Sciences* 10, 11 (2020), 317.
[37] Emre Çoban, Özgen Korkmaz, Recep Çakır, and Feray Uğur Erdoğmuş. 2020. Attitudes of IT teacher candidates towards computer programming and their self-efficacy and opinions regarding to block-based programming. *Education and Information Technologies* 25, 5 (2020), 4097–4114.
[38] Barney Dalgarno, Gregor Kennedy, and Sue Bennett. 2014. The impact of students' exploration strategies on discovery learning using computer-based simulations. *Educational Media International* 51, 4 (2014), 310–329.
[39] Bosede Iyiade Edwards, Kevin S Bielawski, Rui Prada, and Adrian David Cheok. 2019. Haptic virtual reality and immersive learning for enhanced organic chemistry instruction. *Virtual Reality* 23, 4 (2019), 363–373.
[40] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, and George Fitzmaurice. 2017. Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In *Proceedings of the 43rd Graphics Interface Conference*. 156–162.
[41] Min Fan, Alissa N Antle, Maureen Hoskyn, Carman Neustaedter, and Emily S Cramer. 2017. Why tangibility matters: A design case study of at-risk children learning to read and spell. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 1805–1816.
[42] Min Fan, Uddipana Baishya, Elgin-Skye Mclaren, Alissa N Antle, Shubhra Sarker, and Amal Vincent. 2018. Block talks: a tangible and augmented reality toolkit for children to learn sentence construction. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–6.

[43] Aamir Fidai, Hyunkyung Kwon, Gabrielle Buettner, Robert M Capraro, Mary Margaret Capraro, Cynthia Jarvis, Madison Benzor, and Saaransh Verma. 2019. Internet of things (IoT) instructional devices in STEM classrooms: Past, present and future directions. In *2019 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[44] Neil Fraser. 2015. Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. IEEE, 49–50.

[45] Samir Yitzhak Gadre, Eric Rosen, Gary Chien, Elizabeth Phillips, Stefanie Tellex, and George Konidaris. 2019. End-user robot programming using mixed reality. In *2019 International conference on robotics and automation (ICRA)*. IEEE, 2707–2713.

[46] Hong Gao, Efe Bozkir, Lisa Hasenbein, Jens-Uwe Hahn, Richard Göllner, and Enkelejda Kasneci. 2021. Digital Transformations of Classrooms in Virtual Reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–10.

[47] Mirko Gelsomini, Giulia Leonardi, and Franca Garzotto. 2020. Embodied learning in immersive smart spaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[48] Pradyumna Gokhale, Omkar Bhat, and Sagar Bhat. 2018. Introduction to IOT. *International Advanced Research Journal in Science, Engineering and Technology* 5, 1 (2018), 41–44.

[49] Jiangtao Gong, Teng Han, Siling Guo, Jiannan Li, Siyu Zha, Liuxin Zhang, Feng Tian, Qianying Wang, and Yong Rui. 2021. HoloBoard: a Large-format Immersive Teaching Board based on pseudo HoloGraphics. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 441–456.

[50] Myriam Guedey, Anke Pfeiffer, and Dieter Uckelmann. 2020. Transferring Research on IoT Applications for Smart Buildings into Engineering Education. In *2020 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 250–254.

[51] Gaoping Huang, Pawan S Rao, Meng-Han Wu, Xun Qian, Shimon Y Nof, Karthik Ramani, and Alexander J Quinn. 2020. Vipo: Spatial-visual programming with functions for robot-IoT workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[52] Hsiu-Mei Huang, Ulrich Rauch, and Shu-Sheng Liaw. 2010. Investigating learners' attitudes toward virtual reality learning environments: Based on a constructivist approach. *Computers & Education* 55, 3 (2010), 1171–1182.

[53] Ke Huo, Yuanzhi Cao, Sang Ho Yoon, Zhuangying Xu, Guiming Chen, and Karthik Ramani. 2018. Scenariot: Spatially mapping smart things within augmented reality scenes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[54] Qiao Jin, Yu Liu, Ye Yuan, Lana Yarosh, and Evan Suma Rosenberg. 2020. VWorld: an immersive VR system for learning programming. In *Proceedings of the 2020 ACM Interaction Design and Children Conference: Extended Abstracts*. 235–240.

[55] Sheng Jin, Min Fan, and Aynur Kadir. 2022. Immersive Spring Morning in the Han Palac e: Learning Traditional Chinese Art Via Virtual Reality and Multi-Touch Tabletop. *International Journal of Human–Computer Interaction* 38, 3 (2022), 213–226.

[56] Sheng Jin, Min Fan, Yongchao Wang, and Qi Liu. 2020. Reconstructing traditional Chinese paintings with immersive virtual reality. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–8.

[57] Beate Jost, Markus Ketterl, Reinhard Budde, and Thorsten Leimbach. 2014. Graphical programming environments for educational robots: Open roberta-yet another one?. In *2014 IEEE International Symposium on Multimedia*. IEEE, 381–386.

[58] Michal Kapinus, Vítězslav Beran, Zdeněk Materna, and Daniel Bambušek. 2019. Spatially Situated End-User Robot Programming in Augmented Reality. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1–8.

[59] Mohamad Kassab, Joanna DeFranco, and Phillip Laplante. 2020. A systematic literature review on Internet of things in education: Benefits and challenges. *Journal of Computer Assisted Learning* 36, 2 (2020), 115–127.

[60] Hannes Kaufmann, Dieter Schmalstieg, and Michael Wagner. 2000. Construct3D: a virtual reality application for mathematics and geometry education. *Education and information technologies* 5, 4 (2000), 263–276.

[61] Hanbit Kim, Jaeho Sung, Joon Hyub Lee, and Seok-Hyung Bae. 2022. RCSketch: Sketch, Build, and Control Your Dream Vehicles. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–2.

[62] Sara Klingenberg, Maria LM Jørgensen, Gert Dandanell, Karen Skriver, Aske Mottelson, and Guido Makransky. 2020. Investigating the effect of teaching as a generative learning strategy when learning through desktop and immersive VR: A media and methods experiment. *British Journal of Educational Technology* 51, 6 (2020), 2115–2138.

[63] Dimitra Kokotsaki, Victoria Menzies, and Andy Wiggins. 2016. Project-based learning: A review of the literature. *Improving schools* 19, 3 (2016), 267–277.

[64] Charalampos Kyfonidis, Nektarios Moumoutzis, and Stavros Christodoulakis. 2017. Block-C: A block-based programming teaching tool to facilitate introductory C programming courses. In *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 570–579.

[65] Marjan Laal and Seyed Mohammad Ghodsi. 2012. Benefits of collaborative learning. *Procedia-social and behavioral sciences* 31 (2012), 486–490.

[66] Jean Lave and Etienne Wenger. 1991. *Situated learning: Legitimate peripheral participation*. Cambridge university press.

[67] Susan Lechelt, Katerina Gorkovenko, Luis Lourenço Soares, Chris Speed, James K Thorp, and Michael Stead. 2020. Designing for the end of life of IoT objects. In *Companion Publication of the 2020 ACM Designing Interactive Systems Conference*. 417–420.

[68] Susan Zuzanna Lechelt. 2020. *Introducing the Internet of Things in classrooms through discovery-based learning and physical computing*. Ph. D. Dissertation. UCL (University College London).

[69] 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*. 3711–3714.

[70] Zuzanna Lechelt, Yvonne Rogers, Nicola Yuill, Lena Nagl, Grazia Ragone, and Nicolai Marquardt. 2018. Inclusive computing in special needs classrooms: Designing for all. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[71] In Lee and Kyoochun Lee. 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons* 58, 4 (2015), 431–440.

[72] G Michael Lemole Jr, P Pat Banerjee, Cristian Luciano, Sergey Neckrysh, and Fady T Charbel. 2007. Virtual reality in neurosurgical education: part-task ventriculostomy simulation with dynamic visual and haptic feedback. *Neurosurgery* 61, 1 (2007), 142–149.

[73] Shancang Li, Li Da Xu, and Shanshan Zhao. 2015. The internet of things: a survey. *Information systems frontiers* 17, 2 (2015), 243–259.

[74] Wei-Kai Liou and Chun-Yen Chang. 2018. Virtual reality classroom applied to science education. In *2018 23rd International Scientific-Professional Conference on Information Technology (IT)*. IEEE, 1–4.

[75] Somayya Madakam, Vihar Lake, Vihar Lake, Vihar Lake, et al. 2015. Internet of Things (IoT): A literature review. *Journal of Computer and Communications* 3, 05 (2015), 164.

[76] Steven C Mallam, Salman Nazir, and Sathiya Kumar Renganayagalu. 2019. Rethinking maritime education, training, and operations in the digital era: applications for emerging immersive technologies. *Journal of Marine Science and Engineering* 7, 12 (2019), 428.

[77] Richard E Mayer. 2002. Multimedia learning. In *Psychology of learning and motivation*. Vol. 41. Elsevier, 85–139.

[78] Michael J McGuffin and Christopher P Fuhrman. 2020. Categories and completeness of visual programming and direct manipulation. In *Proceedings of the International Conference on Advanced Visual Interfaces*. 1–8.

[79] Edward F Melcer and Katherine Isbister. 2018. Bots & (Main) frames: exploring the impact of tangible blocks and collaborative play in an educational programming game. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–14.

[80] Filipe T Moreira, Mario Vairinhos, and Fernando Ramos. 2018. Internet of Things in education: A tool for science learning. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 1–5.

[81] Meredith Ringel Morris, Andreea Danielescu, Steven Drucker, Danyel Fisher, Bongshin Lee, MC Schraefel, and Jacob O Wobbrock. 2014. Reducing legacy bias in gesture elicitation studies. *interactions* 21, 3 (2014), 40–45.

[82] Michael Nebeling, Shwetha Rajaram, Liwei Wu, Yifei Cheng, and Jaylin Herskovitz. 2021. XRStudio: A Virtual Production and Live Streaming System for Immersive Instructional Experiences. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[83] Eric Nersesian, Adam Spryszynski, and Michael J Lee. 2019. Integration of virtual reality in secondary STEM education. In *2019 IEEE Integrated STEM Education Conference (ISEC)*. IEEE, 83–90.

[84] Eric Nersesian, Adam Spryszynski, Ulysee Thompson, and Michael Lee. 2018. Encompassing english language learners in virtual reality. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 200–203.

[85] Eric Nersesian, Margarita Vinnikov, and Michael J Lee. 2021. Travel kinematics in virtual reality increases learning efficiency. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.

[86] Melanie Njoo and Ton De Jong. 1993. Exploratory learning with a computer simulation for control theory: Learning processes and instructional support. *Journal of research in science teaching* 30, 8 (1993), 821–844.

[87] Sebastian Oberdörfer, David Heidrich, and Marc Erich Latoschik. 2019. Usability of gamified knowledge learning in VR and desktop-3D. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[88] Matthias Oberhauser and Daniel Dreyer. 2017. A virtual reality flight simulator for human factors engineering. *Cognition, Technology & Work* 19, 2 (2017), 263–277.

[89] Jocelyn Parong and Richard E Mayer. 2021. Cognitive and affective processes for learning science in immersive virtual reality. *Journal of Computer Assisted Learning* 37, 1 (2021), 226–241.

[90] Erik Pasternak, Rachel Fenichel, and Andrew N Marshall. 2017. Tips for creating a block language with blockly. In *2017 IEEE Blocks and Beyond Workshop (B&B)*. IEEE, 21–24.

[91] Keyur K Patel, Sunil M Patel, and P Scholar. 2016. Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing* 6, 5 (2016).

[92] Kylie Peppler and Diane Glosson. 2013. Stitching circuits: Learning about circuitry through e-textile materials. *Journal of Science Education and Technology* 22, 5 (2013), 751–763.

[93] Kylie Peppler, Karen Wohlwend, Naomi Thompson, Verily Tan, and AnnMarie Thomas. 2019. Squishing circuits: Circuitry learning with electronics and playdough in Early Childhood. *Journal of Science Education and Technology* 28, 2 (2019), 118–132.

[94] Catlin Pidel and Philipp Ackermann. 2020. Collaboration in virtual and augmented reality: a systematic overview. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. Springer, 141–156.

[95] Johanna Pirker, Johannes Kopf, Alexander Kainz, Andreas Dengel, and Benjamin Buchbauer. 2021. The Potential of Virtual Reality for Computer Science Education-Engaging Students through Immersive Visualizations. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 297–302.

[96] Henning Pohl, Klemen Lilija, Jess McIntosh, and Kasper Hornbæk. 2021. Poros: configurable proxies for distant interactions in VR. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[97] Iulian Radu and Alissa Antle. 2017. Embodied learning mechanics and their relationship to usability of handheld augmented reality. In *2017 IEEE Virtual Reality Workshop on K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR)*. IEEE, 1–5.

[98] Iulian Radu and Blair MacIntyre. 2009. Augmented-reality scratch: a tangible programming environment for children. In *Proceedings of conference on interaction design for children, Como, Italy.*

[99] Ananda Bibek Ray and Suman Deb. 2016. Smartphone based virtual reality Systems in Classroom Teaching—a Study on the effects of learning outcome. In *2016 IEEE eighth international conference on technology for education (T4E)*. IEEE, 68–71.

[100] Abdul Rahman Abdel Razek, Christian van Husen, Marc Pallot, and Simon Richir. 2018. A comparative study on conventional versus immersive service prototyping (VR, AR, MR). In *Proceedings of the Virtual Reality International Conference-Laval Virtual*. 1–10.

[101] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[102] Yvonne Rogers, Venus Shum, Nic Marquardt, Susan Lechelt, Rose Johnson, Howard Baker, and Matt Davies. 2017. From the BBC Micro to micro: bit and Beyond: A British Innovation. *interactions* 24, 2 (2017), 74–77.

[103] Karen Rose, Scott Eldridge, and Lyman Chapin. 2015. The internet of things: An overview. *The internet society (ISOC)* 80 (2015), 1–50.

[104] Udo Schultheis, Jason Jerald, Fernando Toledo, Arun Yoganandan, and Paul Mlyniec. 2012. Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3D tasks. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 117–124.

[105] Rafael J Segura, Francisco J del Pino, Carlos J Ogáyar, and Antonio J Rueda. 2020. VR-OCKS: A virtual reality game for learning the basic concepts of programming. *Computer Applications in Engineering Education* 28, 1 (2020), 31–41.

[106] Natalia Silvis-Cividjian. 2017. Pervasive Computing. *Undergraduate Topics in Computer Science* (2017).

[107] Natalia Silvis-Cividjian. 2019. Teaching internet of things (IoT) literacy: A systems engineering approach. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 50–61.

[108] Hyo Jun Sim and Yun Gil Lee. 2018. Developing an Internet of Things (IoT) service system based on spatial context. In *International Conference on Human-Computer Interaction*. Springer, 510–514.

[109] Mel Slater. 2017. Implicit learning through embodiment in immersive virtual reality. In *Virtual, augmented, and mixed realities in education*. Springer, 19–33.

[110] Gwen Solomon. 2003. Project-based learning: A primer. *Technology and learning-dayton-* 23, 6 (2003), 20–20.

[111] Marcus Specht. 2008. Designing contextualized learning. In *Handbook on information technologies for education and training*. Springer, 101–111.

[112] Anthony Steed and Mel Slater. 1996. A dataflow representation for defining behaviours within virtual environments. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*. IEEE, 163–167.

[113] Livia Ștefan. 2012. Immersive collaborative environments for teaching and learning traditional design. *Procedia-Social and Behavioral Sciences* 51 (2012), 1056–1060.

[114] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[115] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. 2019. Loki: Facilitating remote instruction of physical tasks using bi-directional mixed-reality telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 161–174.

[116] David S Touretzky, Daniela Marghitu, Stephanie Ludi, Debra Bernstein, and Lijun Ni. 2013. Accelerating K-12 computational thinking using scaffolding, staging, and abstraction. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 609–614.

[117] Emiel GG Verdaasdonk, Jenny Dankelman, Johan F Lange, and Laurents PS Stassen. 2008. Transfer validity of laparoscopic knot-tying training on a VR simulator to a realistic environment: a randomized controlled trial. *Surgical endoscopy* 22, 7 (2008), 1636–1642.

[118] Ana Villanueva, Hritik Kotak, Ziyi Liu, Rutvik Mehta, Kaiwen Li, Zhengzhe Zhu, Yeliana Torres, and Karthik Ramani. 2020. ARbits: Towards a DIY, AR-compatible electrical circuitry toolkit for children. In *Proceedings of the 2020 ACM Interaction Design and Children Conference: Extended Abstracts*. 205–210.

[119] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Kylie Peppler, Thomas Redick, and Karthik Ramani. 2020. Meta-AR-app: an authoring platform for collaborative augmented reality in STEM classrooms. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.

[120] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Feiyang Wang, Subramanian Chidambaram, and Karthik Ramani. 2022. Colabar: A toolkit for remote collaboration in tangible augmented reality laboratories. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW1 (2022), 1–22.

[121] Ana M Villanueva, Ziyi Liu, Zhengzhe Zhu, Xin Du, Joey Huang, Kylie A Peppler, and Karthik Ramani. 2021. Robotar: An augmented reality compatible teleconsulting robotics toolkit for augmented makerspace experiences. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

[122] Juraj Vincur, Martin Konopka, Jozef Tvarozek, Martin Hoang, and Pavol Navrat. 2017. Cubely: Virtual reality block-based programming environment. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. 1–2.

[123] 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*. 1–9.

[124] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPturAR: An Augmented Reality Tool for Authoring Human-Involved Context-Aware Applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 328–341.

[125] Zeyu Wang, Cuong Nguyen, Paul Asente, and Julie Dorsey. 2021. Distanciar: Authoring site-specific augmented reality experiences for remote environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[126] David Weintrop. 2019. Block-based programming in computer science education. *Commun. ACM* 62, 8 (2019), 22–25.

[127] David Weintrop and Uri Wilensky. 2015. Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs.. In *ICER*, Vol. 15. 101–110.

[128] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–25.

[129] David Weintrop and Uri Wilensky. 2019. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education* 142 (2019), 103646.

[130] Amanda Wilson and David C Moffat. 2010. Evaluating Scratch to Introduce Younger Schoolchildren to Programming.. In *PPIG*, Vol. 1. 1–12.

[131] Margaret Wilson. 2002. Six views of embodied cognition. *Psychonomic bulletin & review* 9, 4 (2002), 625–636.

[132] Robert A Wilson and Lucia Foglia. 2011. Embodied cognition. (2011).

[133] Mihye Won, Mauro Mocerino, Kok-Sing Tang, David F Treagust, and Roy Tasker. 2019. Interactive immersive virtual reality to enhance students' visualisation of complex molecules. In *Research and Practice in Chemistry Education*. Springer, 51–64.

[134] Wen Xi and Evan W Patton. 2018. Block-based approaches to internet of things in mit app inventor. *ser. BLOCKS+* (2018).

[135] Kexin Yang, Xiaofei Zhou, and Iulian Radu. 2020. XR-Ed Framework: Designing Instruction-driven andLearner-centered Extended Reality Systems for Education. *arXiv preprint arXiv:2010.13779* (2020).

[136] Michael F Young. 1993. Instructional design for situated learning. *Educational technology research and development* 41, 1 (1993), 43–58.

[137] Lei Zhang and Steve Oney. 2019. Studying the Benefits and Challenges of Immersive Dataflow Programming. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 223–227.

[138] Lei Zhang and Steve Oney. 2020. FlowMatic: An Immersive Authoring Tool for Creating Interactive Scenes in Virtual Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 342–353.

[139] Jiayan Zhao, Peter LaFemina, Julia Carr, Pejman Sajjadi, Jan Oliver Wallgrün, and Alexander Klippel. 2020. Learning in the field: Comparison of desktop, immersive virtual reality, and actual field trips for place-based STEM education. In *2020 IEEE conference on virtual reality and 3D user interfaces (VR)*. IEEE, 893–902.

[140] Xiaoyang Zhong and Yao Liang. 2016. Raspberry Pi: An effective vehicle in teaching the internet of things in computer science and engineering. *Electronics* 5, 3 (2016), 56.

[141] Zhengzhe Zhu, Ziyi Liu, Tianyi Wang, Youyou Zhang, Xun Qian, Pashin Farsak Raja, Ana Villanueva, and Karthik Ramani. 2022. MechARspace: An Authoring System Enabling Bidirectional Binding of Augmented Reality with Toys in Real-time. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.