User-Extensible Block-Based Interfaces for Internet of Things Devices as New Educational Tools

Yizhou Tan
Department of Architecture
Rhode Island School of Design
Providence, RI 02903–2717
Email: ytan02@risd.edu

Marina Rizk
Department of Engineering
Mississippi College
Clinton, MS 39058-4026
Email: mnrizk@mc.edu

Gordon Stein, Ákos Lédeczi
Institute for Software Integrated Studies
Vanderbilt University
Nashville, Tennessee 37212–2328
Email: {gordon.stein,akos.ledeczi}@vanderbilt.edu

SoutheastCon 2022 | 978-1-6654-0652-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SoutheastCon48659.2022.9763937

***Abstract*—Internet of Things (IoT) devices are common in students' everyday lives, but connecting these devices to a programming environment for educational use is not always straightforward. This paper presents a framework, IoTScape, for connecting IoT devices to an online block-based programming environment. This system automatically provides both a novice-friendly interface and more advanced tools integrating cybersecurity concepts. By allowing new device types to easily be added to the system, a more diverse set of curricula is possible, ideally attracting more students who may not find the existing curricula engaging. Examples are provided of IoT devices used with this system, both physical and virtual, connected to NetsBlox through this platform, along with potential pedagogical uses of these devices.**

I. Introduction

Students interact with a wide range of Internet of Things (IoT) devices in their daily lives, and their interactions with these devices will continue to increase as these technologies become even more ubiquitous. This demonstrates a clear necessity for students to gain an understanding of distributed computing concepts early in their education, especially as cybersecurity within the IoT and distributed computing domains remains a concern [1]. IoT devices can also be used as an educational tool to facilitate learning in other subjects, both in STEM fields and beyond [2], [3].

Computer science education at an introductory level is conventionally perceived to be boring and irrelevant [4]. Studies have shown that increasing the perception of relevance and usefulness of computers is crucial to the computer science education facing novices and non-majors [5]. Therefore, the education needs to be contextualized. The use of robots, for example, is helpful during this contextualization process to create unconventional motivations for students to learn computer science [6]. The premise of our work is that the introduction of Internet of Things, with a necessary level of abstraction, would be another promising way to contextualize the education and to motivate students.

However, the use of IoT devices in education are met with some barriers to entry. [7] Many devices designed for education require a significant upfront investment, in addition to maintenance and storage costs, while often requiring significant technical knowledge for both setup and use. One way to reduce these costs and enable a wider range of domains would be to use simulated devices, but connecting both them and the wide range of available physical devices to a common, novice-friendly programming environment is not always straightforward.

This work demonstrates a new platform for integrating IoT devices into an existing block-based programming environment. The platform is designed so that any device with sufficient flexibility in its communications and/or firmware can be added to the platform. The use of a block-based programming language creates a low barrier to entry for novice programmers, but additional features are available to provide a cybersecurity curriculum without the devices needing to be modified to support it. The low-level networking concepts can be abstracted away from the student's view, providing them with concepts at a more introductory level. At the same time, the platform is designed to also reduce the technical knowledge required for new devices to be added.

By providing the potential for a wide range of devices accessed through a shared, simple interface, this platform is intended to enable a wider range of curricula using devices relevant to the interests of students. Students interested in nursing or medicine could have access to healthcare related sensors. Students interested in Earth science could have access to environmental sensors, along with scientific datasets to combine their data with. Adopting a distributed computing mindset allows these lessons to be shared by students working remotely. These devices can be added to the platform by the educators themselves, empowering them to create novel content relevant to their class.

A. Related Work

There is a lack of highly relevant work in the field, regarding our project's specific use of IoT programming ability as a motivation for education in other domains. Therefore, in this section we review projects that use a certain branch of computer science as a motivation for CS education, as well as web-based IoT programming platforms facing beginners.

1) Motivational Tools: Kurkovsky [8] describes a curriculum using Sphero, a small robotic ball, as a medium to teach robotics and mobile computing concepts. It is however a

978-1-6654-0652-9/22/$31.00 ©2022 IEEE

course that requires students to work with complex hardware and software systems at a lower level, while our project proposes to abstract away such lower level computing for better engagement of the students. Burd et al. [9] also suggest that the incorporation of mobile computing concepts, potentially across multiple different platforms, is important to CS education for engaging students and increasing the education's relevance. Xu et al. [10] explore how a gaming context can be incorporated into introductory computing courses. McGill [11] examines how the incorporation of personal robots in an introductory CS course motivates non-computer science students by grabbing their attention. This study also addresses that the technical frustrations with robotics is likely to counteract the motivational effects of relevance, which is exactly what our work tackles.

2) Beginner's IoT Platforms: SoEasy [12] is a software framework that integrates the programming and configuration of different IoT peripherals. This is similar to our project in how the low-level compiling and source-coding processes are abstracted away in order for beginners to be engaged in peripheral device control programming. There are also several projects that use a block-based programming approach, same as our project, to better engage beginners. For example, BlockyTalky [13] is a platform designed specifically for introducing youth to programmable networked systems. It is similar to NetsBlox in its collaboration and networking features, but different in that the code is created on each device directly, as opposed to the central server of NetsBlox and pre-made firmwares of IoTScape. Xi et al. [14] propose a block-based programming approach using the MIT App Inventor to enable novices to build mobile apps integrated with IoT technology. Similarly, Ruiz-Rube et al [15] create extensions for the App Inventor to make IoT and mobile app development more accessible for novices. In addition to similar functionalities, our project also allows the user to extend the interface to more custom IoT devices. BIPES [16] is another web-based, block-based, integrated platform for embedded systems. It relies on frameworks such as MicroPython and Google Blockly, to allow programming blocks to be translated into Python code and then deployed to the target board.

II. Background

A. NetsBlox

To provide a novice-friendly interface for software development, a block-based programming environment is used. NetsBlox [17] is a visual block-based programming environment built upon the open-source Snap! project [18] extending it to enable programmers to create networked programs through message passing and Remote Procedure Calls (RPCs). [19] NetsBlox's message passing capability enables programs to send data between connected NetsBlox clients while the Remote Procedure Calls allow access to third party APIs such as Google Maps, TMDB and scientific data sources through simple "call" blocks. Generally, RPCs introduce simple and diverse abstractions for NetsBlox users by providing access to external resources and functions to get data from, which in return aid the user to create multiple programs that vary in complexity. Figure 1 shows an example block which displays a specific location using Google Maps using an RPC call which allows inputting latitude and longitude of the location, width and height of display screen, and the desired zoom in value of the displayed map. The resulting map is displayed below the block, but in a NetsBlox program could be stored in a variable, displayed on the screen, or sent across the network to another program. NetsBlox also assists students in writing code by providing real-time collaboration tools and cloud-based storage for their programs.

NetsBlox allows the creation of custom services by users through uploading of datasets through its "ServiceCreation" service [20]. While these datasets consist only of static data with no interactivity beyond query functions, they demonstrate the potential for students or teachers contributing new services as part of a curriculum.

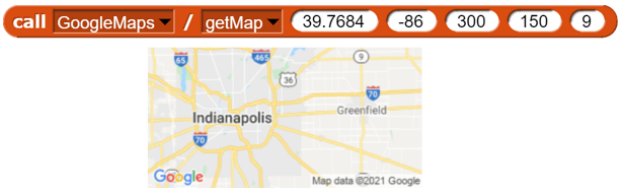

Fig. 1. Calling Google Maps in Netsblox

A recent addition to NetsBlox is the PhoneIoT service [21]. This service presents students' phones as additional components in the distributed system their program exists within, accessible through the RPC blocks. Students connect their phone and programs using an app, however, they do not write code that runs directly on the phone. Instead, students are able to send commands to the app to request sensor data, establish a user interface, and subscribe to events from the phone. Students have been able to use this interface to both create games and to control robots remotely.

Fig. 2. Example blocks for commanding a robot through RoboScape. The text-based interface (bottom) is required for the cybersecurity curriculum.

B. RoboScape

NetsBlox features support for robotics through its RoboScape service and a compatible firmware for Parallax ActivityBot 360 robots. While typical educational robotics platforms require developing software to run directly on the robot, RoboScape introduces robots while embracing NetsBlox's focus on novice-friendly distributed computing by implementing robots as if they were another web service [22]. Students interact with robots through RPCs and message passing, using abstractions shared with other networked programs in NetsBlox. The robots used feature two drive wheels with optical

encoders, an ultrasonic range sensor, a pair of touch sensors ("whiskers"), a buzzer, a button, two LEDs, and an XBee WiFi module. Figure 2 demonstrates sending a "set speed" command to a robot using the "call" blocks. Developing and debugging the robot's code is simplified as the code runs in the user's browser and students can make use of NetsBlox's collaboration tools.

An additional benefit of RoboScape's distributed nature was the easy implementation of a cybersecurity curriculum using it. [22] The robots' communications are intentionally simulated as taking place over an insecure channel. Robots do not perform any authentication of message sources, so students are able to prevent spoofed messages by sending a "set key" message to their robot, activating a Vigenère cipher, which is simple enough that they can write the encryption/decryption logic without prior experience with cryptography. Students can use a "listen" RPC to eavesdrop on the commands sent to a robot, including the "set key" message, requiring the introduction of a hardware key/secure key exchange concept to prevent eavesdropping users from obtaining the original key sent as plaintext. The robots' communication is also designed to be vulnerable to DoS and replay attacks, with defenses also available to students. In previous years, this curriculum, concluding in a final competition where students demonstrate their cybersecurity offenses and defenses, was found to be an engaging and practical way to teach computational thinking and cybersecurity concepts. [23], [24]

Recently, students were prevented from meeting in physical classrooms, so a simulation platform for RoboScape robots, RoboScape Online, was created. This Unity-based simulation environment provides a networked virtual environment for students to collaborate and compete within, regardless of their locations in real life. Besides enabling distance education, this platform also greatly reduces the cost of classroom robotics over the physical robots. Over summer 2021, two groups of high school students attended an educational program using this modality and demonstrated similar outcomes to prior years using physical robots. [25] Currently, RoboScape Online has primarily sought to recreate the existing physical robots and their curriculum extended with the affordances presented by moving to virtual environments, but the addition of IoTScape will allow for the creation of sensors and actuators beyond those present on the original robots.

III. IoTScape

With RoboScape demonstrating the applicability of robots and block-based programming as tools to provide a computational thinking and cybersecurity curriculum suitable for learners at any level, and PhoneIoT demonstrating that the same concepts can be extended to other devices, the need to generalize these interfaces beyond robots or phones has been made apparent. IoTScape is a new service for NetsBlox designed to meet this need and open service creation for new devices to the community of students and educators using NetsBlox. By providing an interface for almost any device, IoTScape enhances NetsBlox with the potential to both add new devices easily but also to facilitate interactions between devices otherwise not intended to be connected. For example, Figure 3 shows a simplified NetsBlox program to control an RGB LED to activate when a PIR motion sensor is triggered. The "listen" RPC subscribes the program to messages from a device, and the motion sensor (connected as the "MotionSensor" service) sends a "motion" message when its status changes. These two devices do not need to be physically connected or on the same network as each other or the client in any way. The only requirement is for them to both be able to connect to the NetsBlox server. Figure 4 shows the connections between components in this distributed system.

Fig. 3. Example program for setting the color of an RGB LED based on a motion sensor through IoTScape

Compatibility with IoTScape only requires a device to be able to send UDP packets to the NetsBlox server. An initial "announcement" is sent to the server with the service definition. This definition is provided as a JSON object, for easier compatibility with the JavaScript-based NetsBlox server software. To protect students from abusive language, a profanity filter is applied to inputted text, and additional content moderation tools are available to the server's administrators.

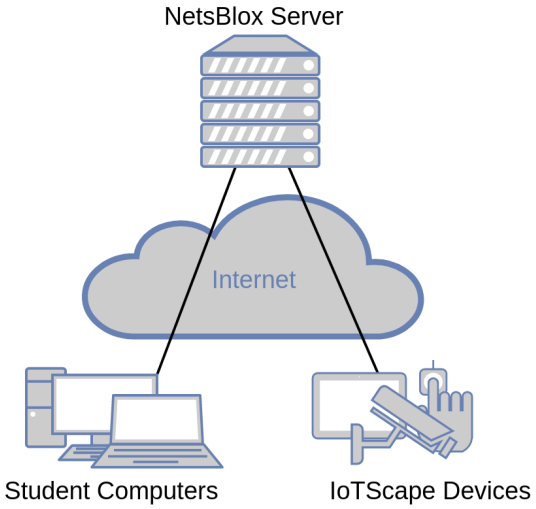

Fig. 4. IoTScape Architecture

By making the system very flexible about what devices can participate, it opens the doors to students and educators providing support for new hardware. IoTScape also includes a reimplmentation of the cybersecurity features included in

RoboScape. The commands and events for each device support both a "basic", unencrypted mode providing additional abstractions, and a "secure" mode where commands must be sent as text and encryption may be required for communication. Through this capability, IoTScape enables partial reuse of the existing curricula used with robots, but for new domains and new devices.

As with RoboScape, requests are made from NetsBlox through the RPC blocks. When a block is activated in a student's code, the request is validated, transmitted through the server, formatted as a JSON message and sent over UDP to the target device, which can send another JSON message in response. If the method's definition indicated that it will return a result, the student's code will wait for a response from the server with the result.

This system requires the devices to have certain capabilities, such as an Internet connection and reprogrammable firmware. However, the system is flexible enough such that a "proxy" device can be created to act as an adapter. For example, a smartwatch with only BLE capabilities could interact with IoTScape using a phone app as an intermediary.

A. Service Definition Tool

To simplify the creation of new IoTScape devices, a tool is provided to generate service definitions. While the JSON formatted service definitions are not expected to be a major barrier to use, a graphical interface for their creation increases the ability of students and educators with limited experience outside of NetsBlox itself to be able to contribute their own services.

This tool is also able to generate a template for use in programming an ESP8266 or ESP32-based IoT device. A general framework for IoTScape devices is provided as a library, requiring only for code to be provided for each of the methods of the device's service.

B. Unity Implementation

The protocol defined for IoTScape does not restrict it to typical, physical IoT devices. A simulation of a device, or an entirely virtual device is capable of being interacted with through a NetsBlox service as long as it can communicate with the server. To facilitate the creation of virtual devices, both for the simulated environment RoboScape Online's purposes, and for students or educators familiar with the engine already, a plugin for the Unity game engine [26] has been created. This plugin also provides a graphical interface for creating service definitions. Users working in the Unity editor create a service definition file, attach it to a Unity object with the IoTScapeObject component, and link the methods defined for the service to methods in the Unity scene. No experience with writing networking code in C# is required, and the service definition files are reusable between projects.

IV. Applications

A. Physical Devices

As a demonstration of a physical IoT device, an ESP8266-based environmental sensor was created, shown in Figure 5. This device was equipped with a PIR motion sensor, an WS2812 "NeoPixel" RGB LED, and an AM2302 digital humidity and temperature sensor. The NetsBlox code shown in Figure 3 was designed to work with the services provided by this device. The components used to construct this device are all low-cost modules commonly available as parts in hobbyist electronics kits, and breakout boards are available so that no soldering is required, making construction of a similar device suitable as a project for students as an in-class activity. With small modifications to the device's firmware, the configuration of the board could be adjusted in many ways, for example to support additional RGB LEDs in a chain.

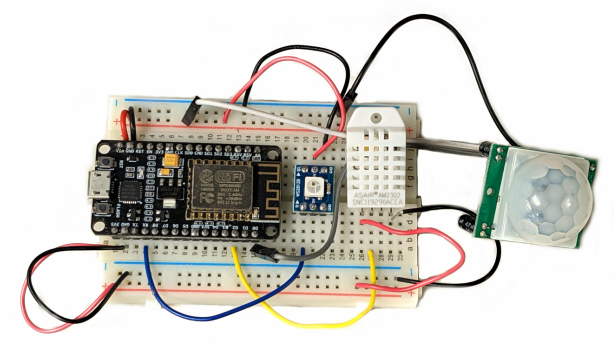

Fig. 5. Physical IoT sensor created to demonstrate IoTScape

B. Virtual Devices

Simulated devices were also created as a demonstration of this platform's flexibility. The Unity plugin for IoTScape was used to extend the existing robot simulation with new components and create new scenarios for students to use them in. These devices were implemented as an extension of the existing RoboScape Online software.

1) Virtual Robot Parts: Multiple virtual robot components were created using the plugin. The virtual platform abstracted the mechanical details to only the idea of the sensor, creating a great opportunity for us to increase the accessibility of the education towards novices. For example, the simulated LIDAR sensor is programmed to detect the distance to objects at various angles as shown in Figure 6 giving the measured distances as a list usable in NetsBlox as shown in Figure 7.

In addition to simulation of real sensors and actuators, the virtual platform also allows further customization and even fantasy functionalities to increase student interest. For example, the hardware can be modified through RPCs in NetsBlox, such as parts with colors customizable through code, or an LED strip with adjustable length. While these virtual components were added onto the robot in Unity, all of the customization is controlled through NetsBlox, as shown in Figure 8.

2) Example Usage Scenarios: A sample nuclear spill detection challenge is shown in Figure 9 where the robot was provided with a radiation sensor and tasked with detecting a nuclear spill. Figure 10 shows a sample code in NetsBlox that

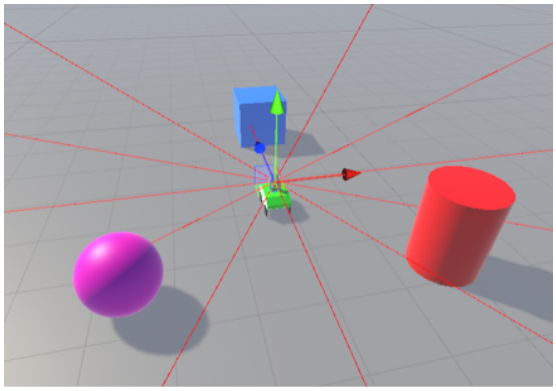

Fig. 6. LIDAR simulation in Unity

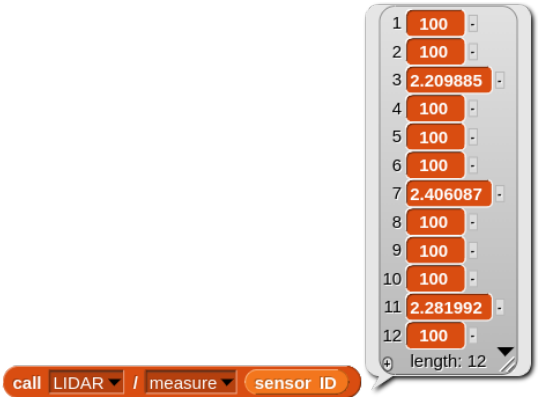

Fig. 7. LIDAR function and measurements in NetsBlox

controls the assembled robot and provides the reading for the nuclear spill intensity.

As more general robotics challenge, an obstacle course was created where the robot must be programmed to autonomously drive itself outside the maze. The robot's whisker sensors can be used to detect the obstacles and walls in its path. For example, a student may choose to have the robot navigate by moving backward and turning away from the direction a whisker sensor was triggered from, and moving forwards otherwise. The obstacles maze is shown in Figure 11.

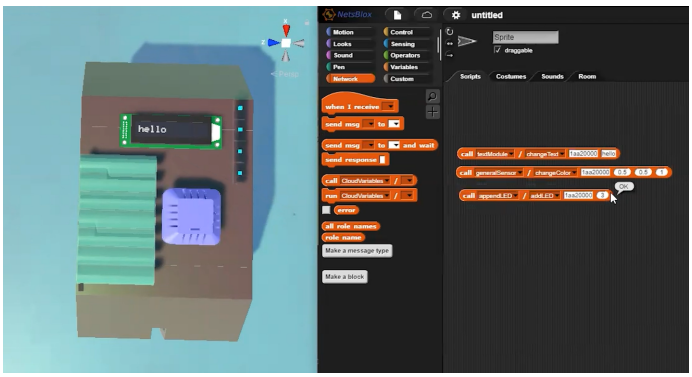

Fig. 8. Customizable virtual modules

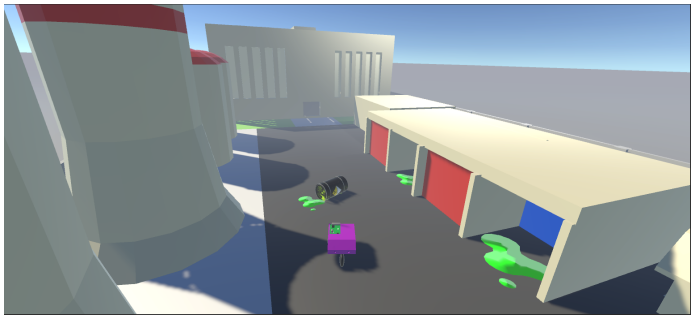

Fig. 9. Virtual robot with radiation sensor in radiation leak scenario

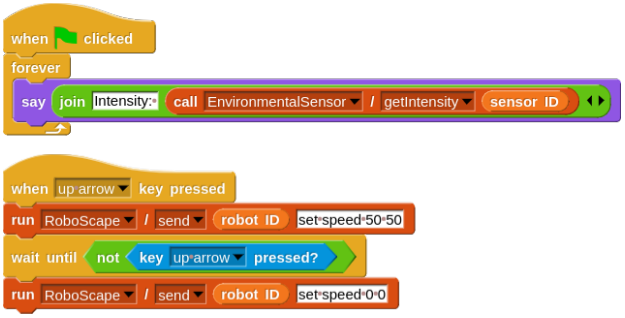

Fig. 10. Sample NetsBlox code for controlling robot and getting measured intensity

Another example challenge created was a firefighting robot task in which a student utilizes a temperature sensor and a fire extinguisher mounted to their robot to detect the fire's location and put it out. Some example code for interacting with the services created for these parts is shown in Figure 12. Figure 13 shows the robot putting out a fire inside a room where a fire was detected.

V. Conclusion

The use of IoT devices as an educational tool will benefit from increased accessibility and reduced barriers to entry in both writing programs to interact with these devices but also

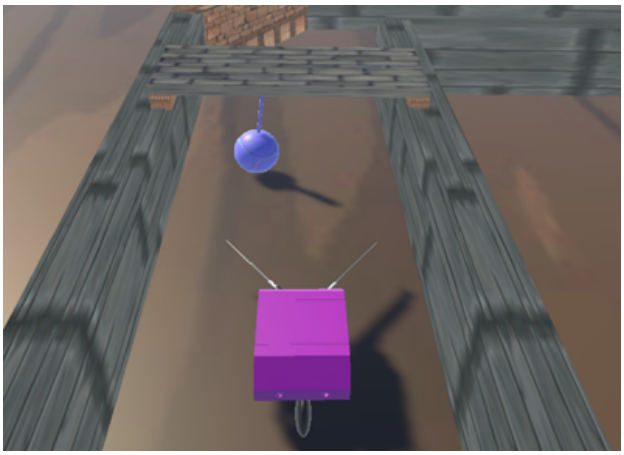

Fig. 11. Simulated robot with whiskers in the obstacle course

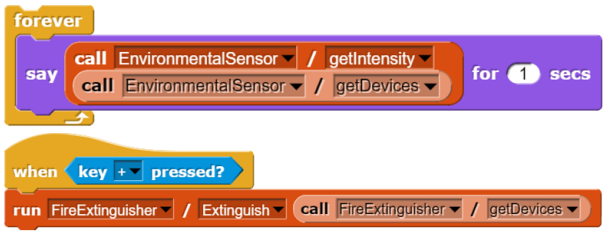

Fig. 12. Example blocks to call the temperature sensor and water nozzle in NetsBlox

Fig. 13. Simulated robot putting out fire after detecting the fire location

in providing support for new types of device. By providing block-based abstractions, IoTScape puts interaction with IoT devices on a novice-friendly level, while not limiting its use to exclusively novices. A simple protocol for connecting new devices and defining their capabilities allows for educators to add their own devices to the platform and take advantage of both the block-based interface and the existing suite of cybersecurity education features. In addition, by keeping all student code in the browser and communicating with devices as if they were web services, this approach prevents students from needing to write low-level firmware and reduces the ability for student code to damage the devices they work with.

The previous work with robotics demonstrated that internet-connected devices can be a useful educational tool, both for the directly applicable domains of computational thinking and distributed computing, but also for other domains such as cybersecurity. By giving educators the freedom to integrate new devices, physical or virtual, into our system, IoTScape will be able to provide for a diverse array of topics and create more engaging education.

A. Future Work

While potential classroom uses have been demonstrated, to fully evaluate the use of this platform as an educational tool, it will be necessary to have students interact with it in a controlled setting where data is being collected. The existing concepts will be extended into curricula targeting middle and high school students, initially building upon the prior work with the RoboScape platform and cybersecurity. These are currently planned for use in the summer of 2022.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1835874, the National Security Agency (H98230-18-D-0010) and the Computational Thinking and Learning Initiative of Vanderbilt University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, Oct 2017.

[2] S. Y. Shapsough and I. A. Zualkernan, "A generic iot architecture for ubiquitous context-aware learning," *IEEE Transactions on Learning Technologies*, vol. 13, no. 3, pp. 449–464, 2020.

[3] K. Gunasekera, A. N. Borrero, F. Vasuian, and K. P. Bryceson, "Experiences in building an iot infrastructure for agriculture education," *Procedia Computer Science*, vol. 135, pp. 155–162, 2018, the 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918314492

[4] J. S. Kay, "Contextualized approaches to introductory computer science: The key to making computer science relevant or simply bait and switch?" in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 177–182. [Online]. Available: https://doi.org/10.1145/1953163.1953219

[5] D. Joyce, "The computer as a problem solving tool: A unifying view for a non-majors course," in *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 63–67. [Online]. Available: https://doi.org/10.1145/273133.273163

[6] D. Blank, "Robots make computer science personal," *Commun. ACM*, vol. 49, no. 12, p. 25–27, Dec. 2006. [Online]. Available: https://doi.org/10.1145/1183236.1183254

[7] A. Fidai, H. Kwon, G. Buettner, R. M. Capraro, M. M. Capraro, C. Jarvis, M. Benzor, and S. Verma, "Internet of things (iot) instructional devices in stem classrooms: Past, present and future directions," in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–9.

[8] S. Kurkovsky, "Mobile computing and robotics in one course: Why not?" in *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 64–69. [Online]. Available: https://doi.org/10.1145/2462476.2465584

[9] B. Burd, J. a. P. Barros, C. Johnson, S. Kurkovsky, A. Rosenbloom, and N. Tillman, "Educating for mobile computing: Addressing the new challenges," in *Proceedings of the Final Reports on Innovation and Technology in Computer Science Education 2012 Working Groups*, ser. ITiCSE-WGR '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 51–63. [Online]. Available: https://doi.org/10.1145/2426636.2426641

[10] M. M. McGill, "Learning to program with personal robots: Influences on student motivation," *ACM Trans. Comput. Educ.*, vol. 12, no. 1, Mar. 2012. [Online]. Available: https://doi.org/10.1145/2133797.2133801

[11] D. Xu, D. Blank, and D. Kumar, "Games, robots, and robot games: Complementary contexts for introductory computing education," in *Proceedings of the 3rd International Conference on Game Development in Computer Science Education*, ser. GDCSE '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 66–70. [Online]. Available: https://doi.org/10.1145/1463673.1463687

[12] J. Lee, G.-i. Park, J.-h. Shin, J.-h. Lee, C. J. Sreenan, and S.-e. Yoo, "Soeasy: A software framework for easy hardware control programming for diverse iot platforms," *Sensors*, vol. 18, no. 7, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/7/2162

[13] A. Kelly, L. Finch, M. Bolles, and R. B. Shapiro, "Blockytalky: New programmable tools to enable students' learning networks," *International Journal of Child-Computer Interaction*, vol. 18, pp. 8–18, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212868918300394

[14] W. Xi and E. W. Patton, "Block-based approaches to internet of things in mit app inventor," ser. BLOCKS+, 2018.

[15] I. Ruiz-Rube, J. M. Mota, T. Person, J. M. R. Corral, and J. M. Dodero, "Block-based development of mobile learning experiences for the internet of things," *Sensors*, vol. 19, no. 24, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/24/5467

[16] A. G. D. S. Junior, L. M. G. Gonçalves, G. A. De Paula Caurin, G. T. B. Tamanaka, A. C. Hernandes, and R. V. Aroca, "BIPES: Block based integrated platform for embedded systems," *IEEE Access*, vol. 8, pp. 197 955–197 968, 2020.

[17] "Netsblox website," https://netsblox.org, 2021, cited 2021 March 8.

[18] "Snap! website," https://snap.berkeley.edu/, 2021, cited 2021 March 8.

[19] B. Broll, A. Lédeczi *et al.*, "A visual programming environment for learning distributed programming," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Mar 2017, p. 81–86. [Online]. Available: 10.1145/3017680.3017741

[20] B. Broll, A. Lédeczi, G. Stein, D. Jean, C. Brady, S. Grover, V. Catete, and T. Barnes, "Removing the walls around visual educational programming environments."

[21] D. Jean, G. Stein, and A. Lédeczi, *Hands-On IoT Education with Mobile Devices: Demo Abstract*. New York, NY, USA: Association for Computing Machinery, 2021, p. 390–391. [Online]. Available: https://doi.org/10.1145/3412382.3458778

[22] A. Lédeczi, M. Maroti *et al.*, "Teaching cybersecurity with networked robots," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Feb 2019, p. 885–891. [Online]. Available: 10.1145/3287324.3287450

[23] A. Lédeczi, H. Zare, and G. Stein, "Netsblox and wireless robots make cybersecurity fun," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1290. [Online]. Available: https://doi.org/10.1145/3287324.3293749

[24] B. Yett, N. Hutchins, G. Stein, H. Zare, C. Snyder, G. Biswas, M. Metelko, and A. Lédeczi, "A hands-on cybersecurity curriculum using a robotics platform," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1040–1046. [Online]. Available: https://doi.org/10.1145/3328778.3366878

[25] G. Stein, D. Jean, and A. Lédeczi, *Distributed Virtual CPS Environment for K12: Demo Abstract*. New York, NY, USA: Association for Computing Machinery, 2021, p. 394–395. [Online]. Available: https://doi.org/10.1145/3412382.3458780

[26] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A General Platform for Intelligent Agents," *arXiv*, 2018.