Demo Abstract: Distributed Virtual CPS Environment for K12

Gordon Stein
gordon.stein@vanderbilt.edu
Vanderbilt University
Nashville, Tennessee, USA

Devin Jean
devin.c.jean@vanderbilt.edu
Vanderbilt University
Nashville, Tennessee, USA

Ákos Lédeczi
akos.ledeczi@vanderbilt.edu
Vanderbilt University
Nashville, Tennessee, USA

ABSTRACT

While educational robotics and makerspaces are useful to modern STEM education, they introduce both physical and economic barriers to entry. By creating a simulated, networked environment, we can facilitate instruction on cyber-physical systems and related topics while reducing cost and complexity. The virtual environment created is connected to a block-based programming language, NetsBlox, to allow students to engage with the curriculum regardless of programming experience. The networked simulation and collaborative programming environment combine to become especially effective for distance learning. This demonstration showcases example scenarios providing students with a simple interface to interact with a simplified sensor network in a small area of a smart city and solve robot challenges.

CCS CONCEPTS

• **Computer systems organization** → *Robotics*; *Sensor networks*;
• **Social and professional topics** → **Computing education**.

KEYWORDS

computer science education, educational robotics, distance learning

ACM Reference Format:
Gordon Stein, Devin Jean, and Ákos Lédeczi. 2021. Demo Abstract: Distributed Virtual CPS Environment for K12. In *Information Processing in Sensor Networks 2021, May 18–21, 2021, Nashville, TN*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3412382.3458780

1 INTRODUCTION

The Internet of Things (IoT), robots, and other smart devices are rapidly becoming ubiquitous, hence, students should be familiar with the underlying technologies. Providing educational content on distributed computing with devices typically requires an upfront cost of purchasing hardware, establishing their network and continued maintenance. In addition, the recent prioritization of distance education may cause difficulty with physical devices. Through simulation, these costs and issues can be greatly reduced.

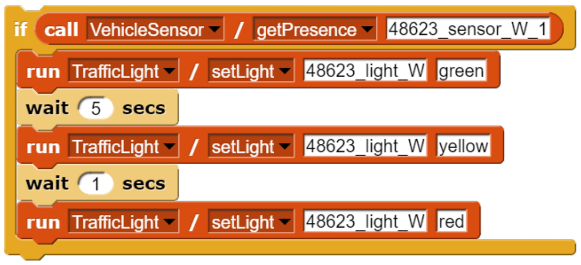
While "Robot as a Service", remote IoT education, and general educational robotics simulators have previously been used, this system intends to expand upon the domain of potential topics and provide new opportunities for students otherwise underserved by existing methods.

2 NETSBLOX

To also flatten the learning curve of programming these devices, the block-based programming language NetsBlox was used [1, 3]. NetsBlox, based on Snap! [2], allows students to access a rich programming environment from a web browser. The server infrastructure for NetsBlox provides access to web services through Remote Procedure Calls (RPC). For example, Google Maps is accessible through the "call" block, with RPCs such as "getMap". The NetsBlox editor and program code run in a web browser, making it compatible with the vast majority of student computers. In addition, NetsBlox includes collaboration features so that students can work in teams. Through message passing, students can develop applications such as multiplayer games while using an abstracted networking block set providing a level of complexity suitable for novices.

The same distributed computing abstractions, RPCs and message passing, also support interfacing with networked robots and other devices. While the typical paradigm for educational robotics is to have students write code to run on the robots, NetsBlox supported devices have pre-programmed firmware that receives commands over the network via the "send command" RPC. Robots, in turn, send acknowledgements and sensor values via messages. The communication can be "intercepted" by other students and robots will accept commands from anybody, motivating the need for cybersecurity, allowing lessons on encryption, denial of service and replay attacks, and other topics [4].

In this work, we expand the concept of interfacing with physical devices into virtual worlds. The approach includes a new component, named IoTScape, that allows arbitrary devices, whether physical or virtual, to expose their network location and capabilities to the NetsBlox server through a similar interface to physical robots. The server then creates a service for each device type accessible through NetsBlox's usual abstractions automatically. By implementing a library for Unity to define services in scene files, scenarios are able to define the capabilities of virtual devices within them. An example of blocks used by a student to retrieve data from simulated sensors and respond to it is seen in Figure 1.



```
if call VehicleSensor / getPresence 48623_sensor_W_1
    run TrafficLight / setLight 48623_light_W green
    wait 5 secs
    run TrafficLight / setLight 48623_light_W yellow
    wait 1 secs
    run TrafficLight / setLight 48623_light_W red
```

Figure 1: NetsBlox code for requesting vehicle sensor data and changing traffic light colors.

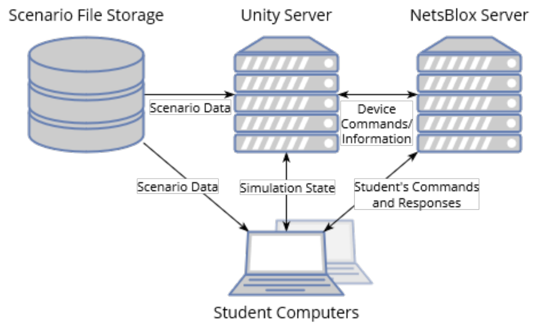


Figure 2: System Architecture

3 PLATFORM DESIGN

Virtual environments used for simulation are created in the Unity game engine. Unity allows for straightforward cross-platform development, and has previously been used for educational robotics simulations [5]. The use of a commercial off-the-shelf game engine allows the program to be designed for extension and modification by end users. Each virtual environment used by the software represents a "scenario" including a game engine scene and objects including information about interactions and evaluation. These scenarios are downloaded upon request, so a large library of content can be provided remotely. In addition, using Unity allows for simpler development of a virtual reality (VR) interface for the software. A VR interface allows students to have a deeper interaction with the virtual devices, such as physically building robots and selecting sensors to meet specifications.

Students accessing virtual environments share a networked space. To prevent student connection speeds or hardware limitations from interfering with the fidelity of the simulation, the physics and networking can run on a remote server. Each student sees a view of the same environment, allowing them to collaborate in this shared space no matter whether they are in the classroom or studying from home. The system architecture is shown in Figure 2.

4 DEMONSTRATION

Two scenarios are used for this demonstration. The first focuses on tasks utilizing sensors at a four-way intersection in a smart city (as seen in Figure 3). In this scenario, students must program a controller for the traffic signal and are evaluated based on efficiency of the intersection. Traffic in this environment comes from all directions, with flow patterns designed to test the quality and robustness of the student's solution. Desirable traits of a traffic signal at this intersection are assumed to optimize both throughput and fairness.

The second scenario focuses on robotics in an environment where multiple students can collaborate to solve a task. Students' robots are placed on an elevated platform with several boxes they are tasked with removing from the area (Figure 4). To remove these boxes, the robots must push each box off the platform's edge. This scenario can be given to students as a remote-control task, that is, they write the code to manually drive the robot with their keyboard for example, or they may be assigned to use the robots' distance sensor to autonomously locate and remove boxes. Performance is evaluated by the time required to clear the platform.

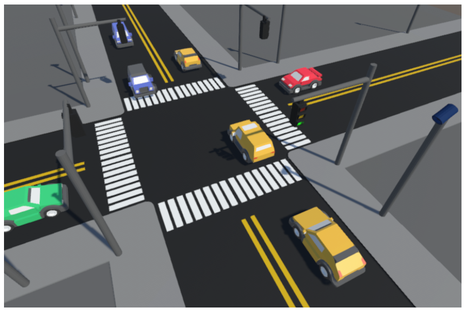


Figure 3: Overhead view of the "smart city" environment

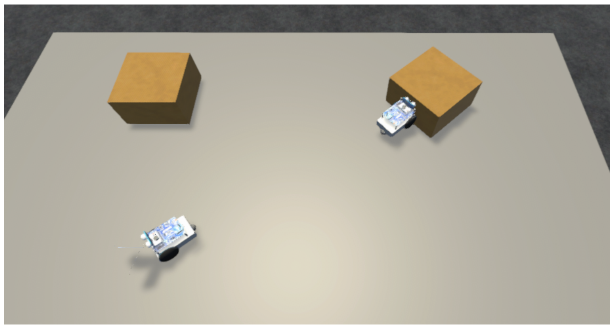


Figure 4: Overhead view of the "box pushing" task

Although this demonstration focuses on programming controllers for cyber-physical systems, STEM disciplines outside of computer science and engineering could take advantage of this system. For example, the virtual sensors could be used to obtain data from simulations of climate or ecology. The underlying framework each scenario is loaded into does not restrict the domain of simulations to robotics or sensors, and the abstractions provided by NetsBlox enable classroom use with minimal programming experience.

ACKNOWLEDGMENTS

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] 2021. NetsBlox website. https://netsblox.org. Cited 2021 March 8.
[2] 2021. Snap! website. https://snap.berkeley.edu/. Cited 2021 March 8.
[3] B Broll, A Lédeczi, et al. 2017. A Visual Programming Environment for Learning Distributed Programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 81–86. https://doi.org/10.1145/3017680.3017741
[4] Á Lédeczi, M Maroti, et al. 2019. Teaching Cybersecurity with Networked Robots. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 885–891. https://doi.org/10.1145/3287324.3287450
[5] Sokratis Tselegkaridis and Theodosios Sapounidis. 2021. Simulators in Educational Robotics: A Review. *Education Sciences* 11, 1 (2021), 11. https://doi.org/10.3390/educsci11010011