ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #40913

Creating Hardware-Accessible Learning with the Robotarium and Block Coding

Mr. Tyler Kinner, Georgia Tech Research Institute

Tyler Kinner is a Research Scientist II at the Georgia Tech Research Institute, where he works on projects related to STEM education, training, and workforce development.

Dr. Sean Wilson, Georgia Institute of Technology

Sean Wilson received the B.A. degree in physics and the B.A. degree in mathematics from the State University of New York College at Geneseo, Geneseo, NY, USA, in 2011, and the M.S. and Ph.D. degrees in mechanical engineering from Arizona State University, Tempe, AZ, USA, in 2017.

He is a Senior Research Engineer with the Georgia Institute of Technology, Atlanta, GA, USA, where he has also served as a Postdoctoral Fellow. His research interests include the areas of remote access robotic hardware and control of multiagent robotic systems.

Avaye Raj Dawadi

Creating Hardware-Accessible Learning with the Robotarium and Block Coding

Abstract:

The Robotarium is a remotely-accessible, swarm robotics testbed using for research and development, as well as educational projects at the undergraduate and graduate level. To enhance the Robotarium's goal of democratizing access to real-world robotics hardware, we sought to bridge the Robotarium with K12 computer science education. In this paper, we will describe the development of a block coding interface, and the piloting of the interface in high school classrooms using a remotely facilitated instructional module.

Introduction:

Computation is of growing importance in K12 education as technology and technology integration advance throughout our lives, and the consequent workforce development needs become more evident [1]. In K12, computer science education may begin as early as elementary school and continue into middle and high school, and incorporates experiences in robotics, physical computing, web design, game design, and software development [2]. Although access has increased over the recent years, the ability for students to take computer science courses in school is not evenly distributed throughout the United States [3].

Our goals with bridging the Robotarium to the K12 audience are two-fold: first, we seek to support robotics education in communities that may not yet have their own robotics learning programs and routines in place, and second, we seek to enable students to learn about robots from a systems-level perspective, wherein robot teams or swarms work together to accomplish a task.

About the Robotarium

The Robotarium is a remotely accessible multi-robot testbed that is free to use for academic or educational purposes [4]. The Robotarium was developed by faculty at a research university in the US southeast and is housed on-campus. Since becoming available for users around the world to use in August 2017, the Robotarium has been used by over 2,750 people from every continent except for Antarctica and executed over 14,500 experiments. Originally, the Robotarium was envisioned as a testbed for academic research. However, over its years of operation, there has been increasing use cases supporting undergraduate and graduate college curriculum. At present, about 40% of submitted remote experiments are for educational purposes.

To use the Robotarium, remote users first create an account on the Robotarium website. This account enables users to submit experiments and receive results back from the Robotarium. Users then can use the Robotarium simulation API available in MATLAB or Python to develop their code that will make the robots on the testbed behave in a desired manner [5]. When users are satisfied with their simulated results, they can submit their code to the Robotarium website where it will be put into a first-come-first-served queue to be executed by the real robots on the Robotarium. After the experiment concludes, users will receive an e-mail instructing them to return to the Robotarium website to view/download a video of the experiment as well as any data they saved.

Methods:

Block coding is a popular tool for teaching younger students the basics of programming [6]. It eliminates the need for students to grapple with documentation or learn language-specific syntax, and instead offers a graphic method of programming. The graphical nature of block coding supports learners also by providing visual cues for various programming structures, such as the ability to place additional blocks of code inside of loops, spaces for blocks to have a variable placed inside them, and staggered structures for conditional statements [7]. The popularity and familiarity of block coding to our target audience of K12 students and teachers, as well as the visual references within block coding platforms that support learners, were our prime motivators for selecting a block coding interface as the top development priority for bridging the Robotarium with K12 audiences.

Developing the block coding interface

To develop the block coding interface for the Robotarium, we leveraged the Blockly library [8]. Blockly provides a variety of built-in structures useful for early learning in programming. In addition, the library provides the ability for the creation of customized blocks that allow for developers to abstract away the line-coding of standard languages. In our case, the specific blocks of code we abstracted within each block were strings of Python. When users download the code they developed with the block coding interface, the strings of Python are concatenated, and can be uploaded to the standard Robotarium website for submission just like any other experiment written in Python.

The ability to make custom blocks seamlessly with Blockly allowed us to port over specific Robotarium concepts such as initialization and drawing images onto the Robotarium testbed itself. This extensibility is what made Blockly especially powerful for our use case, as we

needed to provide the code for the Robotarium robots to initialize, avoid collision, and return to home. Each of these on their own would likely be an insurmountable barrier to entry for our targeted users. In addition to creating blocks that contained the Python code for core Robotarium functions, we also created blocks for pedagogical purposes. Two categories of blocks were created to provide fodder for learning and instruction:

- Maze blocks that would utilize a projected maze image onto the Robotarium testbed that students could navigate
- Sprite blocks that would utilize a selected image from an image library and project it onto the Robotarium testbed at a user defined location and with a user defined scale

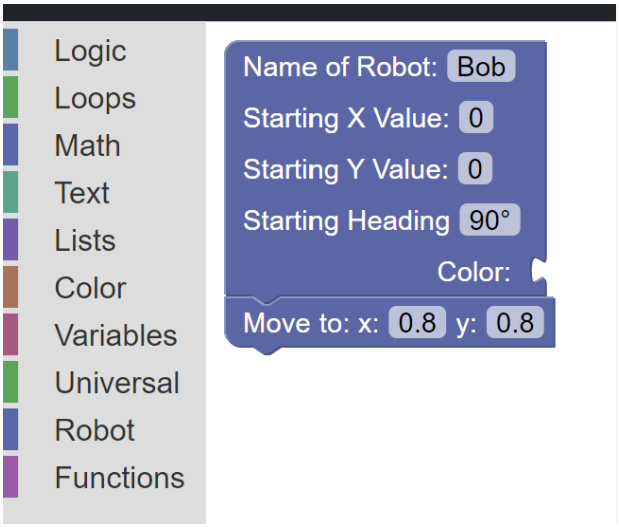

Figure 1 – Block coding interface

User documentation was another priority for development, as we knew students would need the ability to learn about the functionality of each of the blocks available within the block coding interface. A specific documentation page was developed that shows the usage of all the blocks and explains how they work. In addition, within the development workspace, each block's tooltip allows users to easily access the documentation for that specific block, with explanations of block functionality. Another core feature of the block coding interface was the incorporation of the Python simulator for the Robotarium. This simulator, which is available online to general users, requires the creation of a specific local

Python environment [5]. We recognized this would be another difficult barrier to entry for students to engage with the Robotarium. The simulator allows for users to test, debug, and iterate their code more quickly than resubmitting experiments to the Robotarium for each new iteration. It also provides a valuable learning opportunity for users to explore the differences between virtual simulations and running code on physical hardware.

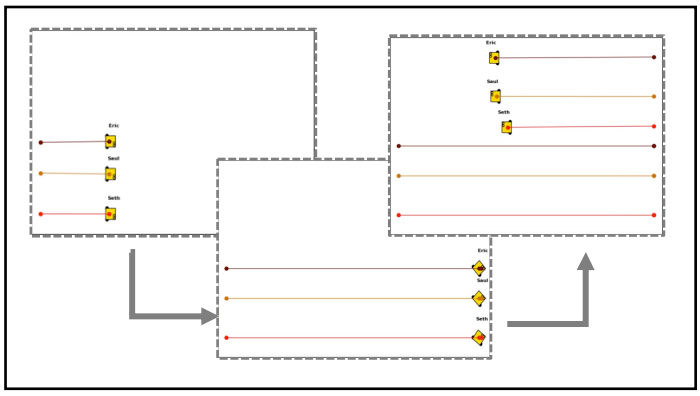

Figure 2 – Sample simulation frames from code generated using the block coding interface

To ensure students had full access to the tools that would benefit their learning, we incorporated the Python Robotarium simulator within the block coding interface. Once students have developed their code, they can select to simulate the code as it would execute on the test bed. The simulator receives the user-assembled Python from the block coding interface, and then executes the provided code and generates the simulation frame by frame (Figure 2). These images are then compiled into an MP4 file using MoviePy [9]. The video is stored in the backend server and a URL that points to the video is returned back as the response from the API, which then allows for the simulation video to be viewed in the frontend.

Flask was used to run the simulator in the backend due to its nature as a lightweight Python development framework that can also be scalable and flexible for our needs [10]. In addition to the abstraction provided by the blocks used by students in the front end of the interact, the simulator abstracts additional details behind the scenes: applying barrier certificates for the robots so that they do not collide with each other or the wall, and handling of other implementation details such as the unicycle position controller. Again, abstraction of the more

sophisticated programmatic components required to submit code to the Robotarium is useful in our pursuit of eliminating barriers to entry in using the Robotarium as a remotely accessible tool for K12 students learning computer science and robotics.

We used Docker to create an isolated, "containerized" environment for the block coding interface that allows for portability and ease of deployment [11]. In addition to making setup of the local environment easier, using containers helps with deployment of the block coding interface to cloud-based virtual machines. This is our preferred method of provisioning the interface for classroom use, as it eliminates the need for students and teachers to handle any local development environment setup or maintenance, and allows students to code through their browsers with access to the block coding interface. After students have used the interface and simulator to create their desired code, the assembled Python code can be downloaded as a .py script from the block coding interface. Students and teachers can then upload their code to the Robotarium using the existing web portal available on the Robotarium website.

Pilot with high school classrooms

To pilot the use of the Robotarium with high school students, we developed a 10-day module that introduced foundational topics in programming through the use of the Robotarium. The first week of the module was delivered remotely via Zoom from university instructors, and the second week was facilitated by teachers in the classroom using the resources provided in the module. As students learned foundational topics through week one, they would practice their understandings through assembling code and then simulating the execution of the code by the robot using the built-in simulator. At the end of the first week, students were tasked with applying their knowledge to craft a solution to the problem, "*How do you mow your lawn in the shortest amount of time?".* Within this problem, students treated the Robotarium test bed as a "lawn" and

had to code multiple robots concurrently for the ultimate goal of mowing the lawn in the shortest amount of time.

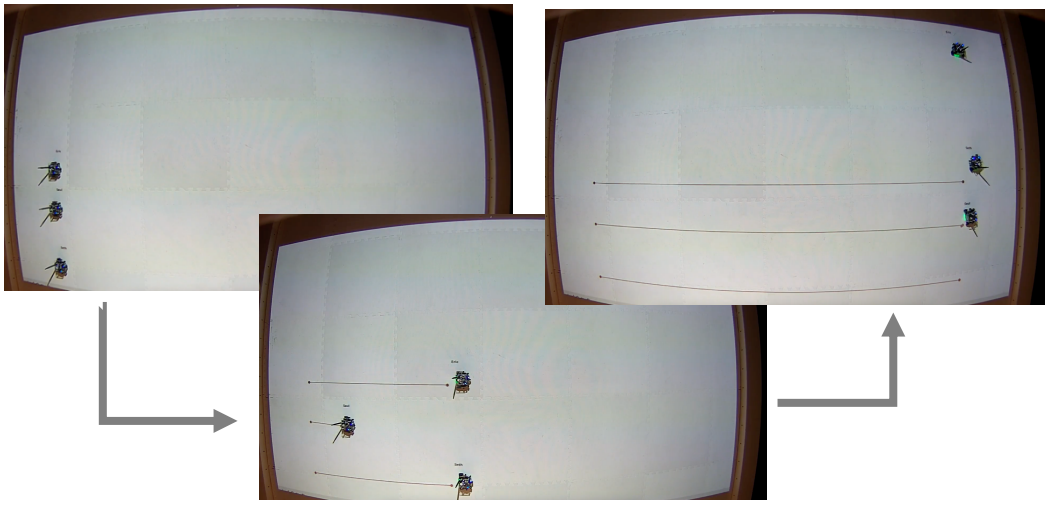

Figure 3 – Student work submitted for the simulated lawn mowing

In the second week, students extended their learning with a variety of projects for self-selection, such as simulating a crop harvest while avoiding obstacles and orchestrating a dance of Robotarium robots in the theme of a "drone lightshow". Students also had time to explore the several puzzles programmed into the Robotarium for additional practice navigating the robots across the testbed using point or turtle movement schemes. Throughout the pilot and at the conclusion, we solicited feedback from participating teachers on the student learning module, the block coding website, and the experience of using the Robotarium for learning in the high school setting.

Module Feedback

As mentioned previously, the goal of the module was to use the Robotarium as a vehicle for teaching introductory coding concepts. The module was primarily implemented in introductory high school courses in computer science, which meant that many students had little prior coding experience. Feedback around the module reflected this, with much of the feedback reporting that students had difficulty managing the learning of introductory coding concepts and the subsequent application of those concepts to the Robotarium. In particular, students struggled with applying the coordinate grid system with which robots are positioned and moved across the testbed. Teachers reported that while students were able to meet the core learning objectives of understanding loops and variables, students struggled to turn their pseudocode into functioning code on the testbed because of difficulty in understanding and navigating the coordinate grid system on the Robotarium testbed. This feedback suggests the need for instructional scaffolds for students to better understand and work within the coordinate grid system, as well as incorporating dedicated instruction or tutorials on the coordinate grid system into experiences involving the Robotarium.

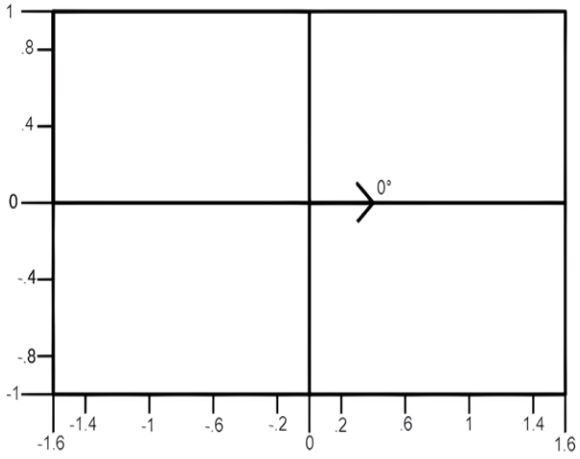

Figure 4 – Graphic embedded on the block coding site to help students understand the coordinate system for the Robotarium

Block Coding Site Feedback

During the implementation, the block coding site's built-in simulator was unable to handle the load of multiple classrooms attempting to simulate the Python code concurrently. This was due to an underdeveloped implementation using serverless architecture to handle the load volume.

Aside from technical feedback that has informed future development and implementation plans, this feedback also strongly suggested that students were using the simulation feature as anticipated to test their code, troubleshoot issues, and iterate in attempts to solve problems.

Overall Feedback

Teachers reported that students were engaged and motivated by the Robotarium and the premise that they were coding real-world hardware operated by a flagship research institution. Teachers also reported that they themselves were learning new concepts as part of the implementation of the modules within their classrooms, and desired more professional learning to help them feel empowered to engage with the Robotarium and the instructional module.

Discussion:

The application of the Robotarium to K12 education is a novel way to create access for students to learn coding and robotics, including multi-agent robotics. The block coding interface provides access to students across grade levels, including those in elementary school, and removes the barrier to entry with understanding the Python code required to initialize the robots, return them to home, and prevent robot collisions. The incorporation of the built-in simulator allows for students to engage in coding best practices through testing and iterative development.

Feedback on the implementation of the Robotarium and our 2-week module in high schools suggests that the Robotarium is indeed a useful tool at engaging students in learning coding, and generates interest and enthusiasm in computer science and robotics learning among students. In the future, we aim to continue to refine both the block coding interface and associated curricular resources to continue to expand access to computer science and robotics learning with the Robotarium. We believe that the extension of the Robotarium to be accessible for computer science education in K12 can serve as a blueprint for institutes of higher education

to more deeply engage in K12 computer science education and outreach through access for students to work with facilities and equipment otherwise unavailable to schools.

References:

[1] Bureau of Labor Statistics (September 6, 2023). *Computer and Information Technology Occupations* [Online]. Available at: https://www.bls.gov/ooh/computer-and-information-technology/home.htm

[2] K12CS (November 2023). *K-12 Computer Science Framework* [Online]. Available at: https://k12cs.org/

[3] Code.org (2022). *2022 State of Computer Science Education* [Online]. Available at: https://advocacy.code.org/2022_state_of_cs.pdf

[4] S. Wilson and M. Egerstedt, "The Robotarium: A Remotely-Accessible, Multi-Robot Testbed for Control Research and Education," *IEEE Open Journal of Control Systems, 2022,* vol. 2, pp. 12-23, doi: 10.1109/OJCSYS.2022.3231523.

[5] *Robotarium Python Simulator* [Online]. Available at: https://github.com/robotarium/robotarium_python_simulator

.

[6] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," in P*roceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07),* New York, NY, 2007, pp. 223–227. DOI: 10.1145/1227310.1227388

[7] Hil H. Dwyer, et al., "Fourth Grade Students Reading Block-Based Programs: Predictions, Visual Cues, and Affordances," in *Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15),* New York, NY, 2015, pp. 111–119. DOI: 10.1145/2787622.2787729.

[8] Google (Nov. 2023). *Blockly* [Online]. Available: https://developers.google.com/blockly

[9] Moviepy (Nov. 2023). *moviepy – PyPI* [Online]. Available: https://pypi.org/project/moviepy/

[10] Pallets (Nov. 2023). *Flask* [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/

[11] Docker Inc. (Nov. 2023). *Docker* [Online]. Available: https://www.docker.com/