A Free and Open Web-Based IDE for the Sphero Bolt

Simon Haller-Seeber [a], Christopher Kelter, Marko Zarić, and Justus Piater [b]

Department of Computer Science, University of Innsbruck
{simon.haller-seeber, christopher.kelter, marko.zaric, justus.piater}@uibk.ac.at
Technikerstr. 21a, 6020 Innsbruck, Austria

Abstract. The Sphero Bolt robot is an educational tool designed to make learning about coding and robotics engaging and hands-on. However, existing software environments present challenges for classroom use, particularly due to limited support for open platforms. We developed a free, browser-based IDE for the Sphero Bolt hardware. It is built on top of Node.js and WebAssembly, using Pyodide. The IDE enables direct interaction with the robot not requiring additional software installations. This paper outlines the technical and educational contributions of our IDE, including an example workshop on behavioral control concepts with practical programming exercises. Inspired by Valentino Braitenberg's vehicles, we demonstrate how students can use the robot's light sensors to simulate reactive behaviors such as light avoidance and attraction. By removing technical barriers and providing a seamless portable coding environment, this IDE lets students and educators focus on the creative and exploratory aspects of programming and robotics.

Keywords: educational robotics, artificial intelligence, web IDE, free software

1 Introduction

The STAIR (Software Testing AI Robotics) Lab is a state-of-the-art facility designed to provide students and educators with hands-on experience in the fields of artificial intelligence and robotics. The lab facilitates activity-based teaching, offering a range of workshops and learning materials to enhance educational experiences for students and teachers (Haller-Seeber et al., 2022).

As part of the Media, Inclusion & AI Space of the INNALP Education Hub, the STAIR Lab contributes to a broader educational ecosystem that explores topics such as inclusion, accessibility, assistive technologies, diversity, and media culture in the context of schooling and teaching (Bouvier et al., 2025).

Several platforms already exist that integrate robotics with educational programming. The Open Roberta framework (Jost et al., 2014), for instance, combines block-based programming with partial simulation capabilities and direct

[a] https://orcid.org/0000-0002-1538-5906
[b] https://orcid.org/0000-0002-1898-3362

programming of physical robots. Similarly, the MakeCode platform, especially when paired with the BBC Micro:Bit, provides basic visualization features while enabling the programming of physical devices. Cardenas et al. (2023) describes a recent example of integrating this platform with AI applications.

A detailed comparison of various programming environments for educational robotics—spanning online/offline, block-based, and text-based approaches—is presented in Table 3.2 (Haller-Seeber, 2024, p.11), reproduced in the Appendix. This analysis highlights the strengths and trade-offs of different platforms, underscoring their potential for robotics and AI education. It also shows that only a limited number of platforms combine installation-free access with the ability to interact with physical devices, particularly for younger children using block-based programming environments. Additionally, it reveals the need for more accessible IDEs that prioritize ease of use and hands-on engagement in educational robotics tailored to children.

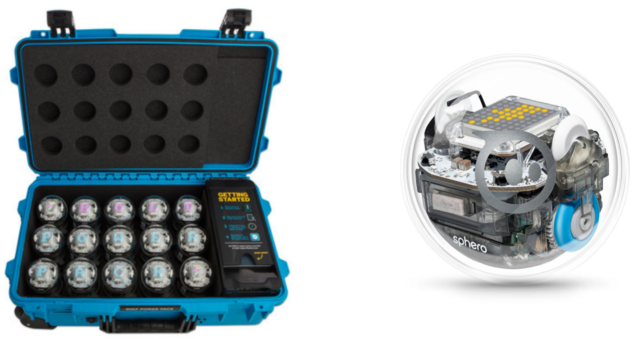


Fig. 1. Sphero Bolt Power Pack containing 15 robots and a single Sphero Bolt. Images taken from https://sphero.com.

The Sphero BOLT (see Fig. 1) is an educational robot that offers an engaging, hands-on approach to learning coding and robotics. The programmable robotic ball is designed to teach coding, problem-solving, and STEAM concepts through interactive activities. Equipped with an IMU, light sensor, LED matrix, and Bluetooth connectivity, the Sphero BOLT supports block-based and JavaScript programming. It has been used in classrooms for coding lessons, physics demonstrations, and robotics challenges, as well as in after-school programs and camps targeting underrepresented groups in STEAM (Mack et al., 2024). However, its current software development environment poses challenges for classroom use. The absence of an official SDK for Linux and the lack of open-source alternatives make it difficult for teachers and students to integrate the robot seamlessly, especially on devices that do not support proprietary SDK installations.

2 Motivation

Our primary goal is to eliminate the need for students or educators to install SDKs on personal computers or tablets, which can be impractical and inconsistent in a classroom setting. On school computers, software configurations are normally outside of our control. Each additional installation step can be time-consuming and might shift the focus away from our teaching goals and disrupt the learning process. Therefore, an installation-free, portable solution is essential—particularly in schools where flexibility and accessibility are paramount.

The Sphero Bolt robot is a viable platform to introduce key concepts from Cybernetics and Behavioral Control, which are prominent in robotics and artificial intelligence. Combined with an easy-to-use IDE, educators can focus on fostering creativity and problem-solving skills in programming while learning about robotics concepts.

To create an accessible and efficient development environment for this robot, we aimed to eliminate the technical and infrastructural barriers. Existing tools, including an open-source Python library for the Sphero BOLT available on GitHub (Wang and Feliciano, 2024), provided a starting point but required significant modification to align with our goals of ease of use, portability, and minimal setup. Motivated by these limitations, we developed a lightweight, web-based IDE utilizing WebAssembly, capable of running directly in any Chromium-based browser (e.g., Chrome, Edge, Opera) without installation or reliance on network connectivity. The IDE functions entirely on the client side, independent of server infrastructure, not storing any user data locally or remotely, thereby addressing common privacy and security concerns in educational settings.

Given our focus on browser-based deployment, we encountered compatibility limitations due to the lack of Web Bluetooth API support in Firefox and Safari.[1] Their stance against implementing this API stems from concerns about the potential risks of granting Bluetooth access to arbitrary websites, given the uncertainty around device readiness and the difficulty of managing generic API interactions securely. By contrast, Chromium browsers address these risks through a permission model that requires explicit user approval for each Bluetooth connection, enhancing the security and control of device interactions.

To overcome the browser support gap, we initially thought of introducing a Bluetooth bridge, similar to the approach used by Mind+[2], which enables broader compatibility via a native application. However, this would compromise the IDE's lightweight, installation-free architecture and impose maintenance overhead for cross-platform support. It would potentially undermine its suitability for classroom environments where simplicity and immediacy are important, especially if one wants to, i.e., visit different schools doing workshops.

Furthermore, the IDE's local execution model ensures stable and consistent performance. It is unaffected by network conditions, which can be an advantage

[1] For an up-to-date list see https://caniuse.com/web-bluetooth. The Mozilla dev team also provides a statement on the Web Bluetooth API at https://mozilla.github.io/standards-positions/#web-bluetooth.

[2] https://mindplus.cc/en.html

in diverse educational contexts. Our solution prioritizes ease of access, privacy, and a consistent user experience. This allows learners to focus entirely on the creative aspects of programming and robotics, fostering a more engaging and productive educational experience.

2.1 Background

This section introduces the key concepts that can be taught implicitly through carefully crafted exercises that use Sphero Bolt robots' movement and sensing capabilities in practical teaching sessions.

Cybernetics provides a framework for analyzing feedback mechanisms in various systems using universal mathematical methods. This approach typically begins with creating models that simulate a phenomenon's behavior under simplified conditions. Valentino Braitenberg's **vehicles** exemplify this methodology. These vehicles are minimalistic yet capable of demonstrating complex behaviors, making them iconic models in the field of cybernetics.

Behavioral control refers to a modular and flexible approach to robotics, where small programs—called **behaviors**—read sensor inputs and control actuators. Each behavior is designed to perform a **simple task**, such as obstacle avoidance, line following, or speed maintenance. A key advantage of behavioral control systems is their modularity: Behaviors are independent yet can function cooperatively, allowing the system to adapt to environmental changes and perform complex tasks. For instance, a robot using behavioral control might simultaneously avoid obstacles while following a light source, enabling sophisticated, emergent behavior.

Braitenberg vehicles (Braitenberg, 1984) are simple autonomous agents that move based on sensor inputs. Each vehicle consists of two motor-driven wheels and sensors, typically for detecting light intensity. The sensor outputs influence the motors, either positively or negatively, depending on how the sensors are connected to the motors.

Despite their simplicity, the placement and wiring of the sensors can result in diverse and seemingly intelligent behaviors of an vehicle, such as:

1. **Fear**: moving away from a light source.
2. **Aggression**: moving to it, increasing speed when approaching.
3. **Love**: approaching a light source but slowing down as it nears the source.
4. **Exploration**: avoiding light but slowing down while exploring illuminated areas.

By introducing multiple light sources or attaching light sources to the vehicles themselves, it is possible to create more intricate and dynamic behaviors. Similarly, integrating additional sensors, such as infrared proximity sensors, allows the vehicles to respond to diverse environmental conditions. This demonstrates how **simple rules and components** can produce complex, emergent, and seemingly intelligent behaviors.

Fig. 2. Long night of research 2024 STAIR-Lab Demos

3 Challenges

To develop a browser-based ID for the Sphero BOLT, we explored several options. One approach involved executing Python code within a Docker container on the local machine, while the alternative focused on running Python code directly in the browser using WebAssembly. Given the educational context and the limited technical expertise of the target audience, the WebAssembly-based approach was considered more suitable, as it eliminates the need for specific setup and resource installation. After careful evaluation, we identified three potential tools. The first, Emscripten[3], is a complete compiler toolchain for WebAssembly that enables the transformation of existing projects written in general-purpose programming languages into a WebAssembly executable. The second tool, Pyodide[4], is a framework that includes a compiled version of CPython for WebAssembly. Pyodide supports nearly all Python features and provides a JavaScript API to enable Python code execution in the browser in a lightweight way. Additionally, it includes a package management system that allows users to import packages from PyPI (The Python Package Index) or integrate custom Python packages. The third tool, Py2wasm/Nuitka[5], is a Python-based compiler that converts Python code into a WebAssembly executable. Although Py2wasm presents potential as a lightweight alternative, its current developmental limitations outweigh its advantages. Moreover, it necessitates compilation before execution in a browser environment. Emscripten, although powerful, lacks the specificity and ease of use required for Python-centric applications. Pyodide's maturity and robustness include features and compatibility make it the most suitable choice for our project, enabling the creation of a browser-based IDE that facilitates seamless interaction with the Sphero Bolt robot.

The following section shows our approach to designing and implementing a free, open-source Integrated Development Environment (IDE) tailored for the Sphero Bolt. Our solution emphasizes portability, eliminating the need for additional software installations and providing an accessible, efficient, and collaborative coding environment for classrooms. We also give one example workshop to evaluate during the upcoming year.

[3] https://emscripten.org
[4] https://pyodide.org
[5] https://github.com/wasmerio/py2wasm

3.1 Technical Implementation

Key steps in our implementation process included:

Public Demonstration and Feedback We directly tested the University of Pennsylvania's Python Sphero v2 library (Wang and Feliciano, 2024) developing a Braitenberg Vehicle demo for the *Long Night of Research* event (see Fig. 2), gathering valuable oral feedback from the general public about the demonstration and robot interactions.

Rewrite of the Sphero v2 API The Sphero v2 Python library offers a way of connecting to different Sphero robots using Bluetooth. The implementation's use of threads prevents it from being directly utilized with our WASM Python interpreter because multithreading support has not been added. To make use of WASM support of asyncio, we rewrote the partially modified asyncio version of the Sphero v2 library.

Access to the Bluetooth Web API in Pyodide In addition to the limitation that Python in WebAssembly does not support threads, there is also no direct access to the Bluetooth API. However, Pyodide has a JavaScript bridge that allows the execution of JavaScript source code in Python. For this reason, we implemented a new Bluetooth adapter for the Sphero v2 asyncio API. Using the JavaScript Bridge, it provides the same methods as the previously-used bleak[6] (Bluetooth Low Energy Platform Agnostic Client) API. This ensures that only the adapter needs to be replaced without any further changes to the source code of the Sphero v2 asyncio API. Fig. 3 gives an overview of how Bluetooth access was implemented. This implementation results in JavaScript code being executed for each command sent to a robot.

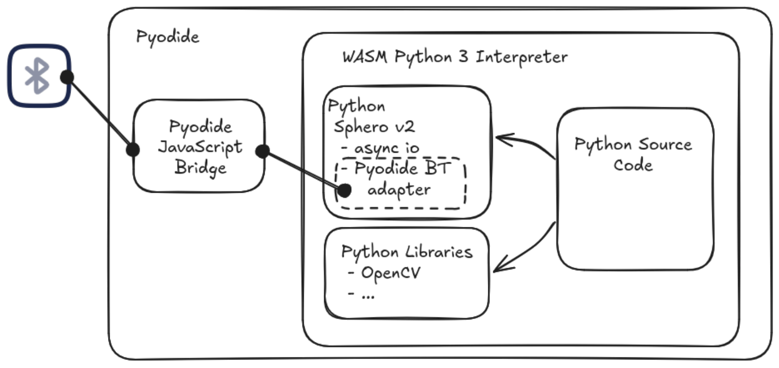


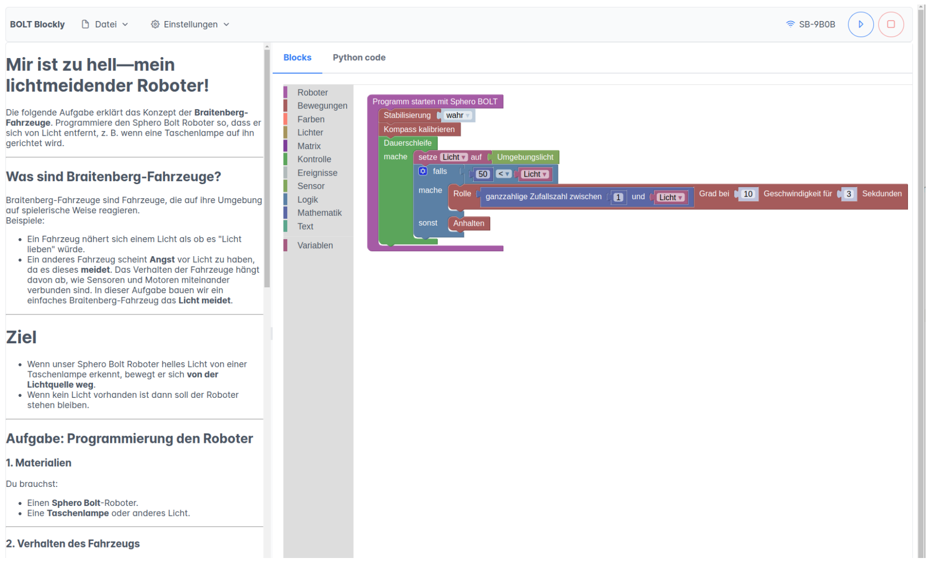
Fig. 3. An overview of the infrastructure that is used to use the Bluetooth Web API.

This process resulted in a fully functional, browser-based IDE capable of interacting with the Sphero Bolt, making programming and robotics accessible across diverse environments without requiring additional software installations.

[6] https://github.com/hbldh/bleak

4 Example Workshop

This section illustrates an example workshop that leverages our web-based IDE to implement light-avoidance behavior in the Sphero Bolt robots. Fig. 4 shows a screenshot of the IDE with an integrated exercise sheet on the left and the coding environment on the right. The exercise sheet can be uploaded as a Markdown file to the IDE, enabling seamless integration of interactive educational content. To solve the programming exercise, students can choose from two distinct editors, a block-based editor and the Monaco Editor[7], which supports Python. These editors empower users to solve tasks suited to their level of expertise. Additionally, the settings menu allows users to connect and configure a Sphero Bolt robot.



Mir ist zu hell—mein lichtmeidender Roboter!

Die folgende Aufgabe erklärt das Konzept der **Braitenberg-Fahrzeuge**. Programmiere den Sphero Bolt Roboter so, dass er sich von Licht entfernt, z. B. wenn eine Taschenlampe auf ihn gerichtet wird.

Was sind Braitenberg-Fahrzeuge?

Braitenberg-Fahrzeuge sind Fahrzeuge, die auf ihre Umgebung auf spielerische Weise reagieren.
Beispiele:

- Ein Fahrzeug nähert sich einem Licht als ob es "Licht lieben" würde.
- Ein anderes Fahrzeug scheint **Angst** vor Licht zu haben, da es dieses **meidet**. Das Verhalten der Fahrzeuge hängt davon ab, wie Sensoren und Motoren miteinander verbunden sind. In dieser Aufgabe bauen wir ein einfaches Braitenberg-Fahrzeug das **Licht meidet**.

Ziel

- Wenn unser Sphero Bolt Roboter helles Licht von einer Taschenlampe erkennt, bewegt er sich **von der Lichtquelle weg**.
- Wenn kein Licht vorhanden ist dann soll der Roboter stehen bleiben.

Aufgabe: Programmierung den Roboter

1. Materialien

Du brauchst:

- Einen **Sphero Bolt**-Roboter.
- Eine **Taschenlampe** oder anderes Licht.

2. Verhalten des Fahrzeugs

```
Programm starten mit Sphero BOLT
  Stabilisierung wahr
  Kompass kalibrieren
  Dauerschleife
  mache setze Licht auf Umgebungslicht
        falls 50 < Licht
        mache Rolle ganzzahlige Zufallszahl zwischen 1 und Licht Grad bei 10 Geschwindigkeit für 3 Sekdunden
        sonst Anhalten
```

Fig. 4. Sphero Bolt Block-based Programming Interface (with example instructions in German) accessible at https://stair-lab.uibk.ac.at/sphero-blockly/. The Code is available at https://git.uibk.ac.at/informatik/stair/sphero-blockly.

4.1 Scope and Goal

As a case study, we showcase **Braitenberg vehicles** using the ambient light sensor of the Sphero Bolt robot[8]. This choice serves multiple purposes:

– To introduce the concept of **Cybernetics** and **Behavioral (or reactive) control**, foundational topics in robotics and artificial intelligence,

[7] https://microsoft.github.io/monaco-editor/
[8] https://edu.sphero.com/cwists/preview/18144x

- to illustrate how our IDE can be combined with the classic tutorials provided by Sphero, and
- to honor Valentino Braitenberg, whose almae matres includes the University of Innsbruck.

The Sphero Bolt IDE and its support for programming Braitenberg like vehicles highlight the educational potential of integrating foundational concepts in cybernetics and robotics with hands-on programming tools. This synergy engages learners and fosters a deeper understanding of how simple rules drive complex systems.

4.2 Programming Exercise

The goal of the practical programming exercise is to use the web-based IDE to implement the following scenario with two behavioral possibilities:

1. The Sphero Bolt detects bright light and moves away from the light source.
2. There is no light, and the Sphero Bolt remains still.

The students will get an exercise sheet with a step-by-step guide (Fig. 4 on the left). The three main programming steps are reading the light sensor value, setting a threshold condition for the light sensor, and making the robot move away from the light if the threshold is exceeded. Upon completion, the students have to test their program by shining a flashlight on the robot. Here, they learn to evaluate if the goal was achieved and are tasked to make adjustments until the desired behavior can be demonstrated iteratively.

5 Outlook

We are actively working on several future improvements and applications to enhance the IDE's capabilities and broaden its educational impact:

Multilanguage Support: This feature is relatively straightforward to implement and will be prioritized next, with at least English language support being added initially.

Python WebAssembly Thread Support: Currently, the IDE uses `asyncio` for managing concurrent tasks. We aim to incorporate WebAssembly thread support to enable more efficient execution and facilitate more complex, interactive applications.

Multi-Robot Support: Currently, the IDE can connect to only one robot. We plan to extend its functionality to support multiple robots simultaneously, enabling more complex, maybe even swarm robotics scenarios and collaborative learning experiences.

OpenCV and Camera Integration: By introducing visual programming blocks for OpenCV and enabling USB camera access, students will be able to explore advanced robotics topics, such as computer vision and image processing.

Workshops and Evaluations: We plan to conduct workshops with children using the IDE and evaluate their experiences. These workshops will take place as part of the teaching and learning labs within the INNALP Education Hub. Additionally, the workshops are organized within the framework of the FFG-funded project AIRE. Feedback from these sessions will provide valuable insights into usability and educational impact, guiding further platform refinement.

By addressing these areas, we aim to enhance the IDE's functionality and continue promoting accessible, hands-on robotics and AI education for learners of all ages.

References

Bouvier, F., Gleirscher, L., Haller-Seeber, S., Hug, T., Kaiserer, M., Sonntag, M.: Innovative Lehr- und Lernansätze in Lernlaboren: Einblicke in den Media, Inclusion & AI Space des INNALP Education Hub. Medienimpulse 63 (3 2025), https://journals.univie.ac.at/index.php/mp/article/view/9253

Braitenberg, V.: Vehicles: Experiments in Synthetic Psychology. MIT Press, Cambridge, MA, USA (1984)

Cardenas, M.I., Molas, L., Puertas, E.: Artificial Intelligence with Micro:Bit in the Classroom. In: Robotics in Education. LNNS, vol. 747, pp. 337–350. Springer (Oct 2023)

Haller-Seeber, S., Gatterer, T., Hofmann, P., Kelter, C., Auer, T., Felderer, M.: Software Testing, AI and Robotics (STAIR) Learning Lab. In: Robotics in Education. LNNS, vol. 515, pp. 182–189. Springer (July 2022)

Haller-Seeber, S.M.: Innovative approaches to developing educational robots. (Sept 2024), Master Thesis, https://ulb-dok.uibk.ac.at/urn/urn:nbn:at:at-ubi:1-161112

Jost, B., Ketterl, M., Budde, R., Leimbach, T.: Graphical programming environments for educational robots: Open roberta - yet another one? In: 2014 IEEE International Symposium on Multimedia. pp. 381–386 (2014)

Mack, N.A., Adeleke, M.B., Ballou, E., Davis, D., Ingram, V., Cox, K.: Breaking Stereotypes and Feeding the STEM Pipeline. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. p. 771–777. SIGCSE 2024, Association for Computing Machinery, New York, NY, USA (2024), https://doi.org/10.1145/3626252.3630793

Wang, H., Feliciano, E.: spherov2.py, an unofficial Python library for Sphero toys that supports its Version 2 Bluetooth low energy API. https://github.com/artificial-intelligence-class/spherov2.py/releases/tag/0.12.1 (2024)

Appendix

Editor	Online/Web	Hardware Platform	Language	Simulator	Compile/Flash	Open-Source Free-Software	User-Level Age
Adafruit webIDE	No[9]	Raspberry PI, Beaglebone	Python, Ruby, JavaScript and more	No	Runs live on Board	Yes (AGPLv3)	10+
Arduino Blocks	Semi	Arduino, ESP32	Blocks, C	No	Yes (with Connector)	No	6+
Ardu Block	No (Addon)	Arduino	Blocks, C	No	No	Yes (GPLv3)	6+
Arduino IDE	No	Arduino	C	No	No	Yes (AGPLv3	10+
Arduino Web Editor	Yes	Arduino	C	No	Yes (with Plugin)	Yes (AGPLv3	10+
BlueJ and Greenfoot	No	Raspberry PI, Beaglebone	Java (for Beginners)	No	Runs live on Board	Yes (GPLv2	10+
BIPES	Yes	ESP32, Arduino, Raspberry Pi, Lego EV3, and more	Blocks, Python	No	Yes	Yes (GPLv3)	10+
Colobot	No	Programming Robot Game	C(like)	Yes	No	Yes (GPLv3	12+
EVCdevelop	No	EV3	C	No	No	Yes (MIT)	10+
Gearsbot	Yes	Configurable / no specific	Blocks, Python	Yes	No	Yes (GPLv3)	6+
Keil Studio (successor of MBED)	Online and Offline	More than 120 platforms	C	No	Yes	No	12+
Micro:Bit Blocks Editor	Yes	Micro:Bit	Blocks, JavaScript	Yes	Yes	Yes (MIT	6+
Micro:Bit MicroPython	Yes	Micro:Bit, Calliope (limited)	Python	No	Yes	Yes	10+
Mind+	Yes[10]	Arduino, Micro:Bit, ESP32 & more	Block	No	Yes	Yes (App is GPLv2)	6+
Mu	No	Raspberry PI, Micro	MicroPython	No	Yes (offline cross-compiling)	Yes (GPLv3)	10+
Open Roberta Lab	Yes	NXT, EV3, Open Roberta Sim, Calliope, Micro , BOB3, NAO, Bot'n'Roll	Blocks	Yes	Yes	Yes (Apache2)	6+
S4A - Scratch4Arduino	No (Addon to Scratch)	Arduino Diecimila, Duemilanove and Uno	Scratch	Yes	No	Yes (MIT)	8+
TinkerCad	Yes	Arduino, Circuits, and 3D Designs	Block Type	Yes	Download for Arduino IDE	No	8+
VS Code using Plugins	No	More than 120 platforms	Various (no Block-Type)	No	Yes	No & Yes (source, MIT)	12+

Table 1. Extended comparison of programming environments suitable for educational robotics, original from Haller-Seeber (2024, Table 3.2, p.11).

[9] Runs a Server on the specific Hardware-Platform

[10] There is a App for Windows, Mac and Linux which support a wider variety of Hardware. The Web-Version does currently not support Hardware connections when using Linux OS.