Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 21 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i21.57355

PAPER

Empowering Computational Thinking through Personalized Robotics: The Odysseus Model

George Lagogiannis (✉)

Agricultural University of Athens, Athens, Greece

lagogian@aua.gr

ABSTRACT

This paper presents Odysseus (Ody), a low-cost, 3D-printable, open-source robotics kit designed to support the teaching of programming, algorithms, and data structures across educational levels. Ody features a dual-mode programming environment, block-based coding for beginners, and Python for advanced learners, enabling a smooth progression in computational thinking. To further support novices, Ody introduces tailored abstract blocks that allow educators to adjust both the level of difficulty and the level of abstraction. Building on this, we extend Vygotsky's Zone of Proximal Development by adding abstraction as a second dimension, and we propose the Dual-Threshold Scaffolding Model. We also introduce the Functional–Pedagogical Mobility Framework, which positions Ody and comparable technologies in terms of mobility. To assess Ody's educational impact and user acceptance, we conducted an evaluation with learners using a Technology Acceptance Model (TAM)-based questionnaire. The findings indicate strong engagement and positive reception, suggesting that Ody is a flexible and promising platform for computer science education from foundational through advanced levels.

KEYWORDS

educational robotics, Arduino, Firmata, Snap4Arduino

1 INTRODUCTION

In contemporary educational systems, robots have become fundamental tools for improving the quality of STEAM (Science, Technology, Engineering, Arts, and Mathematics) education [2, 3, 7, 54, 81], enabling students to combine critical thinking, problem-solving skills, creativity, and teamwork in accessible and participatory environments. As robots become more prevalent in the classroom, schools often form robotics clubs where students design and build robots for competitions, scientific experiments, solving mathematical problems, or simply for fun.

In this paper, we present *Odysseus* (Ody), a new educational robotics kit designed to address the challenge of teaching programming, algorithms, and data structures

Lagogiannis, G. (2025). Empowering Computational Thinking through Personalized Robotics: The Odysseus Model. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(21), pp. 49–76. https://doi.org/10.3991/ijim.v19i21.57355

Article submitted 2025-06-24. Revision uploaded 2025-08-25. Final acceptance 2025-08-26.

to learners of all ages. Ody is constructed entirely from low-cost hardware and open-source software, providing users with maximum flexibility while keeping costs to a minimum [77]. Its plastic components can be 3D-printed at home, while custom electronic parts (designed specifically for the kit and not available in standard stores) are freely accessible and open for modification. Unlike fixed-form robots, Ody adopts a modular approach: electronic units are enclosed in plastic casings that can be combined to build robots of various forms, such as cars, humanoids, or robotic arms. Safety is ensured by design, preventing short circuits or other failures. Finally, to support use across all educational stages, Ody's programming environment integrates specialized commands tailored to different levels of education.

2 EDUCATIONAL ROBOTS, OPEN-SOURCE, 3D PRINTING, AND LEARNING OUTCOMES

Since their introduction into classrooms, educational robots have been the subject of extensive research across grade levels and disciplines [15]. Evidence consistently suggests that they enhance student learning outcomes [4, 85], with demonstrated benefits in engagement and performance across social sciences and cognitive tasks [86].

In STEM education, the impact has also been positive. For example, a meta-analysis of robotics interventions in primary education found moderate learning benefits, despite variability in implementation and reporting [79]. Robotics activities have been shown to improve mathematical abilities and scientific inquiry skills [1, 34], as well as enhance performance in subjects such as physics and geography [20]. A combined approach that integrated educational robotics with visual programming (using Lego Mindstorms and App Inventor) to support the teaching of basic programming structures yielded positive outcomes [57]. These findings are reinforced by a broader meta-analysis [86], which confirmed that robotics interventions lead to measurable gains in STEM knowledge and technical skills [18].

While many studies highlight the benefits of educational robotics, others suggest that not all programs result in direct academic gains [6, 30]. The extent to which educational robotics can enhance motivation also remains uncertain [72]. In [32], it was reported that middle school students who participated in robotics competitions showed no significant difference in academic performance. This suggests that competition-based learning alone may not be as effective as structured educational interventions that integrate robotics into the curriculum. Similarly, it was shown [9, 36] that educational robotics can sometimes divert students' attention from core STEM learning goals, increasing cognitive load and leading to negative learning effects. The duration of the robotics course is also a parameter that can influence the educational impact. In particular, the course can be too short [53] or too long [87]. Moreover, the lack of a clear pedagogical framework in much of the existing research [39] raises further skepticism. Taken together, these findings emphasize that simply providing an educational robot is insufficient; meaningful outcomes depend on well-designed curricula and strong instructor support [55]. Educators thus play a central role in ensuring that robotics produces measurable positive effects.

The 3D printable and open-source nature of Odysseus aligns with a growing body of research regarding the use of open hardware in education [52]. By integrating additive manufacturing into the classroom [13, 69], research shows that 3D printing enables tinkering [14], supports innovative curriculum development, and promotes cross-disciplinary research. It also fosters creativity and innovation among learners [52]. Furthermore, participation in open-source software development projects can significantly enhance computer science students' technical skills, employability,

and self-confidence [62, 50]. However, as with educational robotics, the technology itself is not enough for achieving educational gains without well-structured courses and trained educators [69].

3 ON THE HARDWARE-BASED CATEGORIES OF EDUCATIONAL ROBOTS

A variety of educational robotic tools is available on the market, as well as open-source [33, 56]. Any educational robot is based on hardware that executes the robot's code. This hardware may be a microcontroller or a microprocessor.

- In the case of a microcontroller, the code runs on a platform with limited resources, typically available for less than 10 Euros. Users are usually familiar with the name of the board that carries the microcontroller, rather than the microcontroller itself. For example, there are Arduino-based educational robots, built upon an Arduino Uno that contains an ATmega328P 8-bit microcontroller [5], or Micro:bit-based robots, built upon a Micro:bit board that incorporates a Nordic nRF51822 32-bit ARM Cortex-M0 microcontroller [27].
- In the case of a microprocessor, the code runs on a platform that supports an operating system. The platform also provides additional software and hardware resources. Such platforms are comparable to regular computers. They offer a wide range of connectivity options, including Wi-Fi, wired Ethernet, USB, SATA, HDMI, and specialized connections such as cameras and displays. They are suitable for tasks involving computer vision or processing large amounts of data, but they are costly. An example of such a platform is the Raspberry Pi [82].

Apart from the underlying hardware, we can categorize educational robots as follows [56]:

- Programmable robots (also called "non-versatile"): These are "fixed shape" robots. The motors and sensors are installed in advance, and the user cannot add or remove them. They can be wheeled, zoomorphic, or humanoid. Some examples are Beebot, Edison, Ozobot, Thymio, mBot, Blue-Bot, Openbot, and FOSSbot [17].
- Robotics construction kits (also called "versatile"): These consist of a "programmable component" that contains slots for connecting sensors and motors to the input and output pins of the microcontroller board. This component is also compatible with bricks, meaning it can be embedded into any brick construction to form robots of various shapes and purposes. This approach seems more suitable for educational purposes since (a) the robot-building process allows for creativity and fun, and (b) the number of possible scenarios and projects is significantly larger. Examples of educational robots that follow the versatile approach are Lego Mindstorms, Makeblock mBot, Robotis Dream Kits, VEX V5, and Hexbug VEX.

Finally, we can distinguish the educational robotic tools available today based on the role the user plays in the assembly process. The following categories arise:

- Ready to use: These are sold in the market as complete products. The user is unaware of the underlying hardware and is not expected to interact with it. Generally, the user relies on the company that produces the kit for service and parts. If a plastic casing is broken or a piece of electronics (for example, a sensor) malfunctions, the user can only replace it by buying a new part from the company that sells the kit, usually at a higher cost than the cost of the involved hardware.

- Printed at home: These are constructed from boards, motors, and sensors available in the market. Users download the STL files [29] corresponding to the plastic parts, print them at home, and assemble the robot by combining the electronic and plastic parts. An obvious advantage of this approach is that the user can reprint broken plastic parts at home at minimum cost or replace malfunctioning electronic components by purchasing new ones from electronics stores. Furthermore, this way the user has the opportunity to see the hardware up close and better understand "what is inside". Hopefully, this closer look at the hardware may intrigue students to investigate further by getting directly involved with it through research projects.

In this context, Ody can be categorized as a versatile, printed-at-home robotics construction kit.

4 PEDAGOGICAL THEORETICAL FRAMEWORK

4.1 Learning theories and educational robotics

The design and implementation of educational robotics kits are guided by constructionist learning theory [59], which emphasizes learning through active, meaningful construction. In constructionist environments, learners engage not only with abstract concepts but also with tangible materials, leading to deeper understanding through making. Robotics kits exemplify this approach by providing learners with programmable artifacts that respond to their input in real-time, enabling iterative cycles of planning, building, testing, and revising.

In recent years, the constructionist approach has been expanded through the Positive Technological Development (PTD) framework [8], which integrates constructionist principles with applied developmental science. PTD provides a framework for understanding how technology can support the design and use of tools that promote positive behaviors and how, in turn, those behaviors can foster developmental assets. It focuses on six key dimensions: content creation, creativity, collaboration, communication, ethical decision-making, and community building. In this sense, PTD encourages learners not only to build functional tools but also to develop ethically, socially, and cognitively.

Educational robotics aligns well with the PTD framework, as it offers hands-on, collaborative, and creative experiences. This way, learners engage in problem-solving, teamwork, and purposeful design, features that foster not only technical skills but also social and ethical growth. This alignment is most clearly observed in classroom contexts, where teachers scaffold learning not simply by transmitting knowledge, but by orchestrating collaborative, problem-centered activities. In these settings, teachers act as "More Knowledgeable Others" [83], offering just-in-time feedback and guidance that enable learners to tackle challenges slightly beyond their current ability. Such interactions embody the Zone of Proximal Development (ZPD), where competence emerges through supported participation.

4.2 Learning contexts in educational robotics

The Maker Movement emerged from the convergence of open-source hardware, open-source software, and increasingly accessible digital fabrication tools (such as 3D printers and microcontrollers [26, 70]). It has led to the introduction of

makerspaces: environments filled with diverse materials to support creative projects [37, 74]. The movement aligns closely with constructionist theory and resonates with the PTD framework. It also reflects the principles of open-source by promoting transparency, collaboration, ethical decision-making [49], community participation, and knowledge sharing. Educational practices incorporate these values through the concept of 'working in the open' [67], which encourages learners to document, share, and discuss their approaches with a broader learning community. One way these values manifest in practice is through 3D printing, a key tool of the Maker Movement that has gained traction as an educational technology [61, 35], enabling hands-on creativity and personalization.

Recent technological developments allow robotics education to extend beyond the classroom into informal, learner-driven contexts. Mobile learning refers to educational experiences facilitated by portable digital technologies (such as smartphones, tablets, and laptops) that support learning anytime and anywhere, thus dissolving the traditional constraints of time and location [58, 73, 78]. More than just learning on the move, mobile learning also emphasizes contextual flexibility, personal agency, and connectivity, allowing learners to access content, collaborate, and create in authentic, real-world settings [38]. When applied to educational robotics, these principles enable students to engage with robotics kits not only in schools under teacher supervision but also independently or collaboratively at home or in community spaces, supported by digital platforms that offer resources, interactive feedback, and guidance through AI mentors [51, 84].

4.3 Constructive assembly and customization

While the Maker Movement has popularized hands-on creation using accessible tools and prefabricated components, much of this making still relies on fixed materials provided in advance. In educational robotics, these materials have typically consisted of static, pre-manufactured parts provided to students, constraining opportunities for personalization and deeper creative engagement. The integration of 3D printing technology into education has introduced virtual materials, i.e., digital models that educators and learners can fabricate on demand. While this represents a technological advancement, the materials remain essentially static, presented as fixed digital files that learners reproduce rather than reshape. Since students are still required to assemble their robots and constructions from fixed parts, we have chosen the term "constructive assembly" to describe this level of engagement.

Encouraging students to design their materials marks a critical extension of constructionism, as it shifts the focus from static resources to dynamic, learner-generated artifacts. In this context, the act of realizing the materials becomes recursive: learners imagine, design, and fabricate parts in an iterative cycle of creation and reflection. This fact fosters deeper agency and creativity [10], enabling learners to engage with the design process as a mode of expression and problem-solving. When learners share, remix, and collaboratively improve their designs, the approach also supports socially connected learning and aligns with PTD principles. We refer to this richer, more open-ended mode of engagement as "constructive customization."

In the case of Odysseus, the kit's design incorporates these two modes of engagement. Constructive assembly is supported through a library of ready-to-print STL files and step-by-step guides, enabling learners to quickly build functional robots. Constructive customization is realized through the provision of editable CAD models, allowing learners to redesign parts, adapt them to specific challenges, or incorporate aesthetic and functional innovations.

4.4 The Dual-Threshold Scaffolding Model

In the traditional view of Vygotsky's Zone of Proximal Development (ZPD), the educator guides learners along a continuum of difficulty, raising the "difficulty threshold" as competence grows. In Ody's youngest learner context, the educator also manages a second, equally critical dimension: the abstraction threshold. Before students engage with the robot, the educator scaffolds the interaction by calibrating it to respond to simplified commands, making it accessible to novices. By pre-configuring the interface, concealing loops and conditionals, and presenting only concrete, game-ready actions, the educator determines how much of the robot's underlying complexity is revealed.

We call this the Dual-Threshold Scaffolding Model, and it is a two-dimensional extension of the ZPD. One axis governs task difficulty (e.g., sensor integration, algorithmic depth), while the other governs conceptual abstraction (e.g., high-level block commands vs. low-level code). The educator can adjust these independently, enabling, for example, harder tasks at a low abstraction level or simpler tasks with higher abstraction. Over time, the educator raises the task difficulty threshold and lowers the abstraction threshold. This approach echoes Bruner's spiral curriculum [12], which cyclically revisits concepts with varying complexity. It also aligns with the principle of progressive disclosure in Human-Computer Interaction (HCI) [75], revealing complexity in measured increments. The framework creates a dynamic learning space where progression is scaffolded in both challenge and conceptual exposure.

The space created by the model is depicted in Figure 1. Within this space, the starting point for youngsters is somewhere with high abstraction and low difficulty. Learner progress can be achieved in more than one way, as shown in Figure 1. In particular, the educator may choose to follow the route composed of solid or dashed arrows or any other route towards high task difficulty and low abstraction. At any time, the learning process is located at a point in this two-dimensional space. This point is the intersection of the current abstraction (red dashed line in Figure 1) and the current difficulty (blue dashed line in Figure 1).

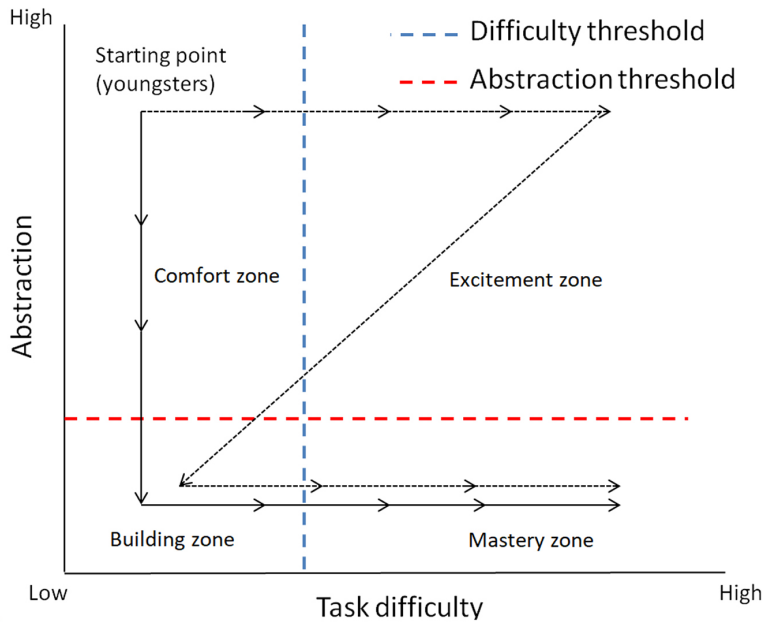

Fig. 1. The Dual-Threshold Scaffolding Model

The two dashed lines divide the plane into four regions, named as follows:

1. Comfort zone: It is the upper left region (low difficulty, high abstraction). The comfort zone is ideal for confidence-building and needs zero or very limited guidance.
2. Building zone. It is the lower left region (low difficulty, low abstraction) and can be seen as an ideal stepping stone for conceptual learning before task difficulty ramps up.
3. Excitement zone: It is the upper-right region (high abstraction, high difficulty). This area contains more challenging tasks, which require underlying logic that is initially hidden. The key challenge when learners enter this region is to reveal just enough elements to guide them toward deeper understanding. Tasks here can generate excitement but also frustration, depending on how much is left visible. Striking the right balance is crucial.
4. Mastery zone: The lower right area (low abstraction, high difficulty). It is the destination that features harder challenges with the underlying logic exposed.

4.5 Levels of mobility: The Functional-Pedagogical Mobility framework

In the landscape of educational innovation, educators and engineers approach mobile learning from distinct perspectives, often working in parallel without a shared language. Yet, as Luckin [41] emphasizes, educational technology design must balance system architecture with learner-centered ecologies. The convergence of technology and pedagogy toward common educational goals is well-recognized, but achieving it requires a conceptual framework intelligible to both communities.

Educators primarily focus on defining learning outcomes, designing curricula, and shaping instructional strategies that foster meaningful learner development. Their perspective centers on what learners need to achieve and how to support their cognitive, affective, and collaborative growth. Engineers, on the other hand, concentrate on building platforms and infrastructures that make these educational goals feasible, scalable, and sustainable. Their lens emphasizes the technical characteristics and functionalities of learning environments, such as interactivity, adaptability, and accessibility.

We propose the Functional-Pedagogical Mobility framework, a layered framework capturing the evolution of mobile learning environments from an engineer's view (centered on technical capabilities and infrastructure) to an educator's view (centered on pedagogical alignment and learner development). The framework maps the deliberate transformation of core concepts (e.g., files into educational material, platforms into digital makerspaces, and AI into intelligent mentorship) to create a shared vocabulary. By integrating these two perspectives within a single framework, the model makes it easier to design, assess, and implement mobile educational technologies.

The first three levels of the framework align with the engineer's view. This is logical because engineers are responsible for building the infrastructure within the framework of their field. From the engineer's standpoint, mobile learning environments are first and foremost technical ecosystems, and the primary concern is whether the platform works reliably, is discoverable, and can be extended or adapted. The levels are:

- Mobility Level 1 (Files and Basic Access): This stage provides raw, open-access resources (i.e., STL/CAD models, firmware code, and schematics), hosted across distributed repositories such as GitHub. While these resources enable reproduction of hardware, they generally lack documentation, pedagogical structure, or learning pathways and are often poorly discoverable due to minimal metadata

or educational tagging. While these repositories ensure technical accessibility, they frequently remain educationally invisible, since they rarely include the structure and support needed to transform raw resources into meaningful learning experiences.

- Mobility Level 2 (Platform and Multimodal Guidance): This stage replaces scattered repositories with a centralized platform that aggregates and organizes resources, improving basic navigation and access. The platform adds instructional supports such as step-by-step assembly guides, annotated diagrams, and multilingual video tutorials, offering learners multiple modes of engagement. These multimodal resources enhance accessibility, personalization, and context-aware learning opportunities, key values in mobile learning. However, the approach still mainly focuses on content, with little guidance on the order of activities or on checking learners' understanding.
- Mobility Level 3 (Programmability and Interactive Environments): At this stage, the platform evolves beyond static content to enable active engagement and direct system control. Learners are no longer limited to consuming resources. Instead, they can design, test, and refine their own creations within interactive environments. Depending on the domain, this might mean experimenting with simulations, manipulating data, or customizing digital models. In educational robotics, for example, learners can write, debug, and improve code directly on connected hardware or within virtual environments. Within a mobile learning framework, AI-based tools such as GitHub Copilot may provide real-time guidance, automated suggestions, and feedback, helping learners design and improve their solutions more effectively. The emphasis here is on technical versatility (i.e., supporting multiple modes of interaction, modular and interoperable architectures, and compatibility with diverse extensions). Pedagogical scaffolding, curricular sequencing, and alignment with educational standards remain secondary concerns.

Levels 1 through 3 focus on the technical capabilities and interoperability of the system rather than its pedagogical design. The transition to Levels 4 and 5 marks a shift in perspective, where the same infrastructure is reframed through an educational lens, integrating structured learning pathways, collaborative creation, and, ultimately, intelligent mentorship.

- Mobility Level 4 (Digital Makerspaces): In this stage, the technical platform evolves into a collaborative learning space where learners, educators, and developers contribute, adapt, and share projects. The environment supports hands-on construction, experimentation, and peer exchange, moving beyond a simple delivery system to a participatory ecosystem. All educational material is developed and validated to comply with established frameworks such as NGSS or ISTE.
- Mobility Level 5 (Participatory Mobile Learning Ecosystem): At this level, the platform incorporates AI-driven mentorship that goes beyond traditional tutoring to address cognitive, metacognitive, emotional, and motivational dimensions of learning. Acting as a More Knowledgeable Other [83], the AI mentor provides personalized guidance, adaptive scaffolding, and affect-aware feedback in mobile and distributed contexts. Unlike the basic AI tools of Level 3, it leverages an educational framework to monitor learner progress and goals, enabling targeted interventions and data-informed support for teachers. Importantly, the mentor sustains learner autonomy by helping students regulate their own learning and pacing through just-in-time assistance [22]. This evolution transforms the platform into a fully adaptive and participatory mobile learning ecosystem.

The Functional-Pedagogical Mobility Framework is graphically depicted in Figure 2.

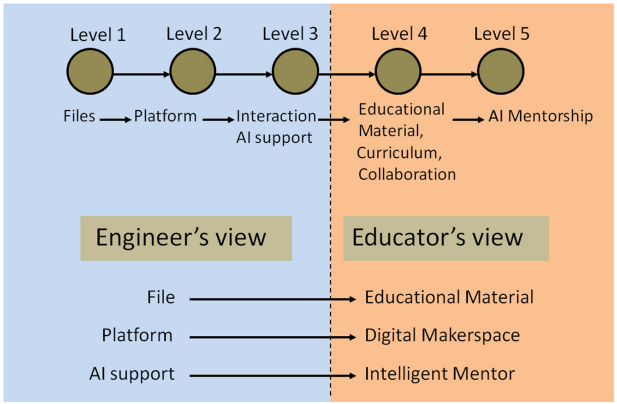

Fig. 2. The Functional-Pedagogical Mobility framework

4.6 Positioning mobile educational robotics technology

While the Functional-Pedagogical Mobility Framework is proposed as a general model for understanding the evolution of mobile educational technologies, it also serves a practical role in assessing the development of specific initiatives. In particular, we can attempt to position well-known educational robotics technologies and platforms within our framework, assuming, at least hypothetically, that they follow its progression. This attempt inevitably carries the risk of interpreting diverse design philosophies and pedagogical models through the lens of our mobility levels. Nonetheless, such positioning can help reveal patterns, identify gaps, and highlight opportunities for both open-source and commercial initiatives.

By definition, all open-source educational robots reach Level 1 through the release of design files, yet very few reach Level 2. Among the many open-source educational robots that can be (not easily) discovered, FOSSbot, Thymio [47], and Edison [46] have reached Level 2, i.e., they are accessible through a web platform. Regarding Thymio and Edison, e-books with educational material are available for students and teachers. Regarding FOSSbot, a YouTube programming course covers all key concepts. Odysseus can be placed somewhere between levels 1 and 2. Here is why: a GitHub repository [40] contains STL and CAD files, thus it has reached Level 1. In advance, a dedicated YouTube channel provides a programming course. What remains for Odysseus to reach Level 2 is the creation of a platform that integrates the existing files and lessons in a more organized and easy-to-access manner. As a general comment regarding the above open-source robots, we can say that they do not lack a pedagogical perspective, and in this sense, all three of them have some characteristics of level 4.

Regarding commercial technology, LEGO Education platforms (e.g., SPIKE Prime, LEGO Mindstorms) typically operate at Level 4. They combine robust, interoperable hardware with structured curricula, teacher training, and clear standards alignment. Their ecosystems resemble digital makerspaces where learners can progress from guided lessons to open-ended design challenges. Similarly, Microsoft MakeCode functions as a cross-platform programming environment that supports the transition from Level 3 to Level 4 by providing accessible coding tools, project-based resources, and community sharing for compatible hardware (e.g., micro:bit, LEGO, and Adafruit boards).

These comparisons reveal a recurring pattern also noted in [68]: while open-source projects often stall at early mobility levels due to limited resources, coordination, and discoverability, well-resourced commercial platforms advance more effectively toward Levels 4 and 5, bridging the technical–pedagogical gap and delivering fully integrated mobile learning ecosystems. The presented framework can serve not only as a descriptive tool but also as a roadmap for the progression of open-source initiatives from basic accessibility to rich, participatory, and adaptive learning environments. In this sense, it also outlines the developmental trajectory envisioned for Odysseus, guiding its evolution from a technically accessible kit to a fully participatory mobile learning ecosystem.

5 MAIN CHARACTERISTICS OF ODYSSEUS

Although it seems proper to collect feedback from users before deciding on the shape of a new robot, such a strategy cannot be effective if the target robot is intended for use in all educational levels. In particular, research suggests that the optimal shape of educational robots varies with children's age. Younger children (under 9) prefer human-like appearances. For 9 to 10-year-olds, a gender-neutral, anthropomorphic, and (at the same time) machine-like robot is preferred [11], while older children and adults prioritize functionality and action skills [71].

Since Ody is intended for use in all educational levels, a compromise in shape is inevitable for some age groups. To minimize this compromise, Ody was designed to be as abstract as possible in shape while incorporating the functionality and action skills expected from older children and adults. The logic behind this choice is that, this way, Ody can be used by older children and adults and still be suitable for younger ages through carefully designed commands. After all, creating a humanoid robot based on Ody is always an option, given that Ody is compatible with bricks.

The choice to make Ody compatible with bricks is also based on published results. From kindergarten to graduate school, bricks have played a role in engineering education, increasing student interest in math and physics [45]. Educational robots utilizing programmable bricks stimulate learning motivation, foster creative thinking, and improve problem-solving abilities [48].

Ody is intended for home printing, aiming to motivate students to participate in the task. The reason for this choice is that 3D printing is expected to be a critical skill in the years to come [80]. Furthermore, such a choice is backed up by existing research. In particular, apart from allowing students to design, develop, and experiment with robotic components [63], 3D printing has been reported to enhance student imagination and to increase their interest in technology [43].

Another decision made is to allow Ody to be assembled without the use of a soldering iron. This choice is dictated by safety and health considerations. Soldering is complicated for most users, requires additional equipment, and involves very high temperatures that may lead to burns or other accidents. Thus, such a construction process may cause skepticism among potential users and is certainly not suitable for kids. Furthermore, evidence of intense nanoparticle generation from a low-power (45 W) flux soldering unit is reported in [25]. This is a familiar device often used in daily life, including home repairs and school electronic laboratories. However, eliminating the need for soldering requires providing an additional board where the soldering task is performed in advance. This is the case with Ody. This additional board is an Arduino Uno shield (a piece of hardware placed on top of Arduino, extending Arduino's circuitry). Through this shield, the construction task is simple, requiring only a screwdriver.

A major decision was the microcontroller used, and the choice was Arduino Uno, although there exist inexpensive microcontrollers that have become very popular lately (for example, micro:bit). The main reason for this choice was the strong community support that contributes to its effectiveness in educational settings [65]. This strong community support and popularity of Arduino have led to the introduction of other microcontroller boards identical to Arduino Uno in dimensions and pinout, such as Arduino Zero or Arduino Uno R4 WIFI. This allows for easy usage of such boards through minor additions to our shield, enabling us to provide more than one implementation that looks the same to the user.

6 THE PROGRAMMABLE COMPONENT AND SUPPORTED SENSORS

Figure 3 (top left) illustrates the programmable component. The upper side provides the outputs: motors and LEDs. Slots 1–4 drive servos, slots 5–6 drive DC motors, and slots 7–8 connect to LEDs. The opposite side provides the inputs (sensors): slots 12–15 accept 3-pin sensors, slot 16 connects an ultrasonic sensor, and slot 17 supports a color sensor or any device using the I²C bus. The middle section integrates an OLED display (9), a power switch (10), and a Bluetooth switch (11). Figure 3 (top right) lists the supported sensors: the obstacle sensor (18), a 3-pin sensor that detects obstacles and can also act as a line follower; the Sharp IR distance sensor (19), a 3-pin sensor that measures distance using infrared; the ultrasonic sensor (20), which measures distance via ultrasound; the color sensor (21, I²C); the soil humidity sensor (22, 3-pin); and the temperature and air humidity sensor (23, 3-pin). The lower part of Figure 3 shows Ody in the form of a wheeled robot. The robot on the left can follow a line using the two obstacle sensors placed facing the floor. The robot on the right can detect obstacles using the obstacle sensor facing forward. Note that the available slots allow for both following a line and detecting obstacles simultaneously, with one empty 3-pin slot remaining. The robot moves via two servo motors encased in plastic casings, plugged into slots 1 and 4.

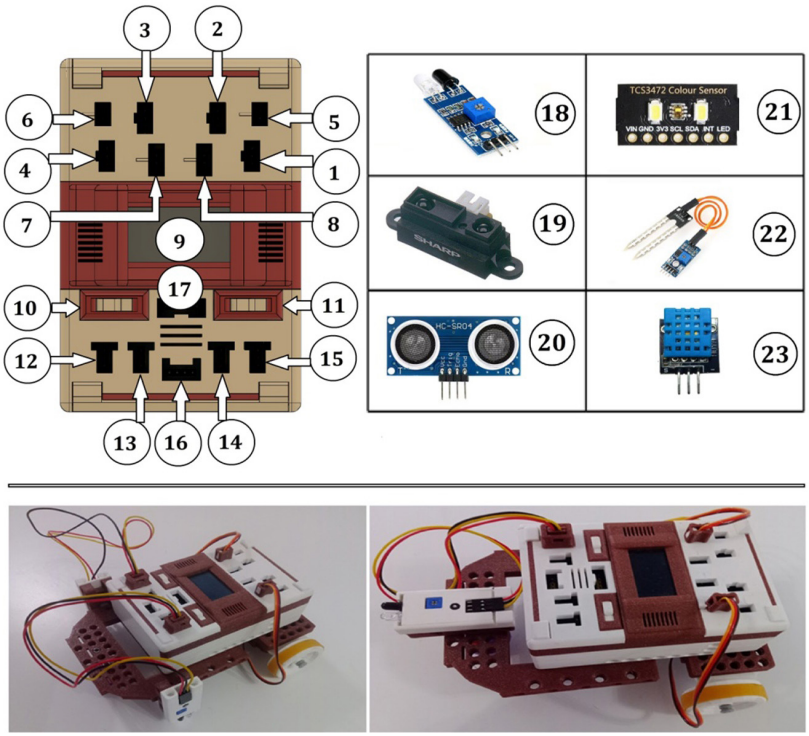

Fig. 3. The physical appearance of Ody

The plugs that fit into the slots are also 3D printable, and each type of plug fits into the corresponding slot type, with the correct orientation only. As a result, the user cannot create a short circuit by plugging in a sensor or a motor. The design also protects both the electronics and the user in case the batteries are oriented incorrectly, preventing any risk of damage.

7 SOFTWARE

Concerning the software, a modified version of standard Firmata [23] called Odysseus Firmata runs on the Arduino. It is a program that runs in a microcontroller (firmware) implementing the Firmata protocol [76], a generic protocol for communicating with microcontrollers from software on a host computer. Through Firmata, the microcontroller becomes an extension of the host computer, handled via a programming environment. In particular, whenever a command dealing with the inputs or outputs of the microcontroller runs in the user's programming environment, a message is sent to Ody (according to the Firmata protocol), and the firmware manipulates inputs and outputs. Standard Firmata supports easy implementation of additional commands defined by the user to match the technical details of any Arduino-based board. Odysseus Firmata is an extension of Standard Firmata to support the exact architecture of Ody.

8 PROGRAMMING PATHWAYS

We have defined three programming levels, each specifying distinct requirements and learning objectives. The three levels are as follows:

1. Level 1: Visual user interface (i.e., block commands identified through figures and not text), no loops or if-statements (loops and if-statements may exist inside block commands in such a way that the user does not ever deal with the notion of a loop or an if-statement). The block commands are "game ready", i.e., they are preset to work on a surface containing a grid or on a map that consists of black lines and colored spots.
2. Level 2: Visual user interface (i.e., block commands), if statements, loops, inputs (accessing sensors), outputs (manipulating motors), variables, lists, and more. It is a third-generation programming language through a visual user interface. The block commands accessing the sensors must return values that are easy to understand and use.
3. Level 3: A typical text-based third-generation programming language.

Visual (block-based) programming leads to a positive impact on academic achievement [28]. There exist a few visual programming languages [19], but the most commonly used ones are Scratch [66], Snap! [24], and some based on Blockly [60], a library used to create visual programming languages. All of these have extensions for connecting to microcontroller boards. Regarding Blockly-based languages, there are a few extensions for programming an Arduino, but either they are not mature enough or have long since stopped evolving. An exception is mBlock, which is mature and complete, but it is not open-source, which is a drawback. Apart from the open-source requirement, we need a programming language that provides users with the means to create new blocks with ease. We found these properties in Snap4Arduino (an extension of Snap! that can communicate with an Arduino), and we have chosen it as the programming environment for levels 1 and 2.

In addition to the advantages outlined above, Snap! also has educational validation. Studies using Snap! in introductory computer science contexts report encouraging results. For example, in an observational study with 40 senior high and vocational students [31], 70% of students with prior programming experience found AI programming using Snap! easy to understand. In another study [42], Snap! proved appealing to novice programmers, as observed by an independent evaluation. Robot control with Snap4Arduino is similar to other block-based languages such as Scratch. Therefore, any user who is familiar with Scratch can easily control Ody using Snap4Arduino.

Level 3 users can program Ody using Python after installing the pyfirmata2 library. Additional code has also been written to simplify this communication.

9 INSTRUCTIONAL FLOW

After assembling the kit, users need to connect it to their computer either via wire or Bluetooth. In the latter case, they have to "pair" Ody's Bluetooth module to their computer and see the "COM port" assigned to this module by the operating system. Then users execute Snap4Arduino, connect to Ody by choosing the correct COM port, and import the Ody-specific commands (it is a simple XML file that can be locally stored). Programming Ody follows, and for this purpose, a programming course using Ody has been developed on YouTube [64], providing a roadmap for teaching fundamental concepts such as programming, algorithms, and data structures. The course is primarily aimed at secondary education students, beginning with simple tasks and progressing through the basics of programming (if-statements, loops, and variables). Then, the course proceeds to basic algorithms (counting items and summing numbers), more advanced programming concepts (real-time programming with timers), and data structures (memory management with lists), including debugging techniques through the use of data structures.

10 PROGRAMMING WITH ODY

10.1 Level 2 programming: The color collector scenario

The concept of data structures is crucial in programming, and teaching this notion to children is a challenge. The list structure is a logical choice for introducing this subject, and Snap4Arduino contains all the necessary block commands for manipulating lists. However, to help children follow the programming lesson, one must materialize the elements of the list. A suitable candidate for these materialized list elements is colors. The goal is to create a list of all the detected colors. We have named this scenario the color collector.

The problem is that the color sensor does not return colors but numbers. In particular, one of the most commonly used and affordable color sensors is the TCS3472, which provides digital output of red, green, blue (RGB), and clear light sensing values. According to the guidelines for Level 2 programming (presented in the previous subsection), users should identify the elements in the list not as numbers but as colors. Furthermore, the sensor is so sensitive that it is unlikely to return the same values each time it detects the same color (due to slight variations). As a result, users cannot easily exploit it. It follows that:

- We want the sensor to return a color (i.e., blue or yellow) and not numbers.
- We want the sensor to be sensitive enough to accurately distinguish colors without being oversensitive, i.e., identifying variations of a given color as different colors.

To solve this issue, we have created specialized block commands to support the color sensor. A set of colors is defined, which we call basic colors (green, red, blue, cyan, magenta, yellow, black, white, and floor). The user can easily activate or deactivate basic colors and then match the colors of real objects to the activated basic colors. For example, if the user has a piece of paper identified as blue, the user can store the values returned by the sensor when detecting this color as the basic blue color. We then say that the basic blue color is "fixed". The commands hide the actual values of the basic colors from users, which is intentional: we want them to see colors rather than numbers.

After activating and fixing all the colors involved in an educational scenario, any detected color can be matched by Ody to the closest basic color, as long as the closest basic color is close enough. The user can define what "close enough" means by setting a parameter, which adjusts the sensor sensitivity to ensure that slight variations that tend to occur when detecting a given color are not identified as a different or unknown color. Figure 4 visualizes the color collector scenario. We assume that the user possesses colored pieces of paper and has placed them on the floor. In this instance, we use three colors. Ody will move as shown by the arrow, with its color sensor facing down. Before the move, the user activates the involved colors and deactivates the rest of the basic colors. The next step is to fix the three activated colors. Users fix a color by placing Ody in such a way that the color sensor is on top of a colored piece of paper and store the detected color by just clicking on a command block. After fixing all involved colors, Ody is ready to start moving. Initially, the list of collected colors is empty. Each time Ody detects a new color that has been activated and is not in the list, it inserts it into the list. Ody stops when all the activated colors are in the list. In this example, the activated colors are yellow, red, and blue, and Ody will stop when it detects the blue color. Figure 5 shows the pseudocode, the flowchart, and the code.

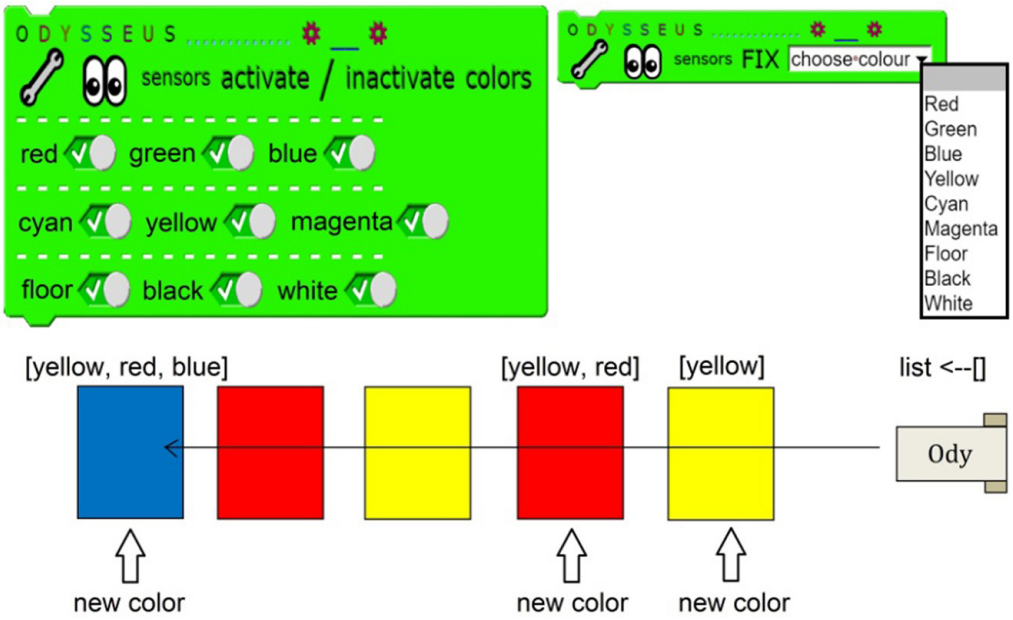

Fig. 4. The color collector scenario: the specialized commands for activating (or inactivating) and fixing colors are shown

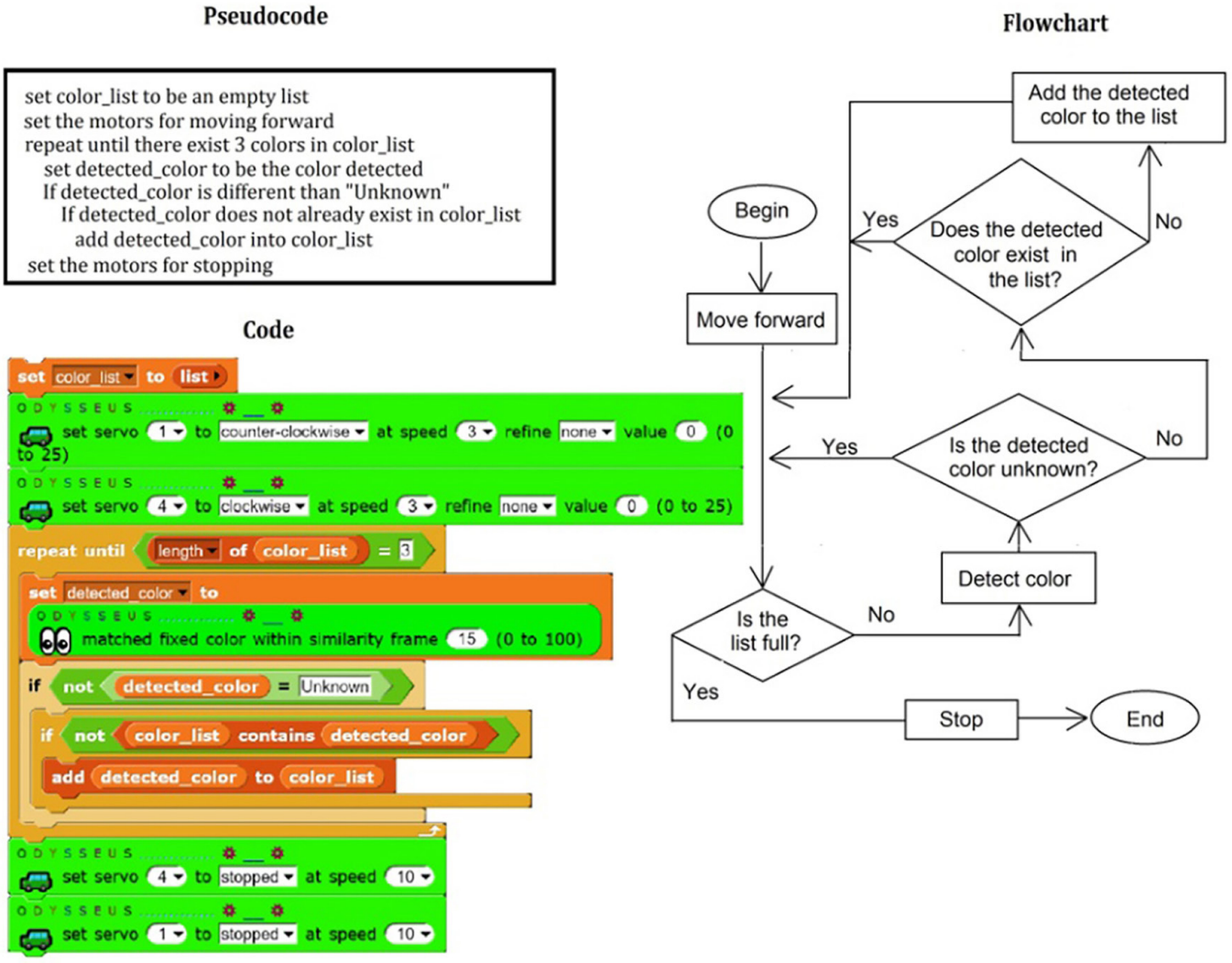

Fig. 5. The code for the color collector scenario

The values of the basic colors, as well as the activated/deactivated colors, are stored in permanent memory inside the Arduino Uno, so the user does not have to initialize and fix the basic colors every time. However, different light conditions could significantly alter the values returned by the sensor. In such cases, the user needs to fix the involved colors again.

Note that the code does not include notions and details that do not exist in the pseudocode or in the flowchart, which means that the transition from pseudocode or flowchart to code is straightforward.

10.2 Level 3 programming: The color collector in Python

For users capable of programming in text-based languages, we have adopted Python: writing the Python code for the color collector scenario is straightforward due to the following abstractions available to the users:

- Ody.move_servo(servo_slot, direction, speed): Sets the speed (0 to 10) of a servo motor connected to servo_slot (1 to 4) and moves it in the specified direction (clockwise, counterclockwise, or stopped).

- Color = ody.matched_fixed_color(20): Returns the basic fixed color that best matches the color detected by the sensor. The parameter 20 defines the sensitivity of the algorithm that matches the colors. A value of 0 means that the command will identify a color if the detected sensor values exactly match those of a predefined basic color. Increasing this parameter decreases the sensitivity of the matching process, though the sensor's actual operation remains unchanged. If the value is set too high, multiple basic colors might be considered close enough to the detected color, resulting in the return of "unknown". Users must determine an appropriate value for this parameter through trial and error.

  Using the above abstractions, the complete code is trivial, and thus we omit it.

10.3 Level 1 programming: Playing games

Although playing and having fun are important aspects of every educational scenario in robotics for all ages, they are even more crucial when dealing with very young kids. One can devise many educational games where the robot plays a role. However, we have created specialized block commands for two general categories of games:

1. Grid games. Such games typically do not require sensing the environment. The robot moves on a surface that can be seen as a grid composed of square elements (e.g., floor tiles). In this setting, the user places the robot within a square element and sets it to move to another.
2. Map games. The robot moves on a light-colored surface that contains dark lines. The robot can follow a line and choose one of two alternatives when it reaches a (two-way) crossroad.

It is important to note that complexity does not always stem from algorithmic difficulty; sometimes, it arises from numerous intricate technical details. While these details might be beyond the grasp of Level 1 users, the algorithmic aspects can often be within their reach. To make educational scenarios more accessible, we abstract away technical details through configuration blocks designed for educators to execute. An educator who is familiar with Level 2 programming can easily perform these configuration tasks. Each configuration block stores specific values in the permanent memory of the Arduino, simplifying the process for young learners.

Level 1 users will initially "drive" the robot rather than program it, and we do not want to expose them to the complexity of handling servo motors. Instead, the educator predefines the servo slots used as wheels and the speed settings for moving straight and turning using the block shown at the top part of Figure 6. The children can then use the Level 1 block commands shown at the bottom part of Figure 6 to move forward, backward, turn left or right, and stop. The Level 1 commands of Figure 6 serve as the foundation for additional commands used on a grid, where Ody moves by performing steps and 90-degree turns. A "step" is defined as the transition from one square element of the grid to the next, either horizontally or vertically. To utilize the grid commands, an additional configuration task is required. Specifically, the educator must define the time it takes for Ody to move one step forward or backward and to turn 90 degrees to the left or right.

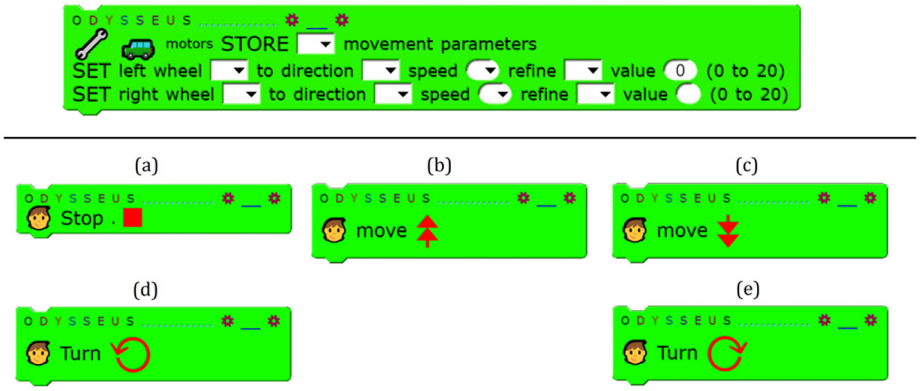

Fig. 6. Configuration for basic movements

This configuration process is depicted in the top left part of Figure 7. Block (a) is used to start the configuration task for steps forward or backward. Block (b) is used to start the configuration task for performing 90-degree turns to the left or to the right. Block (c) ends the configuration tasks initiated through (a) or (b). After the configuration, blocks (d) for stepping backward, (e) for stepping forward, (f) for turning 90 degrees to the right, and (g) for turning 90 degrees to the left can be used. The configuration is performed as follows: the educator places Ody at the center of a square element and sets the type of move (forward or backward) along with the number of square elements Ody will traverse. After executing the configuration block, Ody starts moving, and the educator stops the move once Ody reaches its destination. When the move is stopped, the time required to perform a step is calculated and stored in the Arduino's permanent memory. Similarly, for turning, the educator sets the number of 360-degree turns (left or right) Ody will perform and initiates the move by executing the block. Once Ody completes its turn, the educator stops the move, and the turning time for a 90-degree turn is calculated and stored in Arduino's permanent memory. The lower part of Figure 7 illustrates a grid. The program shown on the right side of the grid will move Ody according to the red arrows.

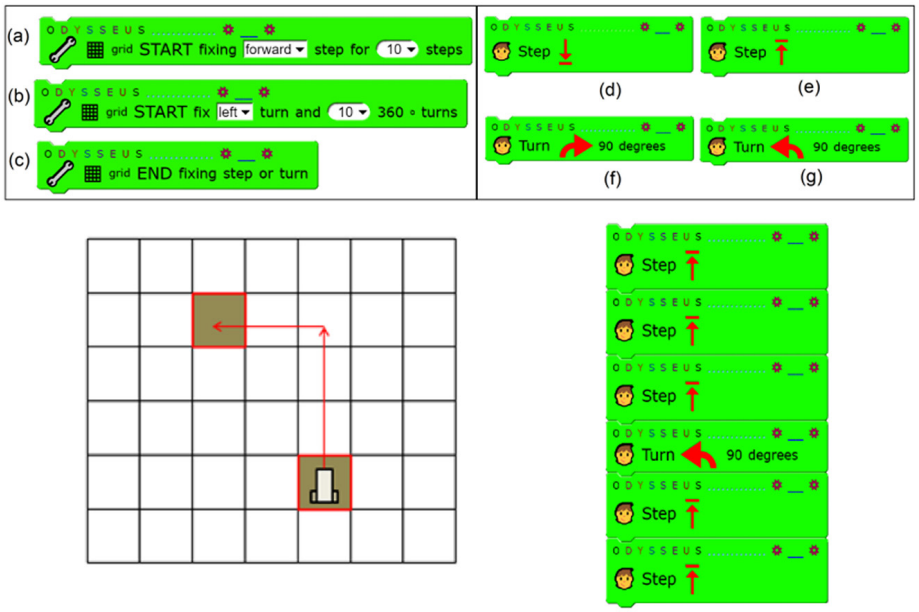

Fig. 7. Configuration for moving on a grid

More games are introduced through maps consisting of lines that Ody must follow via two obstacle sensors. To configure Ody for map games, the educator needs to define the slots where these sensors are plugged in. Once this is done, Level 1 blocks for moving on a map become available to users. Using these blocks, Ody can follow a path to a specific endpoint, even navigating crossroads along the way. Figure 8 shows the configuration block (a), the block for following a line until a crossroad or an obstacle is detected (b), the block for turning left to start following the line in the opposite direction (c), for choosing the left line (d) or the right line (e) when a crossroad is detected, and for turning right to start following the line in the opposite direction (f). Figure 8 also shows a simple educational scenario on a map. In this scenario, the code on the right will make Ody move along the red line to the obstacle. For Ody to detect obstacles in the educational scenario shown in Figure 8, the educator must configure the ultrasonic sensor by defining the detection distance. Any object closer than this distance is then recognized as an obstacle by the Level 1 blocks in Figure 8. Additionally, users can move an actuator by executing a Level 1 block. For this block to function, the educator must configure the actuator by specifying the motor slot, the motor type (continuous servo motor or angle servo motor), and a few movement parameters. After this is done, a Level 1 block can be used that repeats the exact movement created and stored by the educator. The ability to use an actuator is introduced to enhance the engagement and excitement of educational scenarios, helping to capture and sustain users' attention and enthusiasm.

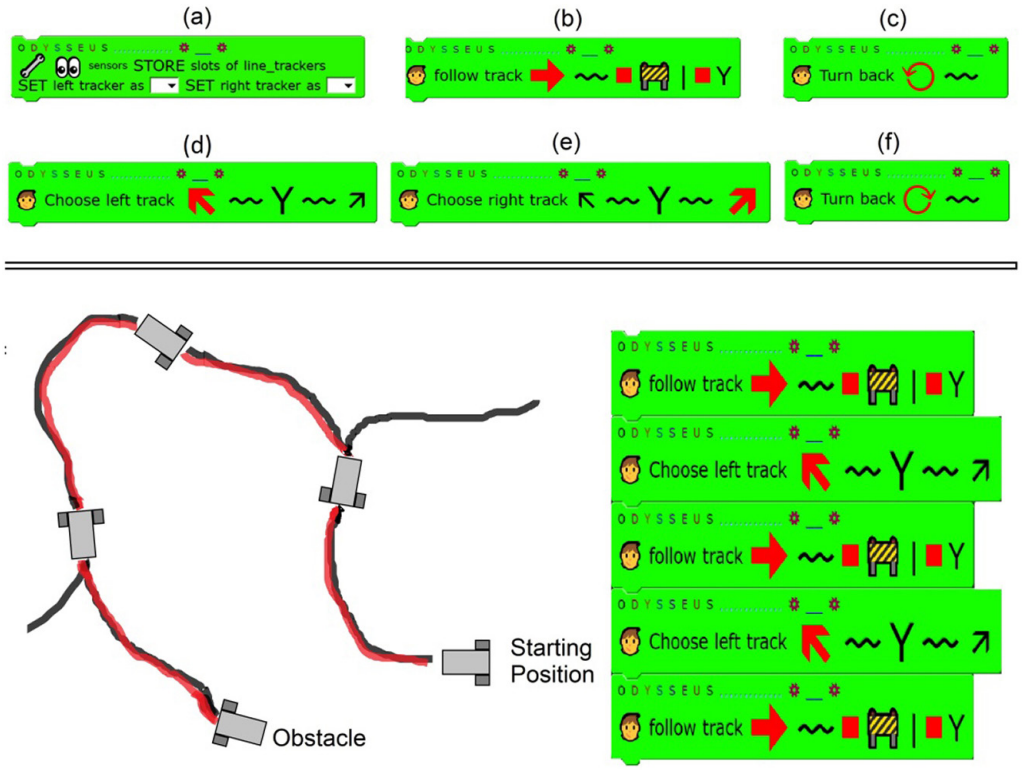

Fig. 8. Configuration for following lines

11 EVALUATION

As outlined earlier, Odysseus currently sits between Mobility Levels 1 and 2, while it has some Level 4 characteristics. The evaluation presented here, therefore,

serves a dual purpose: first, to assess the immediate qualities of Odysseus as an educational tool (visual appeal, reliability, and usability), and second, to provide evidence on whether it possesses the user acceptance and engagement qualities required to progress toward higher mobility levels.

Before beginning the evaluation, it was essential to clarify what aspects of Ody required assessment, given that its foundational components have already been validated. Specifically, Arduino is widely recognized as an effective educational tool, and Snap! has also undergone prior evaluation. Python is also generally accepted as a viable educational programming language. Therefore, the primary objectives of this evaluation were to determine:

- Whether Ody is visually appealing, which is a crucial factor assumed to influence the decision to 3D-print and use the robot.
- Whether Ody exhibits reliability during use, particularly in terms of avoiding errors or unpredictable behavior.

We conducted two rounds of evaluation with participants who already had some familiarity with educational robotics. We obtained informed consent from all adult participants. For child participants, we obtained informed consent from their parents or legal guardians. All participants were informed about the purpose of the study, their right to withdraw at any time, and how their data would be used and protected.

The first evaluation round focused on capturing initial impressions, particularly concerning the aesthetics and robustness of the robot. Before interaction, we asked participants to assemble the programmable kit and provide feedback on the assembly experience.

We formed two separate evaluation groups. Group A consisted of 10 primary school children (ages 9–12), while Group B comprised 10 secondary school students (age 17). Each group received the required electronic components and 3D-printed parts to assemble the robot. In Group A, an educator was present to guide as needed, whereas Group B completed the task independently. After assembly, each group had the opportunity to interact with and program the robot. Group A continued under supervision, while Group B remained unsupervised. Following the activity, we asked all participants to complete a questionnaire, which was the same for both groups and employed a 5-point Likert scale. The questions aimed to reveal the extent of their prior experience with educational robots (Q1), simplicity of the assembling process (Q2), fun during the assembling process (Q3), first impression from an aesthetic point of view (Q4), simplicity of programming (Q5), and finally, overall fun (Q6). The results are positive, as Table 1 demonstrates, with the result on Ody's aesthetics being the most important one, given the limited duration of this evaluation round.

Table 1. Results of first evaluation round

Group A (Primary Education)							Group B (Secondary Education)						
	1	2	3	4	5	Avg		1	2	3	4	5	Avg
Q1			2	4	4	4.2	Q1			2	3	5	4.3
Q2		1	4	3	2	3.6	Q2			1	2	7	4.6
Q3		1	2	4	3	3.9	Q3				2	8	4.8
Q4			1	5	4	4.3	Q4				3	7	4.7
Q5		2	3	4	1	3.4	Q5			1	3	6	4.5
Q6			1	4	5	4.4	Q6				2	8	4.8

These results suggest that Odysseus's physical design is appealing to learners across age groups, which is an important driver of voluntary adoption and aligns with the platform's goal of encouraging users to invest in 3D printing and assembly. This result indicates that by providing clear support materials and instructions, users are less likely to be discouraged; on the contrary, the assembly process itself can strengthen their intention to 3D-print the robot.

In the second round of evaluation, three groups of 20 participants were formed, each representing a different educational level: primary, secondary, and higher education. All participants had used educational robots, either at school or at home. Each group attended a 12-hour programming course conducted outside regular school hours. The course for the primary education group covered an introduction to motors and sensors (3 hours), variables (6 hours), and concluded with a final project (3 hours). It is worth mentioning that all members of this group were 11 or 12 years old. The secondary education group's course was based on Snap4Arduino and included an introduction to motors, sensors, and variables (2 hours), followed by programming techniques, timers, and lists (7 hours), and ended with a final project (3 hours). The higher education group was based on Python and followed the same structure and content as the secondary education group.

All students from all groups completed the project successfully, and then they were asked to fill out a questionnaire created according to the Technology Acceptance Model (TAM). TAM is a widely used framework for understanding how users come to accept and use new technologies. Originally developed by Davis [21], TAM suggests that two key factors (perceived usefulness and perceived ease of use) determine an individual's intention to use a technology, which in turn influences actual usage behavior. In the context of educational technology, TAM provides valuable insights into how students, educators, or administrators evaluate new tools like e-learning platforms or educational robots. Multiple studies have applied TAM to assess various robotic platforms, including a novel modular robot [16] and Lego Mindstorms EV3 [44]. The research indicates that TAM is an effective method for evaluating robotics kits, providing insights into user attitudes and intentions across different platforms and user groups, including teachers and students.

The first four questions assess Perceived Usefulness, aiming to capture how the robot and software contributed to students' understanding of programming concepts and engagement in the learning process. Questions 5 to 8 focus on Perceived Ease of Use, exploring the usability of both the robot and its programming environment. Questions 9 and 10 examine students' Attitude Toward Use (ATU), reflecting their overall feelings about the learning experience. Finally, questions 11 and 12 address Behavioral Intention to Use, identifying students' willingness to use the technology again in future educational contexts. Table 2 shows the results.

Table 2. Questionnaire items and average scores by educational level

ID	Question	Primary	Secondary	Higher
Q1	Using Ody has improved my understanding of fundamental programming concepts.	4.25	4.55	4.45
Q2	Using Ody has made the learning experience more engaging and effective.	4.55	4.85	4.85
Q3	Ody has enhanced my ability to connect programming concepts to real-world physical components (e.g., sensors, actuators).	4.30	4.60	4.30
Q4	The specialized commands in Ody have enabled me to apply programming knowledge more effectively in practice.	4.40	4.55	4.40
Q5	Learning how to use Ody was easy for me.	4.75	4.55	4.40

(Continued)

Table 2. Questionnaire items and average scores by educational level *(Continued)*

ID	Question	Primary	Secondary	Higher
Q6	Establishing a connection between the computer and Ody was straightforward and intuitive.	4.65	4.60	4.25
Q7	The programming environment used with Ody was user-friendly and easy to navigate.	4.75	4.75	4.85
Q8	I experienced minimal technical difficulties while using Ody and its software.	4.20	4.20	4.30
Q9	My overall experience using Ody was positive and enjoyable.	4.70	4.80	4.40
Q10	I would recommend incorporating Ody into classroom teaching.	4.80	4.95	4.40
Q11	I intend to use Ody in future projects or educational activities.	4.60	4.75	4.25
Q12	This experience has increased my willingness to engage with technology in the future.	4.60	4.70	4.65

Table 3 illustrates the average scores per TAM construct across different educational levels. The results indicate high levels of agreement across all constructs, suggesting a positive perception of Odysseus. Notably, ATU received the highest average scores across all groups, particularly in secondary education (M = 4.88), highlighting strong emotional and motivational engagement. These findings suggest that Ody was perceived as both effective and accessible, contributing to high behavioral intentions to reuse the tool in future educational contexts.

Table 3. Average scores per TAM construct across different educational levels

Construct	Primary	Secondary	Higher
Perceived Usefulness (PU)	4.38	4.64	4.50
Perceived Ease of Use (PEOU)	4.59	4.53	4.38
Attitude Toward Using (ATU)	4.75	4.88	4.70
Behavioral Intention to Use (BIU)	4.60	4.73	4.60

The consistently high ratings for PU and ATU across all education levels indicate that Ody already has strong foundations in learner motivation and engagement, both critical for achieving the sustained participation and community-building envisioned in Mobility Levels 4 and 5.

While the evaluation yielded consistently positive findings, certain methodological constraints should be noted.

First, it is important to acknowledge that all evaluators had prior experience with educational robotics, and since they were all volunteers, it is reasonable to assume that they enjoyed programming and interacting with educational robots before this evaluation. As a result, the favorable outcomes observed in this evaluation may not necessarily be replicated in a typical classroom setting (particularly during school hours), where students may be less motivated or less interested in programming. This context was intentional, as it allowed for focused feedback from motivated users capable of identifying subtle design and usability issues at an early stage of development. Furthermore, with a more diverse sample, any negative feedback might reflect general attitudes toward programming or educational robotics rather than an accurate assessment of Ody itself. For this reason, we believe that the chosen evaluation framework is appropriate for assessing the specific qualities of Ody as a tool. However, we do acknowledge the need for additional future evaluations over a more diverse sample.

Second, the study involved a relatively small number of participants, thereby limiting the extent to which the findings can be generalized to broader populations. However, in the context of early-stage evaluation, such a sample size is appropriate for identifying key strengths and potential areas for refinement before undertaking large-scale deployment. The consistency of positive responses across different educational levels suggests that the observed trends are meaningful, even if not yet conclusive. Accordingly, these results should be interpreted as an initial, targeted evaluation of Odysseus's potential rather than a definitive measure of its impact in the general classroom population. Broader trials will be essential for validating and extending these findings, ensuring that they hold across a wider range of contexts and learner profiles.

12 CONCLUSIONS

In this paper, we have presented Ody, a new educational robotic kit that combines several distinctive features. Ody is 3D-printable and open-source, can be assembled with just a screwdriver, relies on inexpensive electronic parts widely available on the market, and has an abstract shape compatible with bricks, enabling the robot-forming process to be integrated into educational scenarios when desired. In addition, it is supported by open-source, educationally appropriate programming languages suitable for all levels of education.

Beyond its practical design, Ody has also contributed to theoretical perspectives. The Functional–Pedagogical Mobility Framework was developed as a roadmap for tracing Ody's evolution from technical accessibility to participatory learning ecosystems. Similarly, the Dual Threshold Scaffolding Model (drawing on both task difficulty and abstraction) was employed to describe how learners progress from simple construction tasks to higher-order problem-solving. In this sense, Ody serves not only as a versatile educational tool but also as a case study that contributes to refining our understanding of how mobile, open-source technologies can be aligned with pedagogy.

The evaluation of Ody is ongoing. Future research should explore and enhance its functionality and ease of use within well-structured educational scenarios across different levels of education. A valuable direction is to compare Ody with other educational robots in classroom contexts to better understand its relative strengths and limitations. The central research trajectory, however, lies in advancing Ody within the Functional–Pedagogical Mobility Framework introduced in this paper. Through this application, the framework demonstrates its practical utility as a developmental roadmap, aligning technical innovations with pedagogical objectives and supporting Ody's evolution into a fully participatory mobile learning ecosystem.

13 REFERENCES

[1] A. O. Ajlouni and S. A. Jaradat, "The effect of integrating an educational robot with hypermedia on students' acquisition of scientific concepts: The case of fifth-grade students," *Int. J. Interact. Mob. Technol.*, vol. 15, no. 11, pp. 113–132, 2021. https://doi.org/10.3991/ijim.v15i11.18537

[2] D. Alimisis, "Educational robotics: Open questions and new challenges," *Themes in Science and Technology Education*, vol. 6, no. 1, pp. 63–71, 2013.

[3] N. Arís and L. Orcos, "Educational robotics in the stage of secondary education: Empirical study on motivation and STEM skills," *Education Sciences*, vol. 9, no. 2, p. 73, 2019. https://doi.org/10.3390/educsci9020073

[4] L. Athanasiou, T. A. Mikropoulos, and D. Mavridis, "Robotics interventions for improving educational outcomes—A meta-analysis," in *Technology and Innovation in Learning, Teaching and Education*, M. Tsitouridou, A. Diniz, and T. Mikropoulos, Eds., Cham, Switzerland: Springer, 2019. https://doi.org/10.1007/978-3-030-20954-4_7

[5] S. F. Barrett, *Arduino Microcontroller Processing for Everyone!* Cham, Switzerland: Springer Nature, 2022.

[6] S. Bai and P. Tian, "Educational robotics may enhance students' conceptual knowledge, applied skills, and learning attitude in STEM education: A meta-analysis," *Educational Technology & Society*, vol. 28, pp. 271–300, 2025. https://doi.org/10.30191/ETS.202510_28(4).SP06

[7] F. B. V. Benitti, "Exploring the educational potential of robotics in schools: A systematic review," *Computers & Education*, vol. 58, no. 3, pp. 978–988, 2012. https://doi.org/10.1016/j.compedu.2011.10.006

[8] M. U. Bers, *Designing Digital Experiences for Positive Youth Development: From Playpen to Playground*. New York, NY, USA: Oxford Univ. Press, 2012. https://doi.org/10.1093/acprof:oso/9780199757022.001.0001

[9] M. Berland and U. Wilensky, "Comparing virtual and physical robotics environments for supporting complex systems and computational thinking," *Journal of Science Education and Technology*, vol. 24, no. 5, pp. 628–647, 2015. https://doi.org/10.1007/s10956-015-9552-x

[10] A. Bicer, S. B. Nite, R. M. Capraro, L. R. Barroso, M. M. Capraro, and Y. Lee, "Moving from STEM to STEAM: The effects of informal STEM learning on students' creativity and problem solving skills with 3D printing," in *Proc. IEEE Frontiers in Education Conf. (FIE)*, Indianapolis, IN, USA, 2017, pp. 1–6. https://doi.org/10.1109/FIE.2017.8190545

[11] M. Blancas *et al.*, "Analyzing children's expectations from robotic companions in educational settings," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, 2017. https://doi.org/10.1109/HUMANOIDS.2017.8246956. [Accessed: July 8, 2025].

[12] J. S. Bruner, *The Process of Education*. Cambridge, MA, USA: Harvard Univ. Press, 1960. https://doi.org/10.4159/9780674028999

[13] K. Castelli and H. Giberti, "Additive manufacturing as an essential element in the teaching of robotics," *Robotics*, vol. 8, no. 3, p. 73, 2019. https://doi.org/10.3390/robotics8030073

[14] S. Çelik and S. Özdemir, "Tinkering learning in classroom: An instructional rubric for evaluating 3D printed prototype performance," *Int. J. Technol. Des. Educ.*, vol. 30, pp. 459–478, 2020. https://doi.org/10.1007/s10798-019-09512-w

[15] S. A. Chatzichristofis, "Recent advances in educational robotics," *Electronics*, vol. 12, no. 4, p. 925, 2023. https://doi.org/10.3390/electronics12040925

[16] A. Chatzopoulos, M. Kalogiannakis, S. Papadakis, and M. Papoutsidakis, "A novel, modular robot for educational robotics developed using action research evaluated on Technology Acceptance Model," *Education Sciences*, vol. 12, no. 4, p. 274, 2022. [Online]. Available: https://doi.org/10.3390/educsci12040274

[17] C. Chronis and I. Varlamis, "FOSSBot: An open-source and open design educational robot," *Electronics*, vol. 11, no. 16, p. 2606, 2022. https://doi.org/10.3390/electronics11162606

[18] C. C. Chung, C. Cartwright, and M. Cole, "Assessing the impact of an autonomous robotics competition for STEM education," *J. STEM Educ.: Innov. Res.*, vol. 15, no. 2, pp. 24–34, 2014. [Online]. Available: https://www.jstem.org/jstem/index.php/JSTEM/article/view/1704

[19] E. Coronado, F. Mastrogiovanni, B. Indurkhya, and G. Venture, "Visual programming environments for end-user development of intelligent and social robots: A systematic review," *J. Comput. Lang.*, vol. 58, art. no. 100970, 2020. https://doi.org/10.1016/j.cola.2020.100970

[20] A. D'Amico, D. Guastella, and A. Chella, "A playful experiential learning system with educational robotics," *Frontiers in Robotics and AI*, vol. 7, no. 33, 2020. https://doi.org/10.3389/frobt.2020.00033

[21] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Q.*, vol. 13, no. 3, pp. 319–340, 1989. https://doi.org/10.2307/249008

[22] P. Denny, S. MacNeil, J. Savelka, L. Porter, and A. Luxton-Reilly, "Desirable characteristics for AI teaching assistants in programming education," in *Proc. 2024 on Innovation and Technology in Computer Science Education V. 1*, 2024, pp. 408–414. https://doi.org/10.1145/3649217.3653574

[23] Firmata, "Firmata protocol (Version 2.5.9)," GitHub, 2022. [Online]. Available: https://github.com/firmata/arduino. [Accessed: July 8, 2025].

[24] D. Garcia, L. Segars, and J. Paley, "Snap! (Build your own blocks) tutorial presentation," *J. Comput. Sci. Coll.*, vol. 27, no. 4, pp. 120–121, 2012.

[25] V. Gómez, S. Irusta, F. Balas, and J. Santamaria, "Intense generation of respirable metal nanoparticles from a low-power soldering unit," *J. Hazard. Mater.*, vol. 256–257, pp. 84–89, 2013. https://doi.org/10.1016/j.jhazmat.2013.03.067

[26] E. Halverson and K. Sheridan, "The maker movement in education," *Harvard Educational Review*, vol. 84, no. 4, pp. 495–504, 2014. https://doi.org/10.17763/haer.84.4.34j1g68140382063

[27] G. Halfacree, *The Official BBC Micro: Bit User Guide*. Hoboken, NJ, USA: John Wiley & Sons, 2017. https://doi.org/10.1002/9781119413752

[28] Y. Hu, C. H. Chen, and C. Y. Su, "Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis," *J. Educ. Comput. Res.*, vol. 58, no. 8, pp. 1467–1493, 2021. https://doi.org/10.1177/0735633120945935

[29] C. Iancu, D. Iancu, and A. Stăncioiu, "From CAD model to 3D print via 'STL' file format," *Fiability & Durability*, no. 1, pp. 73–80, 2010.

[30] A. Ioannou and E. Makridou, "Exploring the potentials of educational robotics in the development of computational thinking: A summary of current research and practical proposal for future work," *Educ. Inf. Technol.*, vol. 23, pp. 2531–2544, 2018. https://doi.org/10.1007/s10639-018-9729-z

[31] K. Kahn, R. Megasari, E. Piantari, and E. Junaeti, "AI programming by children using Snap! block programming in a developing country," in *Proc. European Conf. Technology Enhanced Learning*, 2018.

[32] F. Kaloti Hallak, M. Armoni, and M. Ben-Ari, "Students' attitudes and motivation during robotics activities," in *Proc. Workshop in Primary and Secondary Computing Education*, 2015. https://doi.org/10.1145/2818314.2818317

[33] M. Kalaitzidou and T. P. Pachidis, "Recent robots in STEAM education," *Education Sciences*, vol. 13, no. 3, p. 272, 2023. https://doi.org/10.3390/educsci13030272

[34] M. Kandlhofer and G. Steinbauer, "Evaluating the impact of educational robotics on pupils' technical- and social-skills and science-related attitudes," *Robot. Auton. Syst.*, vol. 75B, pp. 679–685, 2015. https://doi.org/10.1016/j.robot.2015.09.007

[35] C. Kefalis, C. Skordoulis, and A. Drigas, "The role of 3D printing in science, technology, engineering, and mathematics (S.T.E.M.) education in general and special schools," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 20, no. 12, pp. 4–18, 2024. https://doi.org/10.3991/ijoe.v20i12.48931

[36] G. Keren and M. Fridin, "Kindergarten social assistive robot (KindSAR) for children's geometric thinking and metacognitive development in preschool education: A pilot study," *Comput. Hum. Behav.*, vol. 35, pp. 400–412, 2014. https://doi.org/10.1016/j.chb.2014.03.009

[37] A. Keune and K. A. Peppler, "Materials to develop with: The making of a makerspace," *Br. J. Educ. Technol.*, vol. 50, pp. 280–293, 2019. https://doi.org/10.1111/bjet.12702

[38] M. Kearney, S. Schuck, K. Burden, and P. Aubusson, "Viewing mobile learning from a pedagogical perspective," *Research in Learning Technology*, vol. 20, 2012. https://doi.org/10.3402/rlt.v20i0/14406

[39] G. Kyriazopoulos, A. Koutromanos, A. Voudouri, and A. Galani, "Educational robotics in primary education: A systematic literature review," in *Handbook of Research on Using Educational Robotics to Facilitate Student Learning*, S. Papadakis and M. Kalogiannakis, Eds., Hershey, PA, USA: IGI Global, 2021, pp. 377–401. https://doi.org/10.4018/978-1-7998-6717-3.ch015

[40] G. Lagogiannis, "Odysseus – A new educational robotic kit," GitHub, 2025. [Online]. Available: https://github.com/Giorgiolag/Odysseus-A-new-educational-robotic-kit. [Accessed: Aug. 25, 2025].

[41] R. Luckin, *Re-Designing Learning Contexts: Technology-Rich, Learner-Centred Ecologies*, 1st ed. Abingdon, UK: Routledge, 2010. https://doi.org/10.4324/9780203854754

[42] H. Leleux *et al.*, "Work in progress: Programming is a SNAP! Increasing knowledge and interest in computer science," in *Proc. Conf. Inf. Technol. Educ.*, 2015. https://doi.org/10.1145/2808006.2808043

[43] K. Y. Lin, S. C. Lu, H. H. Hsiao, C. P. Kao, and P. J. Williams, "Developing student imagination and career interest through a STEM project using 3D printing with repetitive modeling," *Interact. Learn. Environ.*, vol. 31, no. 5, pp. 2884–2898, 2021. https://doi.org/10.1080/10494820.2021.1913607

[44] M. Masril, Ambiyar, N. Jalinus, Ridwan, and B. Hendrik, "Robotic education in 21st century: Teacher acceptance of LEGO Mindstorms as powerful educational tools," *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 12, no. 2, pp. 119–126, 2021. https://doi.org/10.14569/IJACSA.2021.0120216

[45] S. W. McNamara, M. N. Cyr, C. Rogers, and B. Bratzel, "LEGO brick sculptures and robotics in education," in *1999 Annu. Conf. Proc.*, 1999.

[46] Meet Edison, "Edison Programmable Robot – Ideal for school classroom education," Meet Edison, 2025. [Online]. Available: https://meetedison.com/. [Accessed: Aug. 25, 2025].

[47] F. Mondada *et al.*, "Bringing robotics to formal education: The Thymio open-source hardware robot," *IEEE Robot. Autom. Mag.*, vol. 24, no. 1, pp. 77–85, 2017. https://doi.org/10.1109/MRA.2016.2636372

[48] N. Morze, E. Smyrnova Trybulska, W. Zuziak, P. Kommers, and M. Boiko, "Robotics in primary school in the opinion of prospective and in-service teachers: A comparison study," *Int. J. Continuing Eng. Educ. Lifelong Learn.*, vol. 27, no. 1, 2017. https://doi.org/10.1504/IJCEELL.2017.10003680

[49] T. Mott, A. Bejarano, and T. Williams, "Robot co-design can help us engage child stakeholders in ethical reflection," in *Proc. ACM/IEEE Int. Conf. Human–Robot Interact. (HRI)*, Sapporo, Japan, 2022, pp. 14–23. https://doi.org/10.1109/HRI53351.2022.9889430

[50] M. Müller, C. Schindler, and W. Slany, "Engaging students in open-source: Establishing FOSS development at a university," in *Proc. 52nd Hawaii Int. Conf. Syst. Sci.*, 2019, pp. 7721–7730. https://doi.org/10.24251/HICSS.2019.930

[51] E. Mousavinasab, N. Zarifsanaiey, S. R. Niakan Kalhori, M. Rakhshan, L. Keikha, and M. Ghazi Saeedi, "Intelligent tutoring systems: A systematic review of characteristics, applications, and evaluation methods," *Interact. Learn. Environ.*, vol. 29, no. 1, pp. 142–163, 2018. https://doi.org/10.1080/10494820.2018.1558257

[52] E. Novak, M. Brannon, M. R. Librea Carden, and A. L. Haas, "A systematic review of empirical research on learning with 3D printing technology," *J. Comput. Assist. Learn.*, vol. 37, no. 5, pp. 1455–1478, 2021. https://doi.org/10.1111/jcal.12585

[53] G. Nugent, B. Barker, N. Grandgenett, and V. I. Adamchuk, "Impact of robotics and geospatial technology interventions on youth STEM learning and attitudes," *J. Res. Technol. Educ.*, vol. 42, no. 4, pp. 391–408, 2010. https://doi.org/10.1080/15391523.2010.10782557

[54] A. M. Ortiz, B. Bos, and S. Smith, "The power of educational robotics as an integrated STEM learning experience in teacher preparation programs," *J. Coll. Sci. Teach.*, vol. 44, no. 5, pp. 42–47, 2015. https://doi.org/10.2505/4/jcst15_044_05_42

[55] F. Ouyang and W. Xu, "The effects of educational robotics in STEM education: A multilevel meta-analysis," *Int. J. STEM Educ.*, vol. 11, no. 7, 2024. https://doi.org/10.1186/s40594-024-00469-4

[56] S. Papadakis, "Robots and robotics kits for early childhood and first school age," *Int. J. Interact. Mobile Technol. (iJIM)*, vol. 14, no. 18, pp. 34–56, 2020. https://doi.org/10.3991/ijim.v14i18.16631

[57] S. Papadakis and V. Orfanakis, "The combined use of LEGO Mindstorms NXT and App Inventor for teaching novice programmers," in *Int. Conf. EduRobotics 2016, Cham, Switzerland: Springer Int. Publishing*, 2016, pp. 193–204. https://doi.org/10.1007/978-3-319-55553-9_15

[58] S. Papadakis, "Apps to promote computational thinking concepts and coding skills in children of preschool and pre-primary school age," in *Mobile Learning Applications in Early Childhood Education*, IGI Global, 2020, pp. 101–121. https://doi.org/10.4018/978-1-7998-1486-3.ch006

[59] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*. Reading, MA, USA: Basic Books, 1980.

[60] E. Pasternak, R. Fenichel, and A. N. Marshall, "Tips for creating a block language with Blockly," in *2017 IEEE Blocks and Beyond Workshop*, 2017, pp. 21–24. https://doi.org/10.1109/BLOCKS.2017.8120404

[61] H. Pearson and A. Dube, "3D printing as an educational technology: Theoretical perspectives, learning outcomes, and recommendations for practice," *Educ. Inf. Technol.*, vol. 27, 2022. https://doi.org/10.1007/s10639-021-10733-7

[62] G. Pinto, C. Ferreira, C. Souza, I. Steinmacher, and P. Meirelles, "Training software engineers using open-source software: The students' perspective," in *Proc. IEEE/ACM ICSE-SEET*, Montreal, QC, Canada, 2019, pp. 147–155. https://doi.org/10.1109/ICSE-SEET.2019.00024

[63] D. Popescu, A. Florea, R. Popescu, and H. Roibu, "3D printing for the development of robots by the students," in *2017 27th EAEEIE Annu. Conf.*, Grenoble, France, 2017, pp. 1–6. https://doi.org/10.1109/EAEEIE.2017.8768725

[64] YouTube, "Programming (with) Ody." [Online]. Available: https://www.youtube.com/channel/UC24eA0Korw4JZeXV6Vsv1Ug. [Accessed: Mar. 21, 2025].

[65] G. Recktenwald and D. Hall, "Using Arduino as a platform for programming, design, and measurement in a freshman engineering course," in *ASEE Annual Conference & Exposition*, Pittsburgh, PA, USA, 2011, pp. 1609.1–1609.23. https://doi.org/10.18260/1-2--18720

[66] M. Resnick *et al.*, "Scratch: Programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009. https://doi.org/10.1145/1592761.1592779

[67] R. Santo, D. Ching, K. Peppler, and C. Hoadley, "Working in the open: Lessons from open-source on building innovation networks in education," *On the Horizon*, vol. 24, no. 3, pp. 280–295, 2016. https://doi.org/10.1108/OTH-05-2016-0025

[68] T. Sapounidis and D. Alimisis, "Educational robotics curricula: Current trends and shortcomings," in *Education in & with Robotics to Foster 21st-Century Skills (EDUROBOTICS 2021)*, in Studies in Computational Intelligence, M. Malvezzi, D. Alimisis, and M. Moro, Eds., vol. 982. Cham, Switzerland: Springer, 2021, pp. 146–159. https://doi.org/10.1007/978-3-030-77022-8_12

[69] C. Schelly, G. Anzalone, B. Wijnen, and J. M. Pearce, "Open-source 3-D printing technologies for education," *Journal of Visual Languages and Computing*, vol. 28, pp. 226–237, 2015. https://doi.org/10.1016/j.jvlc.2015.01.001

[70] M. Schad and W. M. Jones, "The Maker movement and education: A systematic review of the literature," *Journal of Research on Technology in Education*, vol. 52, no. 1, pp. 65–78, 2019. https://doi.org/10.1080/15391523.2019.1688739

[71] A. Sciutti, F. Rea, and G. Sandini, "When you are young, (robot's) looks matter," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2014, pp. 567–573. https://doi.org/10.1109/ROMAN.2014.6926313

[72] N. A. Selcuk, S. Kucuk, and B. Sisman, "Does really educational robotics improve secondary school students' course motivation, achievement and attitude?" *Education and Information Technologies*, vol. 29, pp. 23753–23780, 2024. https://doi.org/10.1007/s10639-024-12773-1

[73] M. Sharples, J. Taylor, and G. Vavoula, "A theory of learning for the mobile age," in *Medienbildung in neuen Kulturräumen*, B. Bachmair, Ed., Wiesbaden, Germany: VS Verlag für Sozialwissenschaften, 2010, pp. 87–99. https://doi.org/10.1007/978-3-531-92133-4_6

[74] S. A. Soomro *et al.*, "Makerspaces fostering creativity: A systematic literature review," *Journal of Science Education and Technology*, vol. 32, pp. 530–548, 2023. https://doi.org/10.1007/s10956-023-10041-4

[75] A. Springer and S. Whittaker, "Progressive disclosure: Empirically motivated approaches to designing effective transparency," in *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*, New York, NY, USA: ACM, 2019, pp. 107–120. https://doi.org/10.1145/3301275.3302322

[76] H. C. Steiner, "Firmata: Towards making microcontrollers act like extensions of the computer," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2009, pp. 125–130.

[77] B. Suresh Kumar, M. S. Hemawathi, P. Dhanasekaran, Vishnu Kumar Kaliappan, and D. Sivaganesan, "Maximizing learning outcomes while minimizing costs: A cost-effective approach to M-Learning," *Int. J. Interact. Mob. Technol.*, vol. 18, no. 11, pp. 4–14, 2024. https://doi.org/10.3991/ijim.v18i11.49095

[78] J. Traxler, "Defining, discussing, and evaluating mobile learning: The moving finger writes and having writ …," *International Review of Research in Open and Distributed Learning*, vol. 8, no. 2, pp. 1–12, 2007. https://doi.org/10.19173/irrodl.v8i2.346

[79] I. Trapero-González, F. J. Hinojo-Lucena, J.-M. Romero-Rodríguez, and A. Martínez-Menéndez, "Didactic impact of educational robotics on the development of STEM competence in primary education: A systematic review and meta-analysis," *Frontiers in Education*, vol. 9, art. no. 1480908, 2024. https://doi.org/10.3389/feduc.2024.1480908

[80] T. Trust and R. W. Maloy, "Why 3D print? The 21st century skills students develop while engaging in 3D printing projects," *Computers in the Schools*, vol. 34, no. 4, pp. 253–276, 2017. https://doi.org/10.1080/07380569.2017.1384684

[81] S. Tselegkaridis and T. Sapounidis, "Exploring the features of educational robotics and STEM research in primary education," *Education Sciences*, vol. 12, no. 5, p. 305, 2022. https://doi.org/10.3390/educsci12050305

[82] E. Upton and G. Halfacree, *Raspberry Pi User Guide*, 4th ed. Hoboken, NJ, USA: Wiley, 2016. https://doi.org/10.1002/9781119415572

[83] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA, USA: Harvard University Press, 1978.

[84] W. Villegas-Ch, D. Buenano-Fernandez, A. M. Navarro *et al.*, "Adaptive intelligent tutoring systems for STEM education: Analysis of the learning impact and effectiveness of personalized feedback," *Smart Learning Environments*, vol. 12, no. 41, 2025. https://doi.org/10.1186/s40561-025-00389-y

[85] K. Wang, G. Y. Sang, L. Z. Huang, S. H. Li, and J. W. Guo, "The effectiveness of educational robots in improving learning outcomes: A meta-analysis," *Sustainability*, vol. 15, no. 5, p. 4637, 2023. https://doi.org/10.3390/su15054637

[86] Y. Zhang, R. Luo, Y. Zhu, and Y. Yin, "Educational robots improve K–12 students' computational thinking and STEM attitudes: Systematic review," *Journal of Educational Computing Research*, vol. 59, no. 7, pp. 1450–1481, 2021. https://doi.org/10.1177/0735633121994070

[87] Y. Zhang and Y. Zhu, "Effects of educational robotics on the creativity and problem solving skills of K–12 students: A meta-analysis," *Educational Studies*, vol. 50, no. 6, pp. 1539–1557, 2022. https://doi.org/10.1080/03055698.2022.2107873

14 AUTHOR

George Lagogiannis is with the Department of Agricultural Economics and Rural Development, Agricultural University of Athens, Athens, Greece (E-mail: lagogian@aua.gr).