Developing a Flipped, Interactive Mobile Robotics Course for Computer Science Students

Anthony J. Clark

Pomona College, Claremont CA, USA
anthony.clark@pomona.edu

Abstract. Teaching robotics to computer science students is challenging due to the field's interdisciplinary nature and a common lack of prior experience with hardware and embedded systems. This paper presents a mobile robotics course designed specifically for computer science undergraduates with minimal robotics experience. The course employs a flipped classroom model with interactive web-based materials featuring simulations and visualizations to bridge abstract mathematical concepts with robot behavior. Students build differential-drive robots from off-the-shelf components based on an ESP32-S3 microcontroller; they learn prototyping, electronics, and programming skills. The curriculum progresses from high-level programming in Toit to Arduino C/C++, covering topics from concurrency to kinematics and motion planning. All course materials, interactive diagrams, hardware designs, and software are freely available online.

Keywords: undergraduate course, mobile robot, flipped, interactive materials, computer science

1 Introduction

Teaching robotics is a challenge due to the interdisciplinary nature of our field. Luckily, a variety of high-quality resources already exist. The freely available *Modern Robotics* [21] is an excellent, traditional textbook that includes companion videos and a corresponding MOOC (massive open online course). However, one challenge with MOOCs is that they encourage students to pay for a certificate by removing core functionality if the student does not pay (i.e., assessments and automated feedback). MOOCs are also often removed without notice, as was the case for the courses comprising Penn's robotics specialization [17]. Dr. Peter Corke's *Robotics, Vision & Control* is another wonderful textbook with recorded lecture videos available on his Robotic Academy website [7,6], and *Robotics Book* [9] is a free book available as a website, developed for Georgia Tech's Introduction to Robotics and Perception course. Similarly, Correll et al. developed *Introduction to Autonomous Robots* [8], and made their course materials (including labs and assignments) freely available online.

Other content is available only as online videos, such as the *How to Build Robots and Make them Move* course developed by Dr. Rouse at the University

of Michigan [26], and Dr. Berry's *Mobile Robotics* course at Role-Hullman Institute of Technology [3]. These videos do not include accompanying materials (assignments, labs, slides).

These are all important learning resources, however, they are non-interactive, can be expensive or incomplete, and are written more for engineering students than computer science students. Additionally, students often skip non-interactive reading and video assignments [15,14], and while they can be encouraged with low-stakes quizzes [13,29], it is generally better to make content more interactive [24,5]. *ZyBooks* and *Brilliant* are effective (commercial) services that provide interactive textbooks-like experiences, but they are not open source and require paid subscriptions [5]. Moreover, students lose access to content after their subscription ends.

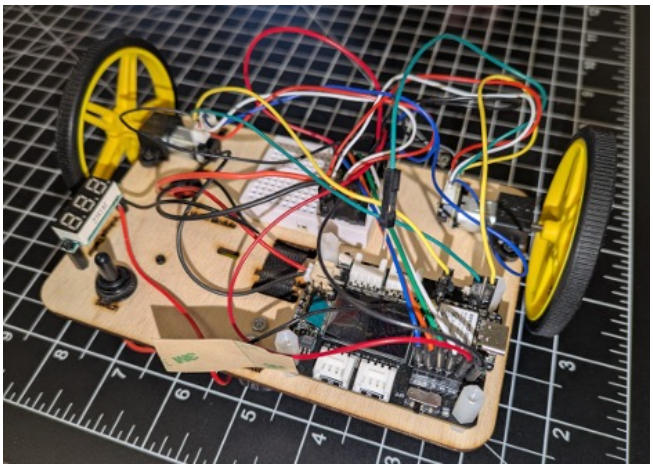


Fig. 1: The course robot platform. The robot was built using off-the-shelf components and an inexpensive microcontroller.

Along with engaging educational materials, hands-on experience is critical for learning robotics. Real-world systems are noisy and imperfect, and students benefit from experiencing these challenges first-hand. Accordingly, many mobile robotics platforms are available for educational use, ranging from simple kits to advanced research platforms. Vourkos et al. taught fundamental feedback control concepts using a differential drive robot based on a Raspberry Pi [30], and Roux et al. developed a course for teaching control theory using a small, commercial airplane platform [27]. The EMARO [11], Baratinha [1], and AMiRo [16] are low-cost, differential drive robots designed with custom PCBs for education and research. Similarly, Robotont [25] and EduRob [18] are omnidirectional robot kits developed for robotics education. Turtlebots, Crazyflies, Jetbots, among others, are popular commercial platforms for robotics education and research. Different from these platforms, our goals are better supported using off-the-shelf

components and an inexpensive microcontroller. An example robot built using our design is shown in Figure 1.

The main contribution of this work is our mobile robotics course that is:

1. focused on the safety, prototyping, and development processes,
2. designed for undergraduate computer science students (aged 20-22) with minimal related background,
3. developed with low-cost components based on a microcontroller,
4. designed for a flipped classroom with interactive, web-based materials, and
5. freely available to the broader robotics education community.

The remainder of this paper is organized as follows. In Section 2, we describe the interactive, web-based materials developed for the course. In Section 3, we describe our robot hardware and the prototyping process developed for the class. In Section 4, we describe the software stack and programming languages used in the course. Finally, in Section 5, we discuss our experiences teaching the course and our future work.

2 Interactive Materials

The course was "flipped" such that students used meeting time for hands-on activities rather than traditional lectures [4]. The course website served as an interactive textbook, where each chapter corresponded roughly to a single meeting in a fifteen-week semester with two meetings per week. The general flow of the course followed these steps:

1. students watched lecture videos embedded in the course website and read related materials before class,
2. class periods started with a short quiz to encourage students to complete the preparatory work and receive feedback on their understanding [4],
3. class periods continued with a brief review of key concepts needed for the day, and then
4. students completed a hands-on activity related to the material.

Since all course materials (reading assignments, lecture videos, robot software, robot hardware, etc.) are available on the course website, anyone can build the robot and work through the assignments independently. Table 1 lists the main modules of the course along with topics covered in each module. The course website includes a more complete concept inventory [12] and a robotics topics mind map.

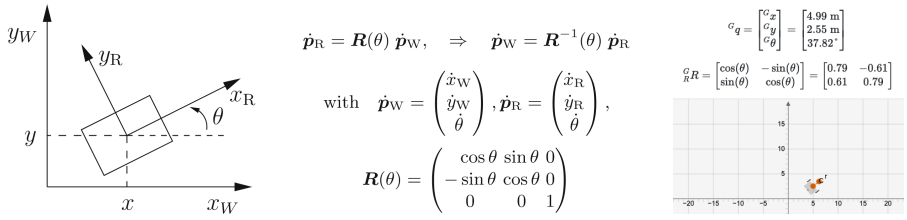
The course started with electronics safety and prototyping. Students were shown how the robot hardware (microcontroller, motors, motor driver, etc.) was selected based on a set of design requirements and reading datasheets. The first hands-on activity involved students laying out a prototype using cardboard—we did not provide advice on the placement of motors or wheels or give strict requirements on the robot's dimensions. This step helped students understand

Module	Topics
Design and Electronics	Safety, electronics, design, and prototyping
Embedded Systems	Programming, communication, and embedded software.
Modeling and Control	Kinematics, sensor fusion, and position control.
Maps and Motion Planning	Simulation, mapping, and path planning.
Societal Impacts	Job impact, history, and ethics.

Table 1: Course modules and topics covered.

design trade-offs and spatial constraints (i.e., a differential drive robot is typically wider than it is long and they were instructed to minimize the overall footprint). Students were next introduced to basic CAD design for laser-cutting plywood and 3D-printing components (e.g., how to account for tolerances and iterate on designs). Most courses skip these early design steps in favor of providing students with a pre-built robot. However, since our students are primarily computer science majors with limited experience in hardware design, we found that these activities helped them gain confidence in prototyping and designing their own devices, which is a useful skill for any IoT application. The trade-off is that students spend less time on advanced robotics topics, but we believe the benefits outweigh this cost for our target audience.

After the initial design and prototyping module, students were introduced to hardware safety and wireless communication. Specifically, we instructed them on best practices for working with electronics and Li-ion batteries. The battery level was always displayed using a voltmeter connected at the switch, and students were instructed to monitor the level and never let it drop below a safe threshold. After writing a simple LED blink program, students implemented a WiFi-based heartbeat protocol to confirm communication between their robot and laptop. All remaining software libraries (e.g., motor control, path planning, etc.) were only activated if a regular heartbeat signal was received over WiFi. Our approach stressed the importance of safety when designing software for an autonomous system.

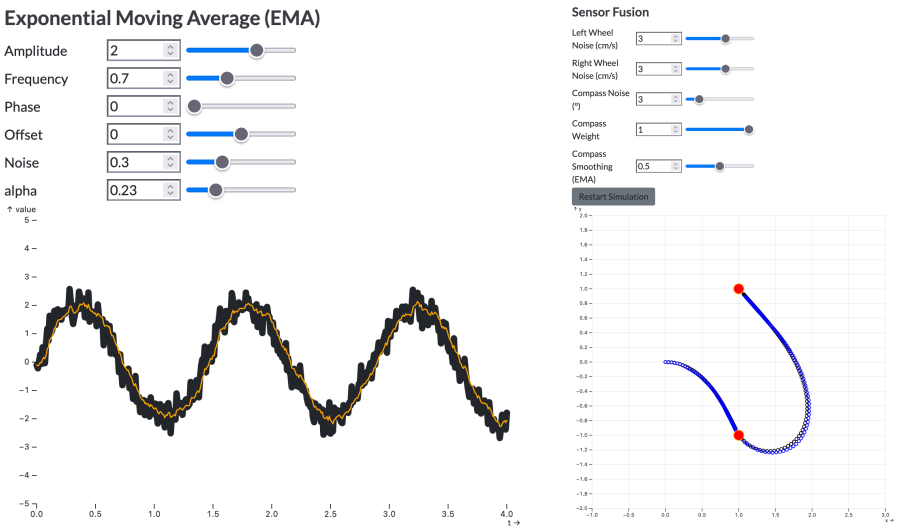


(a) Static (b) Interactive

Fig. 2: Static and interactive kinematics diagrams. (a) A screenshot of a figure from *EduRob: An Educational Robot for Teaching Kinematics of Wheeled Mobile Robots* by Krüger et al. [18]. (b) A screenshot of a kinematics diagram from our course website in which a user can drag the robot and see immediate changes to state values.

Subsequent modules covered progressively more advanced robotics topics, starting with embedded systems and basic control, then modeling and feedback control, and finally mapping and motion planning. The most advanced topic involved using the vision module for simple object detection and avoidance. Throughout each module, students were also introduced to societal impacts of robotics, including ethical considerations, job displacement, and legal issues.

A key component of all course materials was interactive diagrams and simulations embedded directly in the course website. Figure 2a shows an excellent example from Krüger et al. [18] that illustrates the kinematics of a differential drive robot. One issue with a static diagram and its accompanying explanation is the gap between theory and motion. Students cannot always visualize how equations correspond to real-world movement.



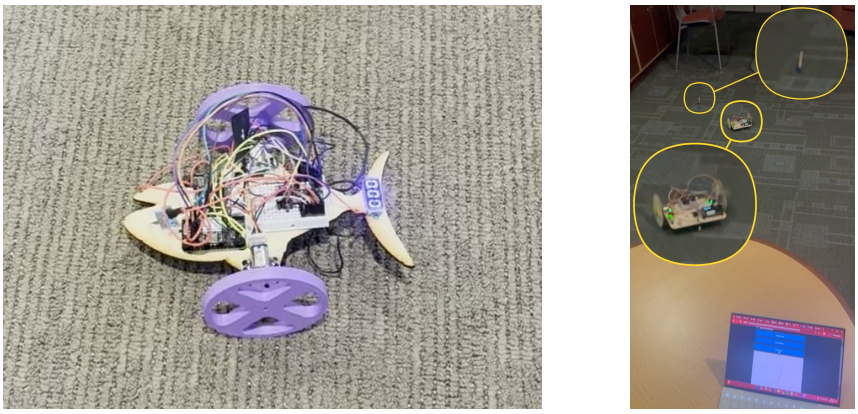
(a) Exponential Moving Average (b) Sensor Fusion

Fig. 3: Interactive diagrams for the exponential moving average and sensor fusion. As students adjust the sliders, the diagrams update in real-time.

Figure 2b shows an interactive version of the same diagram from our course website. We chose a different notation for our course materials, but the core concepts are the same. In our interactive version, students can drag and rotate the robot and then see how this affects the state and rotation matrices (the equations are live updated as the robot is manipulated). Some other robotics teaching material include interactive content, but often as scripts or Jupyter notebooks that students must download and run after installing necessary dependencies [28]. Many of our interactive diagrams and simulations were developed using JSXGraph [22,2], and all of them run directly in the web browser.

Figure 3 shows two additional interactive diagrams from the course website. Figure 3a illustrates an exponential moving average (EMA), which is used for smoothing noisy sensor data. Students can adjust the smoothing factor and see how this affects the filtered output in real-time. Figure 3b demonstrates sensor fusion using a complementary filter to combine data from a magnetometer and motor encoders. Students can manipulate the noise levels of each sensor and observe how the fused output changes accordingly. These interactive diagrams help students build intuition about complex concepts by allowing them to experiment and visualize the effects of different parameters. The course website also includes several simulations with noisy sensing and actuation. As part of an exercise students are frequently asked to run the simulation multiple times and report on the impacts of different noise levels.

Figure 4a shows a custom robot chassis in the shape of a shark with 3D printed wheels—students were encouraged to be creative with their designs. In Figure 4b, we highlight a target location and their robot prototype executing a motion planning algorithm to navigate to that location. In the figure we also display a dashboard on the laptop, which plots live localization data being streamed from the robot over WiFi. Students were able to see how their algorithms' predictions differed from actual robot motion.



(a) Student Prototype (b) Motion Planning

Fig. 4: (a) A prototype student robot with custom laser-cut and 3D printed parts, and (b) a student submission demonstrating motion planning capabilities.

In addition to the chapters including interactive content, since the textbook was developed as a website students were able to use social annotation tools (i.e., Hypothes.is) to take notes, ask questions, and discussion course materials asynchronously with their peers.

3 Robot Hardware

We selected hardware that was suitable for a computer science curriculum and easily accessible to students outside of the course. We chose the ESP32-S3 microcontroller due to its balance of performance, ease of use, cost, and wireless connectivity. The Espressif ESP32 series is a popular choice for educational robotics [1,18,16,20,10]. We designed a simple differential-drive robot that students could easily assemble themselves using cardboard (or laser cut plywood and 3D printed parts if they were available). Other highlights of our hardware design include 18650 lithium-ion batteries with built-in protection circuits, a vision module[1], an expansion module for the microcontroller[2], and a multizone time-of-flight distance sensor[3]. A full list of parts and materials is available on the robot parts list from our course website.

Capped 18650 lithium-ion batteries were chosen for their safety, availability, and relative ease of use. Since most battery holders do not accommodate capped 18650 cells, we designed a custom 3D-printed battery holder and provided the CAD files on the course website. Our lithium-ion 18650 batteries are rated at 3.7V and 2600mAh. The expansion board and motors are powered directly from the battery; the system can run for up to two hours. The expansion board includes a power switch, OLED display, and a common I2C connector port for peripherals. The display was particularly useful for debugging and displaying important information like WiFi status and IP addresses. The ToF sensor enabled simple obstacle detection and distance measurements in multiple directions as well as the iterative closest point (ICP) algorithm. Although the ESP32-S3 is capable of handling vision processing tasks, the vision module takes processing load off the microcontroller freeing it to run other complex algorithms.

Students were only given a subset of components at the beginning of the term, and additional items were introduced as the course progressed. This approach allowed students to focus on learning one concept at a time without being overwhelmed by the full complexity of the robot. For example, students first built a basic differential-drive robot with just the microcontroller, motors, motor driver, and battery. Once they had mastered basic movement and control, we introduced the time-of-flight distance sensor for obstacle detection and avoidance. Finally, we added the vision module for more advanced tasks like object recognition and tracking.

4 Robot Software

The target audience for this course is computer science undergraduates with little to no prior experience with electronics or embedded systems. Pomona College computer science students are only required to complete nine courses in the major, and for most students in the class, this is their first exposure to these

[1]The Grove Vision AI v2 Kit from Seeed Studio.
[2]The Seeed Studio Expansion Board Base for the Seeed Studio XIAO series.
[3]The Time-of-Flight (ToF) multizone ranging sensor from ST.

topics. Our course is designed to teach them both robotics concepts and practical skills for prototyping and developing any IoT device. Students in the course only had experience programming in environments with operating systems—this was their first experience programming a device without virtual memory.

Given this audience, we decided to start with a high-level language that abstracts away many of the low-level details of embedded programming. We explored several options including WARDuino [19], TinyGo, DeviceScript, and Berry Script. After evaluating these options, we ultimately selected the Toit Programming Language.

Listing 1.1 Toit code showing cooperative multitasking.

```
import gpio

LED1 ::= gpio.Pin.out 17
LED2 ::= gpio.Pin.out 18

main:
    task:: my-task-1
    task:: my-task-2

my-task-1:
    while true:
        sleep --ms=500
        LED1.set 1
        sleep --ms=500
        LED1.set 0

my-task-2:
    while true:
        sleep --ms=123
        LED2.set 1
        sleep --ms=123
        LED2.set 0
```

Listing 1.2 Arduino C/C++ code showing poll-based logic.

```
#include "intervaltimer.h"

IntervalTimer led1Timer(500);
const int led1Pin = 17;
int led1State = LOW;

IntervalTimer led2Timer(123);
const int led2Pin = 18;
int led2State = LOW;

void setup() {
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);
}

void loop() {
  if (led1Timer) {
    led1State = !led1State;
    dw(led1Pin, led1State);
  }
  if (led2Timer) {
    led2State = !led2State;
    dw(led2Pin, led2State);
  }
}
```

Toit was chosen for its live reloading capabilities and interesting syntax for cooperative multitasking. As suggested by Moran et al. [23], rapid feedback loops are a positive factor in learning robotics. Using Toit and its tooling, students were able to code on their laptops and upload new binaries to their microcontrollers over WiFi. This quick feedback loop allowed students to rapidly iterate on their code without needing to set up serial connections or re-flash the device using a USB cable. For computer science students, this was a much more familiar work-

flow similar to web development or scripting. Likewise, Toit's syntax resembles Python, which is a language used in the introductory computer science course at Pomona College. We did not provide boilerplate code, and students coded all software modules from scratch. One exception was a WiFi library handling the heartbeat protocol discussed in Section 2.

Listings 1.1 and 1.2 show how to blink two LEDs without a blocking delay using Toit and Arduino C/C++, respectively. The Arduino code includes a simple interval timer library to avoid using the blocking `delay()` function. The Toit code using cooperative multitasking is conceptually easier to explain to students already familiar with high-level programming languages, and students often find the Arduino code more foreign and difficult to understand since they are not in control of the main loop.

Although Toit was an excellent language for the beginning of the course, by the fourth week of the course our software stack was sufficiently complex that we began to encounter limitations with the language and runtime. Namely, the memory requirements of the Toit runtime made it infeasible to implement more complex systems—the microcontroller ran out of memory when motion control algorithms were added and executed alongside the WiFi heartbeat protocol and sensor processing. At this stage in the course, we transitioned students to programming the microcontroller using C/C++ and the Arduino framework.

5 Discussion

Our approach to teaching mobile robotics emphasizes hands-on experimentation, rapid feedback loops, and gradually increasing complexity. The flipped classroom design paired with interactive digital materials proved effective for computer science students with minimal prior exposure to embedded systems and hardware design. We hope robotics learners outside of our institution can benefit from the open-source course materials, robot designs, and software stack developed for this course.

Activities like cardboard prototyping, laser-cutting, and 3D printing design provided hands-on experience with iterative development. Unlike many robotics courses that provide pre-built robots, our students gained confidence in prototyping and hardware design—skills that extend beyond robotics to any IoT application development. Anecdotally, after the course ended, two students entered and won first prize at a Hackathon in which they built an IoT device using skills learned from prototyping and hardware design in this course.

The explicit focus on safety, particularly with the WiFi-based heartbeat protocol, reinforced good design practices for autonomous systems. By requiring all advanced features to be gated behind a regular heartbeat signal, students learned that safety mechanisms should be integrated into software architecture from the beginning rather than added as an afterthought.

Using the Toit language proved beneficial for the early weeks of the course. The language's cooperative multitasking syntax made non-blocking I/O accessible to students unfamiliar with embedded systems concepts. The ability to reload

code over WiFi created a familiar development workflow similar to web development or scripting, reducing friction for computer science students accustomed to rapid iteration cycles. However, the transition to Arduino C/C++ after week four was necessary when algorithm complexity exceeded memory limitations. This transition, while somewhat abrupt, served as a valuable learning experience about the constraints of embedded systems and the trade-offs between abstraction and resource efficiency.

Our approach does have limitations. The prioritization of prototyping and design activities necessarily reduced time available for advanced robotics topics compared to curricula using pre-built platforms. For example, we covered fewer topics in advanced motion planning and computer vision than our initial goals envisioned. However, for our target audience of computer science majors with minimal prior hardware experience, we believe this trade-off supported better overall learning outcomes and increased confidence in applied systems work.

Our future work includes creating additional interactive materials to cover advanced topics like localization, mapping, and computer vision; developing a custom interactive shell for the microcontroller so students can experiment with embedded systems concepts (e.g., bitwise operations) while interacting with the robot (e.g., turning on LEDs); and experimenting with other systems programming languages such as Rust [31].

Acknowledgements

This work was supported by Pomona College. The authors would like to thank Ella Zhu and Jack Chin for their contributions.

References

1. Araújo, J.A.C., Sá, R.C., Junior, J.L.W.O., de Barros Serra, A.: A small, smart, and connected educational robot for university classrooms. In: Robotics in Education, pp. 523–532. Springer Nature Switzerland (2025). DOI 10.1007/978-3-031-98762-5_44
2. Center for Mobile Learning with Digital Technology University of Bayreuth, G.: Jsxgraph. URL https://jsxgraph.org
3. Berry, C.: Mobile robotics. YouTube (2017)
4. Berry, C.A.: Robotics education online flipping a traditional mobile robotics classroom. In: 2017 IEEE Frontiers in Education Conference (FIE), pp. 1–6 (2017). DOI 10.1109/FIE.2017.8190719. URL https://ieeexplore.ieee.org/abstract/document/8190719
5. Cooper, G., Sullivan, W.S.: Improving student experience in an introductory programming course with an interactive textbook. In: 2023 IEEE World Engineering Education Conference (EDUNINE), pp. 1–6 (2023). DOI 10.1109/EDUNINE57531.2023.10102903
6. Corke, P.: Robotics, Vision and Control: Fundamental Algorithms in Python, *Springer Tracts in Advanced Robotics*, vol. 146. Springer International Publishing (2023). DOI 10.1007/978-3-031-06469-2

7. Corke, P., Greener, E., Philip, R.: An innovative educational change: Massive open online courses in robotics and robotic vision. IEEE Robotics & Automation Magazine **23**(2), 81–89 (2016-06). DOI 10.1109/MRA.2016.2548779
8. Correll, N., Hayes, B.: Introduction to Autonomous Robots: Mechanisms, Sensors, Actuators, and Algorithms, 1st edn. MIT Press, Cambridge, MA (2022)
9. Dellaert, F., Hutchinson, S.: Robotics Book: Introduction to Robotics and Perception. n.p. (2022). URL https://www.roboticsbook.org/intro.html
10. Eguchi, A.: Understanding machine learning through AI-powered educational robotics - pilot study with undergraduate students. In: Robotics in Education, pp. 52–62. Springer International Publishing (2022). DOI 10.1007/978-3-031-12848-6_5
11. Gehricke, P., Tassemeier, M., Porrmann, M.: EMAROs: A modular autonomous robot-platform for research and education. In: Robotics in Education, pp. 533–544. Springer Nature Switzerland (2025). DOI 10.1007/978-3-031-98762-5_45
12. Gerndt, R., Lüssem, J.: The robotics concept inventory. In: Robotics in Education, pp. 18–27. Springer International Publishing (2019)
13. Gokcora, D., DePaulo, D.: Frequent quizzes and student improvement of reading: A pilot study in a community college setting. SAGE Open **8**(2), 2158244018782,580 (2018). DOI 10.1177/2158244018782580. Publisher: SAGE Publications
14. Gorzycki, M., Desa, G., Howard, P.J., Allen, D.D.: "reading is important," but "i don't read": Undergraduates' experiences with academic reading. Journal of Adolescent & Adult Literacy **63**(5), 499–508 (2020). DOI 10.1002/jaal.1020. _eprint: https://ila.onlinelibrary.wiley.com/doi/pdf/10.1002/jaal.1020
15. Hoeft, M.: Why university students don't read: What professors can do to increase compliance. International Journal for the Scholarship of Teaching and Learning **6**(2) (2012). DOI 10.20429/ijsotl.2012.060212
16. Klarhorst, C., Quirin, D., Hesse, M., Rückert, U.: Insights from a decade of AMiRo: Where research meets education. In: Robotics in Education, pp. 247–259. Springer Nature Switzerland (2024). DOI 10.1007/978-3-031-67059-6_22
17. Koditschek, D.E., Kumar, V., Taylor, C.J.: Robotics specialization. Coursera, University of Pennsylvania (2023). Accessed: 2025-07-14
18. Krüger, J.E., Miller, A., Lel, A., Erbach, L., Unkrig, R., Röhrig, C.: EduRob: An educational robot for teaching kinematics of wheeled mobile robots. In: Robotics in Education, pp. 283–294. Springer Nature Switzerland (2024). DOI 10.1007/978-3-031-67059-6_25
19. Lauwaerts, T., Gurdeep Singh, R., Scholliers, C.: Warduino: An embedded webassembly virtual machine. Journal of Computer Languages p. 101268 (2024). DOI 10.1016/j.cola.2024.101268
20. Luzardo, G., Luong, H., Vlaminck, M., Philips, W., Ochoa, D.: Smartdrone: An open platform for research and experimentation with autonomous UAVs. In: Robotics in Education, pp. 295–306. Springer Nature Switzerland (2024). DOI 10.1007/978-3-031-67059-6_26
21. Lynch, K.M., Park, F.C.: Modern robotics: mechanics, planning, and control. Cambridge University Press (2017)
22. Lähteenmäki, H., Hurme, J., Porras, P.: Design of interactive STACK exercises using JSXGraph for online course: Exploring strategies for supporting students with mathematical challenges. URL http://www.theseus.fi/handle/10024/854379
23. Moran, R., Zabala, G.: On the importance of live programming and short feedback loops for educational robotics. In: Robotics in Education, pp. 65–76. Springer Nature Switzerland (2024). DOI 10.1007/978-3-031-67059-6_7

24. P.E., A.B., Hariharan, J.: Effectiveness of online web-native content vs. traditional textbooks. In: 2021 ASEE Virtual Annual Conference Content Access, 10.18260/1-2–37011. ASEE Conferences, Virtual Conference (2021). Https://peer.asee.org/37011
25. Raudmäe, R., Vunder, V., Kandlhofer, M., Breiling, B., Sumper-Sabitzer, M., Mõtshärg, E., Schumann, S., Kruusamäe, K.: Open source and portable educational kits for enabling robotics education. In: Robotics in Education, pp. 275–282. Springer Nature Switzerland (2024). DOI 10.1007/978-3-031-67059-6_24
26. Rouse, E.: How to build robots and make them move. YouTube (2023)
27. Roux, U., Baltes, J., Gerndt, R., Saeedvand, S.: Introduction to aeronautics and robotics control using a small electric airplane. In: Robotics in Education, pp. 219–230. Springer Nature Switzerland (2025). DOI 10.1007/978-3-031-98762-5_19
28. Sakai, A., Ingram, D., Dinius, J., Chawla, K., Raffin, A., Paques, A.: PythonRobotics: a python code collection of robotics algorithms. DOI 10.48550/arXiv.1808.10703. URL http://arxiv.org/abs/1808.10703
29. Sotola, L.K., Crede, M.: Regarding class quizzes: a meta-analytic synthesis of studies on the relationship between frequent low-stakes testing and class performance. Educational Psychology Review **33**(2), 407–426 (2021). DOI 10.1007/s10648-020-09563-9
30. Vourkos, E.G., Negkoglou, C., Avgousti, S., Masouras, P., Panayides, A.S., Christoforou, E.G.: Using educational robotics to teach fundamental feedback control concepts. In: Robotics in Education, pp. 231–239. Springer Nature Switzerland (2025). DOI 10.1007/978-3-031-98762-5_20
31. Zhou, Y.: R4r workshop (2026). URL https://sites.google.com/view/r4rworkshop/home