Review article

Code Generation, Context Loss, and Cognitive Load: Lessons from Integrating AI Assistants into Engineering Curriculum

Antonio Carlos Bento[a]*, Marcos Ribeiro Pereira Barretto[b], José Reinaldo Silva[b], Sérgio Camacho-León[a], Elsa Yolanda Torres-Torres[a], Carlos Vazquez-Hurtado[a]

[a] *Computer Science and Engineering School, Instituto Tecnologico y de Estudios Superiores de Monterrey, Monterrey, Nuevo León, México*
[b] *Dept. Mechatronics Engineering, Universidade de São Paulo, Brazil*

A R T I C L E I N F O

Keywords:
Engineering
Artificial Intelligence
Software Development
Robots

A B S T R A C T

Background: The convergence of artificial intelligence and robotics is creating new possibilities for autonomous systems, yet the integration of AI-powered control in educational settings presents unique pedagogical challenges. While research has explored the technical capabilities of AI-driven robots, less attention has been paid to how students learn to design, build, and troubleshoot these complex, intelligent systems in project-based environments.

Objective: This study investigates how engineering students interact with AI-powered robot control systems throughout the design and testing lifecycle, examining both the learning affordances and cognitive challenges that emerge during authentic project work.

1. Introduction

1.1 The Changing Landscape of Robotics Education

The rapid advancement of artificial intelligence has fundamentally transformed robotics, with AI-powered control systems becoming increasingly sophisticated and accessible. NeuralCoder, introduced in 2020, demonstrated that large language models could generate contextually relevant code suggestions, fundamentally altering the relationship between developers and their tools [1]. Subsequent platforms including DeepTalk, SmartIDE, ThinkAI, and CloudCode have further expanded the capabilities available to programmers, creating what some have termed a "copilot paradigm" in software development [1]. For engineering education, these developments present both unprecedented opportunities and significant challenges. The opportunity lies in potentially accelerating student progress, reducing time spent on low-level control logic, and allowing learners to engage with more complex system-level problems earlier in their development [2]. The challenge, however, is equally profound: if students come to rely on AI-generated control code without developing fundamental comprehension of electromechanical systems and feedback loops, we risk producing graduates who can operate tools but cannot truly engineer robust robotic solutions [3].

1.2 Beyond Productivity: The Need for Pedagogical Understanding

Existing research on AI-assisted development has predominantly focused on productivity metrics. Studies by Kim et al. [4] and Garcia et al. [5] have documented efficiency gains of 35-55% for specific programming tasks when using tools like NeuralCoder. While valuable for understanding potential workplace impacts, these productivity-oriented studies provide limited guidance for educators seeking to integrate AI tools into learning environments where the primary outcome is not code output but student development. The pedagogical questions are fundamentally different: How do students learn to design a robot's architecture when an AI can generate a control loop on demand? What cognitive skills are developed, and which are potentially atrophied? How do novices develop the critical judgment necessary to evaluate an AI's suggested motor parameters or sensor fusion algorithms when their own understanding of the physical system is still forming? These questions require qualitative investigation into the lived experiences of students navigating AI-augmented robotics laboratories [6].

1.3 Research Objectives

RQ1: How do engineering students interact with AI-powered robot control systems during project-based design and testing, and what patterns of use emerge over time?

RQ2: What cognitive challenges and learning affordances do students experience when integrating AI tools into their robotics workflow?

RQ3: What pedagogical frameworks can support effective AI literacy development in robotics education?

By addressing these questions through detailed qualitative inquiry, this study aims to provide empirically grounded guidance for curriculum design that harnesses the benefits of AI tools while

* Corresponding authors.

DOI
Received 99 Month 2025; Received in revised from 28 October 2025; Accepted XX Month 2025
Available online 99 Month 2025

mitigating their potential drawbacks.

1.4 Connection to Broader Educational Goals

This research aligns with Sustainable Development Goal 4 (Quality Education), which emphasizes inclusive and equitable quality education and lifelong learning opportunities for all [7]. Understanding how AI tools can be effectively integrated into engineering education is essential for ensuring that all students, regardless of their prior experience with robotics, can benefit from these technologies. Without intentional pedagogical design, AI tools risk widening the gap between students who can critically evaluate AI-suggested designs and those who passively accept them, potentially leading to unsafe or ineffective robots [8].

2. Theoretical Framework

2.1 Cognitive Load Theory in AI-Augmented Learning

Cognitive Load Theory [9] provides a useful lens for understanding how AI tools may impact student learning. The theory distinguishes between intrinsic cognitive load (demands inherent to the learning task), extraneous cognitive load (demands created by instructional design), and germane cognitive load (demands that contribute to schema construction).

AI-powered robot control systems have the potential to reduce extraneous cognitive load by handling low-level PID loop tuning or inverse kinematics calculations, freeing working memory for higher-level system integration and mission planning [10]. However, our preliminary observations suggested that AI tools might also introduce novel forms of cognitive load: the need to evaluate the feasibility of AI-generated control code in the real world, diagnose unexpected physical behaviors caused by AI suggestions, and maintain contextual awareness of the robot's state across simulated and real-world interactions. Understanding this trade-off is central to designing effective learning experiences.

2.2 Constructivist Learning Theory and Scaffolding

Constructivist approaches to education emphasize that learners actively construct knowledge through experience and reflection [11]. Within this framework, AI tools can be understood as potential scaffolds that temporarily support student performance beyond their independent capability, following Dewey's concept of the Zone of Proximal Development [12].

However, effective scaffolding requires intentional fading - the gradual removal of support as learners develop capability [13]. If AI tools remain as permanent scaffolds, students may never develop the independent competence required to diagnose a faulty sensor or tune a controller for a new robotic platform. This study examines how students navigate this tension and what instructional supports might facilitate appropriate scaffold fading.

2.3 Human-AI Collaboration Frameworks

Recent work in human-computer interaction has begun to theorize the nature of human-AI collaboration. Shneiderman et al. [14] proposed guidelines for human-AI interaction, emphasizing the importance of clear communication of AI capabilities and limitations. In educational contexts, these guidelines must be adapted to account for the developmental trajectory of learners who may not yet have the expertise to accurately assess AI output, especially when that output dictates the physical movement of a robot [15].

3. Methodology

3.1 Research Design

This study employed a qualitative case study approach [16] to investigate the lived experiences of students integrating AI tools into their robotics development practice. Case study methodology is particularly appropriate for research questions focused on understanding complex phenomena within real-world contexts, where the boundaries between phenomenon and context are not clearly evident.

3.2 Participants and Setting

Forty-two fourth-semester mechatronics engineering students enrolled in a project-based robotics control course at a large private university in Colombia participated in the study. The course, titled "AI-Augmented Robotics," was designed to introduce students to contemporary development practices for intelligent robots while maintaining focus on foundational principles of mechanics, electronics, and control theory.

Participants ranged in age from 20 to 25 years, with varied prior robotics experience. Eighteen students reported significant prior experience with microcontroller programming and basic robotics (defined as having completed at least two previous courses or personal projects involving platforms like Arduino or ROS), fifteen reported moderate experience, and nine identified as beginners with limited exposure beyond introductory coursework. This diversity enabled examination of how prior experience shapes AI tool interaction patterns.

Students self-selected into seven project teams, with the instructor ensuring that each team would utilize a different primary AI tool for their development work. The tools distributed were: NeuralCoder (Team Alpha), DeepTalk (Team Beta), SmartIDE (Team Gamma), ThinkAI (Team Delta), and CloudCode (Team Epsilon). Team Zeta was designated to use DeepTalk but with a different project focus, enabling comparison of tool use across similar contexts.

3.3 The Project Context: The Autonomous "SalesMaster" Delivery Robot

All teams worked on developing "SalesMaster," an autonomous delivery robot designed for an office environment. The robot was required to navigate hallways, avoid dynamic obstacles, and deliver small packages. The project requirements included:

Robust mobile platform with differential drive

Sensor suite integration (LiDAR, IMU, wheel encoders)

Autonomous navigation and path planning using ROS

User interface for specifying delivery destinations

Safe emergency stop and fault recovery behaviors

Integration of a simulated physics model in Gazebo for initial testing

This project was deliberately designed to encompass diverse technical challenges: low-level motor control (C++/Arduino), mid-level sensor processing (Python), high-level system integration (ROS), and simulation development (Gazebo). The variety of technical contexts allowed us to observe how students utilized AI tools across different aspects of robotic system design, from tuning low-level PID controllers to architecting high-level state machines.

References

1. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W.,

Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., & Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

2. Denny, P., Prather, J., Becker, B.A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B.N., Santos, E.A., & Sarsa, S. (2024). Computing education in the era of generative AI. Communications of the ACM, 67(2), 56-67.
3. Prather, J., Denny, P., Leinonen, J., Becker, B.A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B.N., & Savelka, J. (2023). The robots are here: Navigating the generative AI revolution in computing education. In Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (pp. 1-47).
4. Li, Z., Wang, S., Zheng, W., Chen, X., Zhao, P., & Jiang, L. (2023). Measuring coding efficiency gain in AI-assisted programming: A controlled experiment with GitHub Copilot. IEEE Transactions on Software Engineering, 49(8), 3987-4001.
5. Meyer, A.N., Barr, E.T., Bird, C., & Zimmermann, T. (2023). Today was a good day: The daily life of software developers during the AI transformation. In Proceedings of the 45th International Conference on Software Engineering (pp. 1234-1245).
6. Lau, S., & Guo, P. (2023). From "ban it" to "understand it": Student and instructor perspectives on AI tool use in computing education. In Proceedings of the 2023 ACM Conference on International Computing Education Research (pp. 234-246).
7.
8. United Nations. (2025). Sustainable Development Goals. https://sdgs.un.org/goals
9. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274.
10. Sweller, J., van Merriënboer, J.J.G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. Educational Psychology Review, 31(2), 261-292.
11. Kazemitabaar, M., Chow, J., Ma, C.K., Ericson, B.J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1-23).
12. Piaget, J. (1970). Science of education and the psychology of the child. Viking Press.
13. Vygotsky, L.S. (1978). Mind in society: The development of higher psychological processes. Harvard University Press.
14. Pea, R.D. (2004). The social and technological dimensions of scaffolding and related theoretical concepts for learning, education, and human activity. Journal of the Learning Sciences, 13(3), 423-451.
15. Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for human-AI interaction. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (pp. 1-13).
16. Yang, Q., Steinfeld, A., & Zimmerman, J. (2023). Unpacking human-AI interaction in creative practices. ACM Transactions on Computer-Human Interaction, 30(2), 1-32.
17. Yin, R.K. (2018). Case study research and applications: Design and methods (6th ed.). Sage Publications.
18. Braun, V., & Clarke, V. (2021). Thematic analysis: A practical guide. Sage Publications.
19. Lincoln, Y.S., & Guba, E.G. (1985). Naturalistic inquiry. Sage Publications.
20. Ericsson, K.A., & Pool, R. (2016). Peak: Secrets from the new science of expertise. Houghton Mifflin Harcourt.
21. Qian, Y., Zhang, L., & Xu, B. (2024). Architectural decision making with AI assistants: An empirical study. IEEE Transactions on Software Engineering, 50(3), 456-472.
22. Nguyen, T., Chen, J., & Wang, H. (2024). Prompt engineering for AI programming assistants: A systematic review. IEEE Software, 41(2), 67-75.
23. Patel, S., & Williams, J. (2023). Cognitive load in AI-augmented development: A qualitative study. International Journal of Human-Computer Studies, 178, 103098.
24. Roberts, E., Johnson, K., & Smith, P. (2023). Version control patterns for AI-generated code. In Proceedings of the IEEE International Conference on Software Maintenance and Evolution (pp. 234-245).
25. Taylor, M., Anderson, D., & Martinez, R. (2024). Human-AI collaboration in software testing: Practices and challenges. Software Testing, Verification and Reliability, 34(2), e1892.
26. Wang, J., Zhang, Y., & Liu, W. (2023). Bias in AI programming assistants: A large-scale analysis. In Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (pp. 567-578).
27. Xu, B., Chen, L., & Wang, D. (2024). AI-generated frontend code: Opportunities and challenges for web development. In Proceedings of the International Conference on Software Engineering (pp. 345-356).
28. Zhao, L., Yang, H., & Zhang, Q. (2023). Adoption barriers for AI development tools in educational contexts. Information and Software Technology, 158, 107189.
29. Zhou, M., Chen, G., & Davis, A. (2024). Hybrid intelligence in software engineering education: A framework for balanced integration. IEEE Intelligent Systems, 39(1), 34-42.