(Work in Progress) AI as a Cognitive Partner: Designing a Metacognitive Programming Workshop for Novice Learners

Dr. Sindia M. Rivera-Jiménez, University of Florida

Dr. Rivera-Jiménez is an Assistant Professor at the Department of Engineering Education (EED) and an affiliate faculty to the Department of Chemical Engineering at the University of Florida. Her research focuses on understanding the role of engineering communities while enacting their agency in participatory and transformational change. She is particularly interested in broadening the participation of minoritized communities by studying the role of professional development in shaping organizational cultures. As an education practitioner, she also looks at evidence-based practices to incorporate social responsibility skills and collaborative and inclusive teams into the curriculum. Dr. Rivera-Jiménez graduated from the University of Puerto Rico at Mayagüez with a B.S. and Ph.D. in Chemical Engineering. She earned an NSF RIEF award recognizing her effort in transitioning from a meaningful ten-year teaching faculty career into engineering education research. Before her current role, she taught STEM courses at diverse institutions such as HSI, community college, and R1 public university.

Dr. Lilianny Virgüez, University of Florida

Dr. Virgüez is an engineering educator who teaches foundational electrical engineering courses for students across multiple disciplines and class standings, with prior experience teaching first-year engineering students. Her work includes curriculum development and program-level coordination, and her research focuses on student motivation, workforce alignment, and the role of emerging technologies, including AI, in engineering education. She brings prior industry experience in the telecommunications sector and holds a Ph.D. in Engineering Education and a Master's degree in Management Systems Engineering from Virginia Tech, as well as a bachelor's degree in Telecommunications Engineering.

Mr. Diego Alvarado, University of Florida

Diego Alvarado is an Instructional Assistant Professor in the Department of Engineering Education at the University of Florida. He obtained his M.Eng. in Industrial Engineering from Texas A&M University and his B.S. in Industrial Engineering from Texas A&

AI as a Cognitive Partner: Designing a Metacognitive Programming Workshop for Novice Learners

Sindia M. Rivera-Jiménez, Lilianny Virguez, and Diego Alvarado
Department of Engineering Education, University of Florida, Gainesville, FL

Abstract

Introductory programming remains a persistent challenge for novice learners in engineering, particularly in two-year college and early undergraduate contexts. This paper presents Thinking About Thinking in Code, a 90-minute workshop introducing foundational programming logic through a physical computing challenge using Arduino, piloted across three engineering design course sections with 120 undergraduate participants. Grounded in self-regulated learning theory, the workshop positions AI tools (e.g., Claude) as cognitive partners that scaffold metacognitive regulation rather than generate solutions. Preliminary findings suggest that structured AI interaction protocols shifted students' perceptions of AI toward a reasoning support tool, with behavioral intentions to use AI for design tasks increasing by 23 percentage points. Contributions include a replicable workshop blueprint, a structured cognitive partner protocol, and a low-burden evaluation framework adaptable across early undergraduate engineering contexts.

1. Introduction

National organizations and workforce-focused reports have emphasized the growing importance of computational thinking, digital literacy, and programming-related problem-solving for engineering innovation and workforce readiness [1]. As a result, engineering students are increasingly expected to engage with programming logic early in their academic pathways, even when they do not enter college with prior coding experience. However, introductory programming remains a well-documented challenge for novice learners [2]. These challenges extend beyond learning syntax and include difficulties with planning solutions, monitoring program behavior, debugging errors, and evaluating alternative strategies, all of which require learners to regulate their thinking while solving problems. These barriers may be particularly salient for early undergraduate students in two-year college contexts, transfer pathways, and first-year programs, where learners often have diverse educational backgrounds, variable prior exposure to programming, and limited access to structured computing experiences outside of coursework. In these settings, low-stakes, hands-on learning environments can provide a valuable entry point for developing foundational coding skills and confidence [3], [4]. Yet, even within supportive environments, novice learners may struggle to make their reasoning visible, sustain productive debugging behaviors, and connect abstract programming logic to real-world engineering systems.

Generative artificial intelligence (AI) tools such as ChatGPT have recently emerged as accessible resources for students to support learning [5]. Because ChatGPT is widely available at no cost, it may be especially relevant in college settings where cost, access, and tool availability shape instructional choices. At the same time, generative AI introduces instructional risks for novice programmers when used primarily as an “answer engine,” potentially reducing sensemaking, increasing dependence, or encouraging shallow solution adoption without understanding. These tensions are real, but they also create a timely opportunity for educators to explore how AI can be repositioned. Rather than serving as a shortcut to code, AI can function as a cognitive partner that supports metacognitive regulation by helping learners plan, monitor, and evaluate their thinking during programming work.

To address this opportunity, this Work-in-Progress paper presents the design of Thinking About Thinking in Code, a 90-minute workshop that introduces foundational programming logic through a physical computing challenge using Arduino. The workshop integrates structured AI interaction protocols, external representations, and metacognitive regulation routines to support novice learners in two-year college and early undergraduate contexts.

2. Conceptual Framing for Workshop Design

This work is grounded in a practical instructional stance that treats novice programming difficulty as not only a challenge of learning syntax but also of learning to regulate cognition during problem-solving, particularly planning, monitoring, debugging, and evaluating. In computing education, metacognition and self-regulation have been identified as important contributors to programming success and are increasingly used to explain why beginners struggle and how instruction can better support them [6], [7]. Accordingly, the workshop design integrates (1) metacognitive regulation routines, (2) external representations that make reasoning visible, and (3) structured use of AI tools (e.g., ChatGPT) as a cognitive partner to scaffold reflection while discouraging over-reliance on AI-generated code. The workshop is operationalized through student-facing design materials that prompt learners to specify system behavior using thresholds, represent logic through flowcharts, and outline variables and if/else structures before implementing and testing code.

2.1 Metacognitive Programming. Programming tasks require learners to coordinate multiple cognitive processes, such as decomposing problems, anticipating program behavior, interpreting feedback, and iteratively refining solutions through debugging. For novice learners, these demands can be difficult because they are still developing mental models of how code executes and how small changes in logic produce different outcomes. Prior research highlights that metacognition and self-regulation are central themes in programming education research and reviews the theories and instructional exemplars that have been applied to support learners' regulation during programming tasks [8].

To frame these processes, this workshop draws on self-regulated learning (SRL) perspectives that conceptualize learning as cycles of forethought and planning, performance monitoring, and self-reflection [9]. These cycles align closely with the behaviors novice programmers must practice: articulating a plan, checking what the system is doing, and revising strategies based on evidence. In this workshop, metacognitive regulation is made explicit through structured activities that require learners to (a) define desired behavior in measurable terms, (b) represent decision logic before coding, and (c) use debugging evidence (e.g., serial monitor output) to evaluate whether the program behaves as intended.

2.2 ChatGPT as a Cognitive Partner with Guardrails. Generative AI tools are rapidly becoming part of students' learning ecosystems, and their accessibility (e.g., ChatGPT, CoPilot) makes them especially relevant in early undergraduate contexts. Recent research in programming instruction suggests that GenAI-based supports can shape novice learners' outcomes, including programming performance and cognitive load, particularly when used in structured ways aligned with SRL-focused instruction [10]. However, a core tension in GenAI-supported learning is that these tools can either deepen reasoning or reduce it, depending on how learners engage with them [11]. If students treat generative AI primarily as an "answer engine," they may skip productive struggle, adopt code without understanding it, or disengage from debugging practices essential to learning programming logic. Thus, this workshop adopts a design stance of AI as a cognitive partner, where

AI supports metacognitive dialogue by prompting learners to explain decisions, compare alternatives, and interpret evidence, rather than producing complete solutions.

To enact this stance, the workshop incorporates guardrails that maintain student ownership of reasoning: "explain-first" routines (students articulate goals and predicted behavior before seeking AI support), verification steps (students test outputs against predictions), and reflection prompts that focus on planning–monitoring–evaluation cycles. This design choice is aligned with SRL frameworks that emphasize intentional monitoring and reflection as mechanisms for improving learning outcomes.

2.3 Physical Computing and External Representations to Support Knowledge Transfer. Physical computing environments can be particularly useful for beginners because they create rapid, observable feedback loops: changes in logic produce visible changes in system behavior. In the workshop design, Arduino activities allow students to connect abstract programming structures (variables and conditionals) to physical cause-and-effect relationships (sensor readings triggering LED/buzzer outputs). To further support knowledge transfer and reduce cognitive overload, the workshop also uses external representations, specifically a flowchart, to help novices organize reasoning before implementation. Students document decision thresholds (e.g., distance cutoffs) and map those thresholds to system responses, then translate the logic into action. This approach aligns with metacognitive instructional strategies that emphasize making thinking visible and supporting learners in monitoring and evaluating their own decisions during problem solving [6], [8]. These design elements ensure the workshop targets both foundational programming knowledge and process skills that support persistence and adaptability in novice coding, particularly during debugging and iteration.

3. Workshop Design

3.1 Workshop Overview and Intended Audience. *Thinking About Thinking in Code* is a 90-minute interactive workshop intended for early undergraduate students, including community college learners and first-year students, who have little or no prior programming experience. The workshop functions as a low-stakes, high-engagement entry point to programming logic and debugging practices through a collaborative, hands-on activity. Students were allowed to use any AI tool (e.g., Claude, ChatGPT), as a cognitive partner to support metacognitive regulation through structured reflection prompts rather than as a tool for generating complete solutions. The workshop positions generative AI as a cognitive partner that supports metacognitive regulation through structured reflection prompts, rather than as a tool for generating complete solutions. This design choice aims to preserve student ownership of reasoning while providing accessible support for planning, monitoring, and evaluating programming decisions during the activity.

3.2 Learning Outcomes. By the end of the 90-minute workshop, students will demonstrate technical learning outcomes by defining an engineering design goal for a real-world sensing scenario and describing expected system behavior using measurable distance thresholds. Students will also identify core inputs and outputs in an Arduino-based system, document basic component connections including an ultrasonic sensor, LED, and buzzer, translate threshold-based behaviors into foundational programming logic using variables and conditional if/else structures, and represent program logic using a flowchart to anticipate system behavior prior to coding. In addition to these technical outcomes, students will demonstrate affective and metacognitive outcomes by engaging in metacognitive regulation processes, including articulating predictions, interpreting debugging evidence (e.g., serial monitor outputs), and revising logic or thresholds based on observed outcomes during iteration.

3.3 Core Design Task and Student Packet. The workshop is structured around a scenario-based design prompt that requires students to build a "proximity alert" system for a delivery worker using an ultrasonic sensor, LED indicator, and buzzer. Students complete a structured packet with worksheets that guide them through iterative design reasoning from problem definition to logic specification and implementation**.**

Worksheet 1: Mini Challenge: Design for a Real Person. Worksheet 1 scaffolds students' reasoning from the problem definition to a structured implementation plan. In Step 1, students define the intended user and articulate the design goal by specifying what the device should do. In Step 2, they identify the required system components and map key inputs and outputs, including documenting Arduino pin connections. In Step 3, students specify system behavior using distance-based thresholds (e.g., far/medium/close) and assign corresponding alert responses such as LED and buzzer states. In Step 4, they translate these rules into a flowchart that makes decision points and actions visible before any coding begins. In Step 5, students prepare for implementation by identifying key variables (e.g., distance) and outlining the conditional logic structure (if/else) needed to execute the designed behavior.

Worksheet 2: Build Instructions and Annotated Code Example. Worksheet 2 provides implementation scaffolding, including materials, wiring guidance, and an annotated starter code example. The packet explicitly encourages students to use serial monitor outputs to interpret sensor readings and validate whether threshold logic produces expected behavior. The packet also includes troubleshooting prompts (e.g., checking wiring, verifying pin numbers) and optional extension challenges for students who complete the core task early.

3.4 Workshop Facilitation Flow and Timing. The workshop is facilitated using a structured sequence that emphasizes logic before code and prediction before debugging. Students begin by reviewing workshop norms and the role of AI tools (e.g., Claude, ChatGPT) as cognitive partners, then define the design goal and user need that will guide the system's behavior. Next, students identify the required components and map inputs and outputs to prepare for implementation. The workshop then moves into logic development, where students construct a threshold-based behavior plan and translate this plan into a flowchart to externalize decision-making before writing code. After establishing the logic model, students outline key variables and conditional if/else structures, then build and test the system using Arduino while collecting evidence through observed outputs and serial monitor readings. The session concludes with a short reflection and exit ticket to capture student sensemaking, strategy use, and areas for improvement.

3.5 ChatGPT as a Metacognitive Scaffold (Proposed Implementation Protocol). To align generative AI use with the workshop learning outcomes, AI tools (e.g., Claude, ChatGPT) are positioned as resources for structured reflection and reasoning support rather than direct code generation. Students follow a brief protocol designed to promote metacognitive regulation during debugging by encouraging them to articulate predictions, interpret evidence through serial output and observed system behavior, and test revisions iteratively. Table 1 summarizes the protocol and example prompts used to support this AI as a cognitive partner approach. This protocol is intended to preserve students' ownership of problem-solving while using AI tools to scaffold planning, monitoring, and evaluation behaviors central to effective debugging and iterative design.

Table 1. ChatGPT "Cognitive Partner" Protocol (Metacognitive Scaffold)

	Student action (before/with ChatGPT)	***Example prompt***
Explain-first	State expected behavior, observed behavior, and evidence (serial output + physical response).	"Don't give me the answer yet—ask me questions to help me debug based on what I expected vs what happened."
Ask for questions, not answers	Request guidance and reasoning support, not full code generation.	"Help me interpret my serial monitor values and suggest what to check next."
Verify + reflect	Test one change, document what changed, and identify the next step.	"Given this new output, what is one small change I should test next and why?"

4. Planned Pilot and Evaluation Plan

4.1 Pilot Implementation Context. The workshop was piloted across three sections of an engineering design course, two at the University of Florida and one at Santa Fe College, where an articulated program supports engineering transfer students. A total of 116 undergraduate students participated across the three sections. Participants included early undergraduate students from multiple engineering majors and academic levels, with the majority being first-year students. The workshop was implemented as a single 90-minute session embedded within existing course activities, allowing for consistent facilitation across sections while maintaining the core sequence of defining system behavior, specifying threshold logic, modeling logic through a flowchart, and implementing and debugging using evidence. AI tools (e.g., Claude) were made available to all students during the session, with the structured cognitive partner protocol used consistently across sections.

4.2 Participants. Participants were undergraduate engineering students enrolled in engineering design courses at two institutions. The majority were first-year students, reflecting the workshop's intended audience of early undergraduate learners with limited prior programming experience. The sample also included students from Santa Fe College participating through an articulated transfer pathway, providing diversity in academic background and institutional context. Prior to the workshop, most students reported moderate to high confidence in using AI for academic work, yet 67% had received no formal AI training. Nearly all participants had already used AI tools independently, suggesting that students are developing AI practices informally and without structured guidance, creating a gap between using AI and using it effectively to support reasoning and design work.

4.3 Data Sources and Evidence of Learning. For this WiP, data collection included a pre- and post-survey measuring behavioral intentions to use AI for engineering design tasks, a filled workshop handout capturing students' logic planning and flowchart development, open-ended survey questions, and evidence of the AI interactions students engaged in during the activity. The pre- and post-surveys included items from three constructs: attitudes toward using AI to support design thinking, perceived usefulness of AI for design decision-making, and behavioral intentions to use AI for future design tasks. Facilitator observation notes documented pacing, recurring challenges, and patterns of group interaction across sections. This study was reviewed and approved by the authors' institutional review board. Participation was voluntary and not contingent on providing research data. Student artifacts and survey responses were de-identified for analysis and reporting. Informed consent was obtained from all participants prior to data collection.

4.4 Preliminary Findings. Preliminary results are descriptive and not causal, reflecting the exploratory nature of this pilot. They are presented here to illustrate the promise of the workshop design and inform future iterations.

Prior to the workshop, approximately half of the students already reported positive attitudes toward using AI in design tasks, indicating that students were not starting from zero. Following the structured prompting intervention, positive responses increased across all three survey constructs. Attitudes toward using AI as a cognitive partner increased from 50% to 67%. Perceived usefulness for design decision-making increased from 51% to 69%. The most notable shift was in behavioral intentions, which increased from 48% to 71%, a 23-percentage-point gain. These shifts suggest that structured prompting may be helping reposition AI from an answer-generating tool toward a resource that supports reasoning, decision-making, and problem-solving in engineering design tasks.

Open-ended responses revealed that students consistently framed AI as something that supports thinking rather than something that replaces engineers' judgment or creativity. Across responses, students clustered their descriptions of AI's role around three practical functions: speeding up work through automation, coding assistance, and data processing; generating ideas through brainstorming and exploration; and improving work by finding flaws and checking logic. Importantly, students were not naïve about AI's limitations. Many explicitly mentioned the importance of verification, ethical considerations, and the risks of overreliance, and consistently affirmed that engineers remain accountable for decisions and outcomes. This suggests that the structured cognitive partner protocol may be supporting more intentional and critically aware AI use among novice learners.

5. Contributions and Next Steps

This paper presents a replicable workshop design that addresses a persistent challenge in engineering education: helping novice learners develop foundational programming logic and debugging skills in two-year college and early undergraduate contexts. The workshop offers three contributions to the engineering education community. First, a replicable 90-minute blueprint emphasizing logic before code through system behavior planning, threshold rules, and flowcharting, adaptable across engineering clubs, bridge programs, and introductory labs. Second, a structured approach for using AI tools (e.g., Claude) as a cognitive partner, including explain-first and verification routines that scaffold metacognitive reflection during debugging. Third, a low-burden evaluation framework combining student artifacts, pre- and post-survey data, and open-ended responses to generate early evidence of students' logic representation and conditional reasoning. Next steps include refining the workshop based on preliminary findings from the three pilot sections, with attention to pacing, scaffolding, and the cognitive partner protocol. Future iterations will explore additional practice cycles and alternative AI-supported environments, with the long-term goal of developing scalable resources that support early undergraduate learners in building transferable programming reasoning skills for engineering design.

Acknowledgments: The authors gratefully acknowledge the undergraduate engineering students who participated in the workshop sessions and Dr. Andrea Goncher for graciously allowing the research team to implement the workshop in two of her engineering design course sections.

Authors' Contribution: *Conceptualization:* Rivera-Jiménez, Virguez, and Alvarado. *Methodology:* Rivera-Jiménez, Virguez, and Alvarado. *Investigation:* Rivera-Jiménez and Virguez. *Formal analysis:* Rivera-Jiménez and Virguez. *Writing, original draft:* Rivera-Jiménez. *Writing, review and editing:* Rivera-Jiménez, Virguez, and Alvarado.

References:

[1] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006, doi: 10.1145/1118178.1118215.

[2] G. Zhu, J. F. Kow, X. Fan, I. H. Yeter, L. S. Chit, and Y. S. Ong, "Exploring Undergraduate Students' Computational Thinking Skills Across Engineering Design Processes," *Computer Applications in Engineering Education*, vol. 33, no. 3, p. e70035, 2025, doi: 10.1002/cae.70035.

[3] N. A. Bowman, L. Jarratt, K. C. Culver, and A. M. Segre, "How Prior Programming Experience Affects Students' Pair Programming Experiences and Outcomes," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, in ITiCSE '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 170–175. doi: 10.1145/3304221.3319781.

[4] M. Butler and M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts," in *Proceedings ascilite Singapore*, 2007.

[5] A. Cortez and P. Schmelzenbach, "BOARD# 78: Student Use of ChatGPT and Claude in Introductory Engineering Education: Insights into Metacognition and Problem-Solving Patterns," in *2025 ASEE Annual Conference & Exposition*, 2025.

[6] R. M. Marra, D. J. Hacker, and C. Plumb, "Metacognition and the development of self-directed learning in a problem-based engineering curriculum," *Journal of Engineering Education*, vol. 111, no. 1, pp. 137–161, 2022, doi: 10.1002/jee.20437.

[7] M. P.-C. Lin, A. L. Liu, S. Saffari, D. Chang, and J. Ryoo, "Mapping AI Tools in Education: A Topic Modeling Analysis of Cognitive, Metacognitive, and Affective Insights," in *Generative Systems and Intelligent Tutoring Systems: 21st International Conference, ITS 2025, Alexandroupolis, Greece, June 2–6, 2025, Proceedings, Part I*, Berlin, Heidelberg: Springer-Verlag, Jul. 2025, pp. 88–101. doi: 10.1007/978-3-031-98281-1_7.

[8] D. Loksa *et al.*, "Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use," *ACM Trans. Comput. Educ.*, vol. 22, no. 4, p. 39:1-39:31, Sep. 2022, doi: 10.1145/3487050.

[9] E. Panadero, "A Review of Self-regulated Learning: Six Models and Four Directions for Research," *Front. Psychol.*, vol. 8, Apr. 2017, doi: 10.3389/fpsyg.2017.00422.

[10] A. Y. Q. Huang, C.-Y. Lin, S.-Y. Su, and S. J. H. Yang, "The impact of GenAI-enabled coding hints on students' programming performance and cognitive load in an SRL-based Python course," *British Journal of Educational Technology*, vol. 56, no. 5, pp. 1942–1972, 2025, doi: 10.1111/bjet.13589.

[11] D. Cambaz and X. Zhang, "Use of AI-driven Code Generation Models in Teaching and Learning Programming: a Systematic Literature Review," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, in SIGCSE 2024. New York, NY, USA: Association for Computing Machinery, Mar. 2024, pp. 172–178. doi: 10.1145/3626252.3630958.