2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
Paper ID #51846

AI-assisted Arduino Coding in First-Year Engineering Team Projects

Dr. Esther Tian, University of Virginia

Esther Tian is an Associate Professor of Engineering in the School of Engineering and Applied Science at the University of Virginia. She received her Ph.D. in Mechanical Engineering from the University of Virginia. Her research interests include engineering design education and generative AI integration.

Alexander Mason Boback

AI-assisted Arduino Coding in First-Year Engineering Team Projects

Abstract

This Complete Evidence-based Practice paper examines how first-year engineering students engage with generative artificial intelligence (AI) tools to support Arduino coding in team-based design projects. Through pre-post surveys and reflection data, we explore students' AI usage patterns, learning experiences, ethical considerations, and team dynamics. Results indicate that most students used AI as a learning scaffold for explanations and debugging, demonstrating shepherding behaviors rather than passive acceptance of complete solutions. However, while most reported understanding their code, only 19% of students expressed confidence in reproducing similar work independently, suggesting dependency rather than skill transfer. Students generally demonstrated ethical awareness in their AI use, though practice gaps emerged. Team dynamics were predominantly positive, though limited communication about AI use revealed implicit rather than coordinated integration. These findings underscore the need for structured pedagogical scaffolding that balances AI support with learning outcomes. The study contributes evidence-based recommendations for integrating generative AI into first-year engineering curricula in ways that enhance rather than undermine student learning, promote ethical awareness, and support effective team collaboration.

Introduction

As AI tools such as GitHub Copilot and ChatGPT become increasingly integrated into programming environments, engineering educators face the dual challenge of preparing students for AI-enabled professional practice while ensuring foundational skill development. Arduino-based projects provide a tangible platform for this learning, bridging digital programming with physical computing through team collaboration and rapid prototyping. Yet as AI tools generate code, explain syntax, and troubleshoot hardware integration, questions arise about how interactions between students and AI tools reshape students' learning trajectories, teamwork processes, and development of programming confidence and competence.

Theoretical Frameworks

This research draws on two interconnected theoretical frameworks to examine how generative AI tools reshape learning and collaboration in team-based Arduino programming projects.

Learning occurs most effectively when appropriate scaffolding bridges the gap between what learners can accomplish independently and what they can achieve with support. In technology-enhanced learning environments, digital tools can provide scaffolding that complements or extends human support [1]. Educational contexts require determining when learners need scaffolding that enables independent performance versus tools that permanently augment their capabilities [1]. Meta-analyses of computer-based scaffolding in STEM education demonstrate moderate positive effects, with consistent outcomes across diverse contexts and designs [2]. The fundamental goal of educational scaffolding is developing learners' ability to perform independently. However, AI code generators challenge this principle as they can complete entire programming tasks independently, potentially bypassing the cognitive engagement necessary for

skill development. For AI tools in programming education, this raises a fundamental question: Do these tools provide appropriate scaffolding that supports learning, or do they enable learners to bypass necessary developmental struggles, creating a perception of competence without actual learning?

In team-based engineering projects, cognition is distributed across team members, physical artifacts, and representational tools [3], [4]. This perspective shifts the unit of analysis from individual learners to the cognitive system, the ensemble of people and tools working together to accomplish goals. Computer-supported collaborative learning extends this framework, emphasizing how digital tools mediate group knowledge construction and reshape how teams organize tasks, communicate ideas, and construct shared understanding [5]. When AI code generators enter this distributed cognitive system, they potentially reconfigure fundamental aspects of teamwork: how tasks are distributed, how knowledge and expertise are shared, what roles emerge or disappear, and how teams engage in collective reasoning. If AI tools fundamentally alter team dynamics, for example, by enabling one member to complete all coding tasks independently or by reducing the need for collaborative troubleshooting, the pedagogical benefits of team-based learning may be compromised, undermining development of communication, negotiation, and integrative problem-solving skills essential to professional practice.

Together, these frameworks provide complementary lenses for examining AI-assisted learning in collaborative engineering contexts: how AI functions as scaffolding for individual skill development and how AI tools reconfigure team-based cognitive systems.

Literature Review

Recent research on AI-assisted programming reveals important patterns about tool use, learning outcomes, and pedagogical design, while highlighting critical gaps, particularly regarding team-based learning in physical computing contexts.

Research on programmers using AI code generators identified two primary modes: acceleration, where users employed tools like GitHub Copilot to speed up routine tasks they already understood, and exploration, where they used AI to discover implementation options when uncertain how to proceed [6]. Examining novice programmers specifically, Prather et al. documented two interaction patterns: shepherding, where spent excessive time iteratively guiding AI to generate code rather than coding independently, and drifting, where students passively accepted incorrect suggestions and became progressively lost [7]. Both patterns represent forms of dependency that undermine learning, underscoring the critical pedagogical challenge of designing instruction that promotes productive AI engagement, using AI as a learning tool that builds autonomous capability while preventing dependence.

While AI tools can enhance productivity, their impact on actual learning remains contested. Johnson et al. compared novice students who wrote their own Arduino programs, the same platform used in our course, versus those using ChatGPT [8]. Students who coded independently developed significantly greater self-efficacy and programming ability, while the ChatGPT group demonstrated lower programming knowledge and confidence in their coding abilities, suggesting

they achieved task performance without developing equivalent understanding [8]. Zviel-Girshin's mixed-methods study of novice programmers identified both benefits and risks of AI tool use [9]. On the positive side, AI tools can broaden access to programming support, provide immediate feedback, and help students overcome initial barriers to entry. However, significant concerns persist regarding academic integrity, over-reliance that inhibits skill development, reduction in problem-solving practice, and the potential for students to submit AI-generated work without understanding it [9]. These concerns are amplified in introductory courses where students are simultaneously developing foundational knowledge, technical skills, and professional practices.

Despite growing interest, several critical gaps remain. First, most research on AI-assisted programming focuses on individual learners in text-based programming environments rather than on teams engaged in open-ended, physical computing projects [6], [7]. While Johnson et al. examined Arduino programming, their study focused on individual learners rather than collaborative teams [8]. From a distributed cognition perspective [3], [4], this gap is significant: when AI tools enter team-based cognitive systems, they likely reconfigure how teams coordinate work, communicate ideas, distribute tasks, and construct shared knowledge, yet these dynamics remain largely unexamined.

Second, few empirical studies have systematically examined structured pedagogical scaffolds that guide responsible AI use in practice. While researchers identify the need for explicit instruction on appropriate AI use, documentation of what such instruction looks like in project-based learning contexts and evidence of its effectiveness remains limited [7], [9]. Building on scaffolding theory [1], [2] and addressing the needs identified in current research, effective AI integration requires explicit instructional support that includes: (1) guidance on when AI use is appropriate versus when independent problem-solving is more valuable, (2) instruction in effective prompting and iterative refinement, (3) strategies for critically evaluating AI outputs, (4) ethical frameworks for attribution and academic integrity, and (5) structured reflection on learning processes and outcomes.

Third, the ethical dimensions of AI use in educational contexts deserve more explicit attention. The ethical landscape extends beyond plagiarism to include attribution, transparency about AI contributions, fairness in collaborative contexts where team members may use AI differently, and the long-term implications of AI dependency for professional development.

This study addresses these gaps by examining how first-year engineering students use generative AI for Arduino coding in team-based projects through three research questions:

1. How do students leverage generative AI for Arduino coding in their team-based projects?
2. How do students consider ethical implications of AI use?
3. How does AI use affect team dynamics?

Approaches and Methods

This study was conducted at the University of Virginia in Engineering Foundations I, a required first-semester course for all first-year engineering students taken during the fall semester. In the

first half of the term (approximately 5 weeks), students undertake team-based Arduino projects that combine microcontroller programming, circuit design, and mechanical/3D design to create functional prototypes [10]. Students enter the course with highly variable programming backgrounds: approximately 15% have extensive programming experience, 61% have some programming experience, and 24% have no programming experience. However, Arduino-specific experience was far more limited, with 61% of all students having never used Arduino IDE prior to the course. This heterogeneity in technical preparedness combined with the authentic engineering challenges of hardware-software integration makes the Arduino project ideal for examining how AI tools can support diverse learners in collaborative settings.

Research Design and participants

This study employed a mixed-method, pre-post design to examine students' AI tool usage patterns, learning experiences, ethical considerations, and team dynamics in the context of team-based Arduino projects. We collected both quantitative survey data and qualitative open-ended responses.

Participants included first-year engineering students who consented to participate in the research study in Fall 2024 and Fall 2025. Demographic data were not collected. (The study was approved by the university's Institutional Review Board for Social and Behavioral Sciences, Protocol Numbers: 6614 and 6881.)

Students worked in teams of four to five members formed through a combination of random assignment and skill diversity considerations. To ensure individual learning accountability despite collaborative work and potential AI use, each team member was required to demonstrate mastery of at least one sensory component (e.g., ultrasonic sensor, photoresistor) or actuator component (e.g., servo motor, buzzer), regardless of whether that component was incorporated into the team's final prototype. Each student maintained and submitted a study journal documenting source code for their chosen component, including AI-generated code when used, along with required modifications demonstrating understanding. Team-level code iterations were recorded in the team's engineering notebook. These dual documentation requirements served both as evidence of individual learning accountability and as a mechanism to track appropriate AI use.

Data Collection

Baseline Survey: Captured students' pre-college AI experience, including familiarity with AI tools, frequency and purposes of use, perceived reliability, confidence levels, and expected advantages and limitations.

Post-Project Survey: Administered following Arduino project completion. Students reported on AI tool usage during the project (which tools, how often, for what purposes), learning outcomes (code comprehension, ability to explain AI-generated code, confidence in debugging, and independent reproduction), ethical practices (verification of outputs, documentation of AI use, consideration of ethical appropriateness), team dynamics (communication about AI use,

perceived impacts on productivity, shared understanding, and fairness), and open-ended reflections on benefits, challenges, and suggestions.

Pedagogical Approach

AI tool integration evolved across the two-year study period based on initial findings. In Fall 2024, AI was introduced with basic guidance covering AI tool overview, prompting strategies, citation requirements, and a simple Arduino demonstration (blinking LED). Student reflections revealed struggles with ethical ambiguity and uncertainty about when AI use was appropriate, which directly informed the development of more explicit and structured scaffolding in subsequent iterations.

In Fall 2025, grounded in scaffolding theory [1], [2] and distributed cognition frameworks [3], [4], we developed a comprehensive AI-Assisted Arduino Coding Guide to support responsible and effective AI use. The guide provided explicit instruction on: (1) effective prompting strategies, (2) when and how to use AI in Arduino coding without compromising learning, (3) proper citation and attribution of AI tools, (4) ethical considerations including academic integrity, learning trade-offs, transparency, bias and reliability, dependence, and sustainability, and (5) implications for team collaboration. The guide was implemented through in-class sessions with interactive case studies on AI ethics to promote discussion and reflection, supplemented by ongoing support.

Results

Baseline survey data revealed that nearly all students (93%) had prior experience with AI tools before entering college. Regarding frequency of use, students most commonly used AI tools monthly (36%), followed by weekly (25%), or several times a week (15%). However, only about one-third (34%) had used AI specifically for coding or debugging purposes. When asked about their confidence using AI tools, 94% reported moderate to high confidence levels. Students recognized AI's potential to support their learning but also expressed concerns about output inaccuracy and the risk of overreliance.

Students entered the course with highly variable programming backgrounds. Among the 59 participants, 9 (15%) reported extensive programming experience, 36 (61%) reported some programming experience, and 14 (24%) reported no programming experience. Importantly, even among those with programming experience, many lacked Arduino-specific knowledge: 4 of the 9 "extensive" programmers and 18 of the 36 "some experience" programmers had never used Arduino IDE. Overall, 36 students (61%) had no prior Arduino coding experience, creating substantial heterogeneity in the technical skills students brought to their team projects.

To support all students, particularly those with limited programming or Arduino experience, while promoting responsible AI use, we developed and implemented an AI-Assisted Arduino Coding Guide. This guide provided explicit scaffolding on when and how to use AI effectively without compromising learning, along with ethical considerations and best practices. Following implementation, we examined how students leveraged AI tools, their ethical awareness, and their perceptions of AI's influence on team dynamics.

Learning experiences with AI-assisted Arduino coding

Survey responses indicate that AI use was common during the Arduino project. Of 59 respondents, 36 (61%) reported using AI to support their coding, while 23 students (39%) completed the project without AI assistance. Among AI users, ChatGPT was the predominant tool, employed by 30 students (83% of AI users).

When asked about frequency of use, the majority of AI users (28 of 36, 77%) reported using AI tools either "frequently" or "a few times" throughout the project, while the remaining 8 students (22%) reported using AI "once or twice." Notably, no students reported using AI for their entire coding process, suggesting that AI served as supplemental support rather than a complete replacement for independent programming. Comparing these patterns to baseline data collected at the beginning of the academic year reveals increased AI engagement during the Arduino project. At the start of the year, only 5 students (8%) reported using AI tools at least daily for any purpose. During the Arduino project, however, 17 students (29%) described their use as "frequent," indicating heightened adoption of AI tools in the context of a challenging, authentic engineering task.

Reported use patterns suggest that students leveraged AI primarily as a learning and troubleshooting aid rather than a shortcut for generating complete solutions. Among the 36 AI users, 25 students (69%) reported using AI for code explanations or syntax help, 20 students (56%) used it for debugging, and 12 students (33%) used it to generate small code snippets (Figure 1). In contrast, only 8 students (22%) reported using AI for one-shot generation of complete programs. These patterns suggest students actively engaged with AI for specific learning needs rather than passively accepting complete solutions.

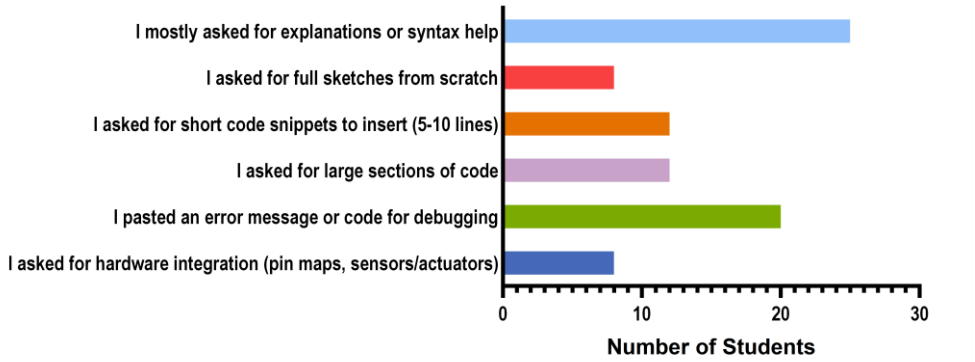

Figure 1. AI tools used for different tasks

Qualitative data from open-ended survey responses corroborated these quantitative patterns. Some of the most frequent themes were concept and code explanation (n = 24), debugging assistance (n = 13), and learning Arduino's syntax and structure (n = 6). Given that 61% of students had no prior Arduino IDE experience, many described using AI to bridge knowledge gaps. Representative quotes include:

“AI use supported my learning because I do not have experience programming at all so it allowed me to understand the code examples in the lecture slides by carefully explaining them and modifying, so then I was able to modify myself.”

“Arduino IDE is a coding language that I did not know prior to joining this class, so the use of generative AI helped me make sense of some syntax or functions that I was not aware of.”

“Without AI, I wouldn’t have been able to figure out how to integrate my prior programming knowledge into code for an Arduino.”

These responses suggest that AI served as an accessible learning resource, particularly valuable for novices navigating an unfamiliar programming environment.

Students’ perceptions of their learning suggest that AI use generally supported code comprehension and skill development. Among the 36 AI users, 30 (83%) reported understanding most or all lines of code in their final project, though slightly more than one-third (36%) reported understanding every line (Figure 2).

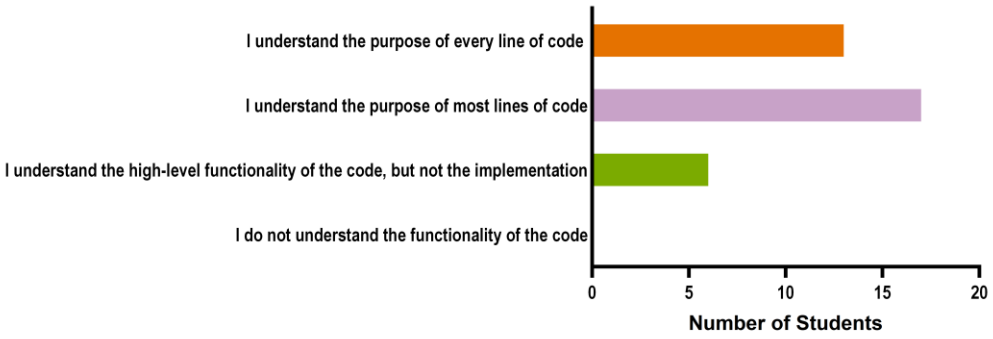

Figure 2. The degree students understand the code they wrote for their projects

Similarly, 28 students (78%) indicated they could articulate the purpose of key variables and functions in AI-generated code (Figure 3), and 25 students (69%) reported that AI contributed to stronger overall understanding of their project code (Figure 4). Beyond code comprehension, students perceived benefits in deeper learning and debugging skills. Twenty-eight students (78%) indicated that AI helped them learn why code works, and 29 students (81%) reported increased confidence in debugging abilities (Figure 3). These patterns suggest that when used appropriately, AI tools may function as effective scaffolds for developing programming competence.

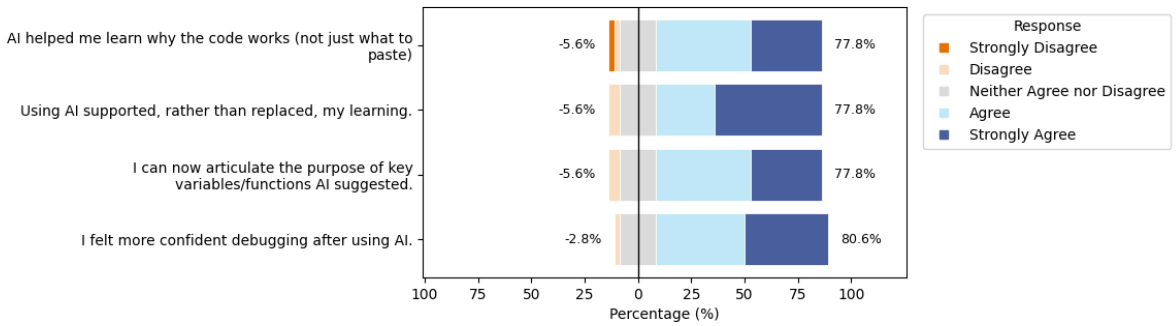

Figure 3. Student-reported learning outcome and confidence from AI use

Students acknowledged practical efficiency gains from AI use: 29 (81%) reported that AI saved time during the project (Figure 4). However, how students leveraged this efficiency varied considerably. While 22 students (61%) indicated that AI enabled them to achieve more complex and ambitious code, and 13 students (36%) reported redirecting saved time toward other project components they found intellectually engaging, such as mechanical design innovation, circuit optimization, or creative problem-solving.

This pattern reveals an important distinction: AI tools appeared to support vertical depth within the coding domain (more sophisticated programming) more than horizontal breadth across project domains (deeper engagement with non-coding components). This raises questions about how AI tools function within the distributed cognitive system of team-based projects [3], [4]: rather than redistributing cognitive effort across team members and project domains, AI may concentrate expertise in the coding domain for those who use it.

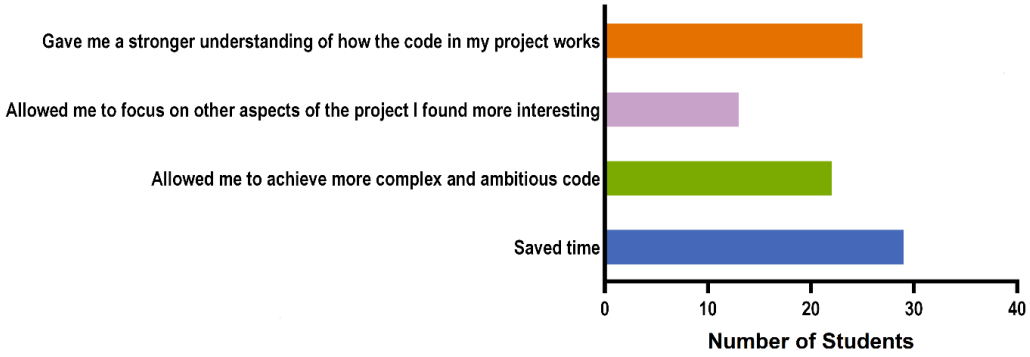

Figure 4. Ways students view the impact of using AI to help with writing code for their projects

Despite reporting positive learning experiences, AI users demonstrated limited confidence in their ability to reproduce similar work independently, suggesting limited genuine skill development. Only 7 of 36 AI users (19%) reported they could recreate their project code without external resources, while 29 students (81%) indicated they would need assistance from AI tools, documentation, or other resources to reproduce similar Arduino projects (Figure 5). This finding suggests a potential gap between perceived learning and actual transfer of skills, aligning with Johnson et al.'s [8] finding that ChatGPT users achieved task performance but demonstrated lower programming ability and self-efficacy than students who coded independently. Pea [1] distinguishes scaffolding, temporary support that fades as learners

develop autonomous competence, from distributed intelligence, where tools permanently enable performance. The finding that 81% of students remained dependent on external resources suggests AI functioned more as distributed intelligence than as fading scaffolding, enabling task completion without building the independent capability that educational scaffolding aims to cultivate.

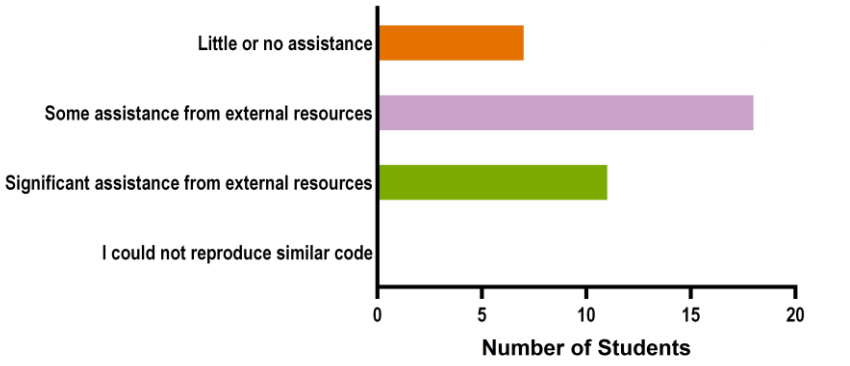

Figure 5. Students' confidence level in writing a program similar to the one created for their projects.

Interestingly, students themselves rarely identified AI as problematic for their learning. Among open-ended survey responses, only 6 students explicitly described AI as hindering active learning or reducing their engagement with problem-solving. The disconnect between low confidence in independent reproduction and few explicit concerns about AI's learning impact may reflect several factors. First, students may not recognize dependency until faced with transfer tasks requiring independent application. Second, the limited scope of the Arduino project (one program over five weeks) may have been insufficient for students to develop or assess independent capability. Third, students may value task completion and immediate understanding over longer-term skill development, particularly in the context of time-pressured coursework.

These findings highlight a critical pedagogical challenge: AI tools can simultaneously support immediate learning (as students reported) while undermining development of independent capability (as reproduction confidence suggests). This tension underscores the importance of explicit instructional scaffolding that not only guides how to use AI but also ensures students practice independent problem-solving and build toward self-sufficiency.

While 36 students (61%) used AI tools during the Arduino project, 23 students (39%) completed the project without AI assistance. Understanding non-users' reasoning and experiences provides important context for interpreting AI users' outcomes by illustrating that AI was not the only viable form of support. Students who did not use AI drew on alternative resources such as online tutorials, documentation, and peer/instructor support, achieving successful task completion through different pathways.

Among the 23 non-users, motivations for avoiding AI varied: 10 students (43%) reported either wanting to learn independently or lacking time to learn how to use AI effectively, 8 students (35%) indicated they did not believe AI would be helpful for Arduino programming or doubted its reliability, and 5 students (22%) cited ethical concerns.

When asked what resources they used instead, non-users predominantly relied on online tutorials and documentation. Among 23 non-users, 15 (65%) reported using online resources such as YouTube, while 8 (35%) used official Arduino documentation. The effectiveness of any learning resource, whether AI or traditional, ultimately depends on how students engage with it: as a scaffold that supports building understanding through active cognitive processing, or as a shortcut that enables task completion without fostering independent capability.

Ethical awareness

Students demonstrated considerable attention to ethical dimensions of AI use, though with notable gaps between awareness and consistent practice. Half of AI users (18 of 36, 50%) identified incorrect or unreliable outputs as a primary concern in open-ended responses, with representative quotes including: "It had some programming errors that I had to correct" and "Sometimes generative AI is just plain wrong or does not understand what question I am asking."

Consistent with this awareness of AI limitations, 26 of 36 AI users (72%) reported verifying AI outputs before incorporating them into their projects, and 23 (64%) documented their AI use in code comments or reports (Figure 6). When directly asked whether they considered the ethics of their AI use, 32 (89%) indicated they had reflected on whether their AI use was ethically appropriate in the project context, while 4 (11%) gave neutral responses.

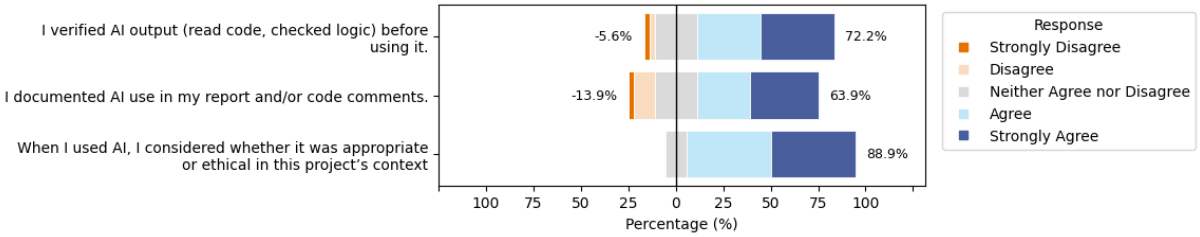

Figure 6. Ethical consideration of AI use in the project's context

While these rates represent meaningful engagement with responsible use practices, gaps remain: 28% did not verify outputs, and 36% did not document AI use despite explicit course guidance on attribution. These awareness-practice gaps suggest that understanding ethical principles does not automatically translate to consistent behavior. These patterns suggest that while the AI-Assisted Arduino Coding Guide successfully raised ethical awareness, translating awareness into consistent practice requires ongoing reinforcement.

Ethical considerations also influenced decisions not to use AI. Among the 23 non-users, 5 (22%) explicitly cited ethical concerns as reasons for avoiding AI tools. These diverse responses from those who used AI with attribution to those who avoided it entirely reveal a spectrum of ethical reasoning about AI's appropriate role in learning.

Effect of AI use on Teamwork

Students reported predominantly positive perceptions of AI's impact on team collaboration, though with limited explicit coordination about AI use. Fewer than half of AI users (14 of 36,

39%) discussed their AI use with teammates, and no teams established explicit agreements about AI use (Figure 7). This limited coordination suggests students treated AI use as an individual choice rather than a team-level decision with potential collective implications.

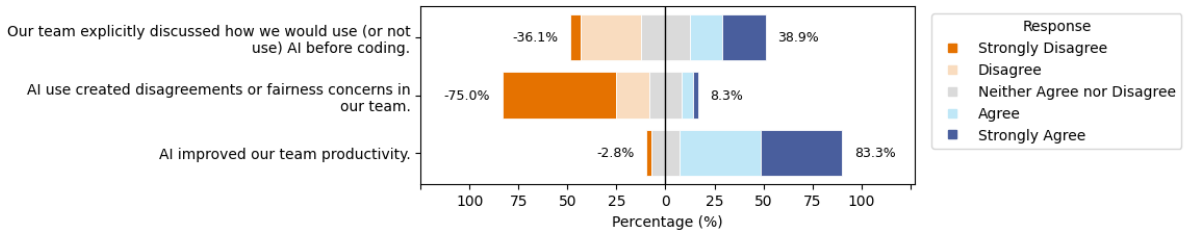

Figure 7. AI use and its effect on teamwork

Despite this lack of explicit coordination, students overwhelmingly reported that AI enhanced team productivity, with 30 of 36 AI users (83%) indicating AI improved their team's efficiency (Figure 7). Open-ended responses revealed two primary mechanisms through which AI supported teamwork:

Enhanced shared understanding (n = 16): Students described how AI helped bridge varying expertise levels within teams. Representative quotes include: "It acted as a neutral explainer when teammates had different coding backgrounds, helping us align on terminology and logic" and "AI was able to help share how others coded their projects, where it gave a better description and allowed others to understand and share what they learned." These responses suggest AI functioned as a common reference point that facilitated knowledge sharing among team members with diverse programming backgrounds.

Expedited project progress (n = 8): Students valued AI's ability to accelerate debugging and coding tasks, freeing time for other project components. As one student noted, "It helped us dedicate less time to debugging and more time to creating the actual code and designing the float," while another explained, "AI helped my team's teamworking abilities by helping some of the members understand the code and have the code for our project earlier than when it was due. This allowed us all to focus on bigger aspects of the project so everything would work smooth on presentation day."

From a distributed cognition perspective [3], [4], these perceived benefits suggest AI tools can reshape how cognitive work distributes within teams, serving as shared representational resources that reduce expertise barriers and facilitate collective problem-solving. These findings indicate that AI can support team productivity when it functions as a democratizing resource that enhances shared understanding. However, the limited team coordination around AI use may create hidden dependencies or contribution imbalances that teams do not recognize or address collaboratively.

Discussion

This study examined how first-year engineering students use AI tools for Arduino programming in team-based projects. Our findings reveal AI tools were widely adopted (61%) and perceived as

beneficial, yet raised concerns about independent capability development, ethical consistency, and team coordination.

AI as Learning Scaffold Versus Dependency

Most students engaged in active rather than passive AI use, primarily seeking explanations or syntax help (69%) and debugging assistance (56%) rather than complete program generation (22%). These patterns suggest students attempted to maintain agency over AI outputs rather than passively accepting complete solutions. However, a critical tension emerged: while 83% reported understanding most or all of their code, only 19% felt confident reproducing similar work independently. This reveals the pedagogical challenge identified in our theoretical framework [1]: students successfully completed projects with AI assistance yet did not develop the self-sufficiency that defines successful scaffolding.

The finding that 81% reported time savings but only 36% redirected time toward deeper project engagement reveals that efficiency alone does not guarantee enhanced learning. More encouragingly, 61% reported that AI enabled more complex and ambitious code than they could achieve independently, suggesting AI successfully supported students toward greater programming sophistication within the coding domain. However, this vertical depth in programming did not translate to horizontal breadth across engineering disciplines as students invested AI-gained capacity in more sophisticated code rather than the multidisciplinary design exploration that Arduino projects are intended to cultivate.

Ethical Awareness-Practice Gaps

Students demonstrated meaningful ethical awareness: 72% verified outputs, 64% documented AI use, and 89% reported ethical reflection. The AI-Assisted Arduino Coding Guide successfully raised awareness about reliability, attribution, and fairness. However, awareness-practice gaps remain: 36% did not document AI use despite explicit requirements, and 28% did not verify outputs. This suggests understanding ethical principles does not automatically translate to consistent practice.

The 22% of non-users who avoided AI due to ethical concerns reveal normative uncertainty about what constitutes legitimate support. Some viewed AI as acceptable when properly attributed, while others perceived inherent ethical tension in its use, considering it to compromise authentic learning even when permitted.

Team Dynamics and Coordination Gaps

From a distributed cognition perspective [3], [4], AI reconfigured team cognitive systems in both productive and problematic ways. Students overwhelmingly reported improved productivity (83%) and described how AI helped bridge varying expertise levels and expedite progress. However, only 39% discussed AI use with teammates and zero teams established explicit agreements, meaning reconfigurations occurred implicitly rather than through negotiated processes. When AI use remains uncoordinated, potential issues such as uneven contribution,

hidden dependencies, or divergent skill development may go unrecognized by teams focused on task completion.

Implications for Practice

Engineering educators integrating AI tools should provide explicit scaffolding on when AI is appropriate (not just how to use it), require independent practice alongside AI-assisted work, frame time efficiency as opportunity for deeper exploration across project domains, require teams to discuss AI strategies explicitly, integrate ethical considerations throughout projects rather than one-time coverage, and assess transfer capability to reveal dependency patterns. At the curricular level, programs should balance AI-assisted and AI-free assignments to ensure independent skill development, track capability longitudinally, and position team coordination around AI use as an explicit learning objective.

Conclusion

This study provides empirical evidence from team-based, physical computing contexts where hardware-software integration adds complexity beyond typical programming studies. Findings reveal AI offers genuine learning benefits, accessible explanations, debugging support, efficiency gains, and scaffolding toward more complex code, while raising concerns about dependency, ethical practice consistency, and limited team coordination.

The AI-Assisted Arduino Coding Guide demonstrates that comprehensive instructional scaffolding can raise awareness and promote responsible practices, though persistent awareness-practice gaps indicate the need for ongoing reinforcement and iterative support. Key tensions emerged between efficiency and learning depth, perceived and actual competence, vertical programming sophistication and horizontal multidisciplinary exploration, and individual versus team-level integration.

As AI becomes embedded in professional practice, the challenge is not whether to integrate AI but how to do so in ways that develop both AI literacy and independent capability, promote ethical awareness alongside consistent practice, and support equitable collaborative learning that prepares students for AI-augmented engineering careers.

Limitations and Future Work

This study relies on self-report data and is therefore subject to social desirability and recall biases. We lacked performance-based assessments of actual code quality or transfer capability and assessed outcomes immediately after project completion. Future research should address these limitations through multiple approaches. Individual oral examinations would provide direct behavioral evidence of understanding by requiring students to explain their code logic, debug errors in real-time, and reproduce work without AI assistance. Mixed-methods approaches incorporating individual interviews or focus groups would complement surveys by capturing nuanced experiences, ethical tensions, and learning barriers that students may filter from written responses but reveal in verbal discussion. Additional needed work includes longitudinal tracking of skill development across multiple projects, team-level analysis via observations, and cross-

institutional studies to examine how early AI-supported learning experiences influence students' later tool use and broader professional preparation across their undergraduate careers.

References

[1] R. D. Pea, "The Social and Technological Dimensions of Scaffolding and Related Theoretical Concepts for Learning, Education, and Human Activity," *J. Learn. Sci.*, vol. 13, no. 3, pp. 423–451, Jul. 2004, doi: 10.1207/s15327809jls1303_6.

[2] B. R. Belland, A. E. Walker, N. J. Kim, and M. Lefler, "Synthesizing Results From Empirical Research on Computer-Based Scaffolding in STEM Education: A Meta-Analysis," *Rev. Educ. Res.*, vol. 87, no. 2, pp. 309–344, Apr. 2017, doi: 10.3102/0034654316670999.

[3] E. Hutchins, *Cognition in the Wild*. Accessed: Jan. 17, 2026. [Online]. Available: https://direct.mit.edu/books/monograph/4892/Cognition-in-the-Wild

[4] J. Hollan, E. Hutchins, and D. Kirsh, "Distributed cognition: toward a new foundation for human-computer interaction research," *ACM Trans Comput-Hum Interact*, vol. 7, no. 2, pp. 174–196, Jun. 2000, doi: 10.1145/353485.353487.

[5] G. Stahl, *Group cognition: computer support for building collaborative knowledge*. in Acting with technology. Cambridge, Massachusetts: MIT Press, 2006.

[6] S. Barke, M. B. James, and N. Polikarpova, "Grounded Copilot: How Programmers Interact with Code-Generating Models," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, pp. 85–111, Apr. 2023, doi: 10.1145/3586030.

[7] J. Prather *et al.*, "'It's Weird That it Knows What I Want': Usability and Interactions with Copilot for Novice Programmers," *ACM Trans. Comput.-Hum. Interact.*, vol. 31, no. 1, pp. 1–31, Feb. 2024, doi: 10.1145/3617367.

[8] D. Johnson, W. Doss, and C. Estepp, "Using ChatGPT with Novice Arduino Programmers: Effects on Performance, Interest, Self-Efficacy, and Programming Ability," *J. Res. Tech. Careers*, vol. 8, no. 1, p. 1, May 2024, doi: 10.9741/2578-2118.1152.

[9] R. Zviel-Girshin, "The Good and Bad of AI Tools in Novice Programming Education," *Educ. Sci.*, vol. 14, no. 10, p. 1089, Oct. 2024, doi: 10.3390/educsci14101089.

[10] A. L. P. Starling and E. Tian, "GIFTS: Integrating Generative AI into First-Year Engineering Education: From Knowledge Acquisition and Arduino Projects to Defining Accessibility Problems and Solutions," presented at the 2025 ASEE Annual Conference & Exposition, Jun. 2025.