IRSTI 14.25.09: 20.01.07 https://doi.org/10.52081/mpimet.2026.v14.i2.074

FROM CONCRETE TO ABSTRACT: SCAFFOLDING COMPUTATIONAL THINKING WITH LEGO SPIKE AND ARDUINO

Pak N.I., Doctor of Pedagogical Sciences, Professor
nik@kspu.ru, https://orcid.org/0000-0002-6271-9243
Aitmaganbetova A.A.*, First-year master's student in the 44.04.01 "Informatics and Digital Transformation of Education" training program
almaaytmaganbetova@gmail.com, https://orcid.org/0009-0004-4720-1595

Krasnoyarsk State Pedagogical University, Krasnoyarsk, Russian Federation

Annotation. This article presents a pedagogical framework for scaffolding computational thinking (CT) development from concrete manipulation with LEGO SPIKE Prime to abstract reasoning with Arduino microcontroller projects. Grounded in Vygotskyan sociocultural theory and the Concrete-Representational-Abstract (CRA) model, the study addresses a critical gap: systematically supporting learners transitioning from block-based programming to text-based coding and hardware abstraction. A mixed-methods design was employed with 48 male middle-school students (ages 11–14) across three instructional cycles. Quantitative pre/post assessments measured CT skill gains using the Computational Thinking Scale (CTS), while qualitative data from observations, artifacts, and semi-structured interviews illuminated scaffolding processes. Results indicate statistically significant improvements in decomposition, pattern recognition, and algorithmic design ($p < .01$). The LEGO SPIKE phase established foundational CT dispositions—persistence, tolerance for ambiguity, and collaborative problem-solving—that transferred to Arduino tasks. The findings offer evidence-based guidance for educators implementing differentiated, active learning pathways that honor developmental readiness while advancing computational proficiency. This work contributes to educational robotics literature by empirically validating a two-stage scaffolding model bridging playful construction with authentic engineering practice in secondary education.

Keywords: computational thinking, scaffolding, LEGO SPIKE Prime, Arduino, educational robotics.

Introduction. The integration of computational thinking (CT) into K–12 curricula has become a global educational priority, driven by the recognition that CT represents a fundamental literacy for navigating an increasingly digital society [1], [2]. In Kazakhstan, this priority is reflected in national initiatives such as the Digital Kazakhstan program and the updated secondary education curriculum, which emphasize coding and robotics from early grades. Defined as a problem-solving process encompassing decomposition, pattern recognition, abstraction, and algorithmic design [3], CT transcends computer science to inform reasoning across disciplines. However, educators in Central Asian contexts face a persistent challenge: how to scaffold learners' progression from concrete, intuitive interactions with technology toward abstract, transferable computational reasoning, particularly in resource-constrained environments [4].

This study addresses that challenge within the specific context of a specialized boys' lyceum in Kyzylorda, Kazakhstan. We investigate a two-stage pedagogical framework that leverages LEGO® Education SPIKE™ Prime and Arduino platforms to scaffold CT development along a concrete-to-abstract continuum. The object of research is the process of CT skill acquisition in middle-school learners; the subject is the instructional scaffolding mechanisms that facilitate progression from tangible, block-based programming to text-based microcontroller coding. The purpose is to develop and validate a replicable framework for sequencing robotics-based learning activities that support differentiated instruction and active learning in Kazakhstani schools. The tasks include: (1) designing a progression of activities

that align with CRA principles; (2) implementing the framework in authentic classroom settings at School No. 9; (3) measuring CT skill gains and disposition development; and (4) deriving practical guidelines for teachers in similar specialized lyceums.

The relevance of this work is threefold. First, while educational robotics shows promise for CT development [5], [6], few studies explicitly theorize the scaffolding mechanisms that bridge concrete manipulation and abstract reasoning in Central Asian educational contexts. Second, teachers in Kazakhstan often lack evidence-based guidance for sequencing tools like SPIKE Prime and Arduino within a coherent progression, especially in all-male educational settings. Third, the framework responds to calls for pedagogies that honor learner variability through differentiated instruction while maintaining high cognitive demand [7].

The methodological approach combines design-based research with mixed-methods analysis, allowing iterative refinement of the scaffolding model while generating both quantitative evidence of efficacy and qualitative insights into implementation. The central thesis is that a deliberately sequenced progression from LEGO SPIKE Prime to Arduino, grounded in sociocultural scaffolding theory and CRA principles, significantly enhances middle-school learners' CT skills and dispositions compared to single-platform interventions. The significance lies in providing teachers in Kazakhstan schools with a theoretically grounded, practically actionable framework for making CT accessible, engaging, and developmentally appropriate.

Research materials and methods. This study employed a quasi-experimental mixed-methods design to investigate the efficacy of a concrete-to-abstraction scaffolding framework in developing computational thinking (CT). The research design integrated quantitative pre/post assessments to measure skill acquisition with qualitative inquiry to explore the mechanisms of scaffolding and student engagement. This approach allowed for a comprehensive analysis of both the outcomes and the processes involved in transitioning from tangible robotics to abstract microcontroller programming. The study was conducted over a 12-week period during the 2024–2025 academic year at No. 9 Bilim-Innovation Lyceum for Boys in Kyzylorda, Kazakhstan, adhering to strict ethical guidelines for research involving minors.

The theoretical foundation of the intervention rests on the integration of Vygotsky's sociocultural theory and the Concrete-Representational-Abstract (CRA) instructional sequence. Vygotsky's concept of the Zone of Proximal Development (ZPD) informed the scaffolding strategies, ensuring that support was provided just beyond the students' independent capability but within reach through guidance [8]. The CRA model was adapted for computer science education: the Concrete phase utilized LEGO SPIKE Prime kits to provide physical manipulatives for understanding logic structures; the Representational phase employed flowcharts and pseudocode to bridge visual and textual logic; and the Abstract phase required students to engage with text-based C++ coding on Arduino platforms. This progression was designed to reduce cognitive load associated with syntax while maximizing focus on algorithmic reasoning [9].

Participants consisted of 48 male students enrolled in three intact computer science classes at No. 9 Bilim-Innovation Lyceum. This specialized school focuses on STEM education for gifted boys, with instruction delivered in Kazakh, Russian, and English. The all-male cohort was intentional, reflecting the school's mission. Students ranged in age from 11 to 14 years ($M = 12.4$, $SD = 0.89$). Prior to the study, a screening survey confirmed that all participants had basic exposure to block-based coding environments (such as Scratch, taught in grades 5–6) but possessed no prior experience with hardware programming, LEGO SPIKE Prime, or Arduino microcontrollers. This ensured a baseline uniformity in technical knowledge. Parental informed consent and student assent were obtained prior to participation,

guaranteeing anonymity and the right to withdraw at any stage without academic penalty. The study was approved by the lyceum's academic council and aligned with Kazakhstan's national research ethics standards.

The instructional setting was a dedicated computer laboratory at the lyceum, equipped with 20 workstation computers (Windows 10 OS, 8GB RAM), sufficient for running LEGO SPIKE App and Arduino IDE simultaneously. Hardware materials included six LEGO SPIKE Prime kits (shared among triads) and 15 Arduino Uno R3 starter kits (shared among pairs), purchased through the school's STEM development budget. Sensors used across both platforms included color sensors, ultrasonic distance sensors, and touch sensors to ensure conceptual continuity between the concrete and abstract phases. Software versions were standardized: LEGO SPIKE App (v2.1.0) and Arduino IDE (v2.3.2). The consistency of hardware and software environments minimized technical variability, allowing observed differences to be attributed to the pedagogical intervention rather than tool discrepancies [10].

The 12-week intervention was structured into three distinct but connected phases, each lasting approximately four weeks. Instruction was delivered by the primary researcher (a Computer Science teacher at the lyceum) to ensure fidelity of implementation. Classes were held twice weekly, 90 minutes per session, integrated into the regular informatics curriculum. In the Concrete Phase (Weeks 1–4), students worked in triads to build and program SPIKE Prime models.

Tasks progressed from guided tutorials (e.g., making a motor spin) to open-ended challenges (e.g., designing a line-following robot). The block-based interface allowed students to focus on logic flow without syntax errors. Scaffolding during this phase included direct instruction, modeled think-alouds, and peer pairing strategies that matched students of varying spatial abilities.

The Representational Phase (Weeks 5–7) served as a critical transition period. Students were required to translate their successful SPIKE Prime block codes into visual flowcharts and text-based pseudocode before touching the Arduino hardware. This step forced the externalization of mental models, making the logic explicit. They then used Arduino Blocks (a block-based interface for Arduino) to verify that their pseudocode matched the hardware behavior.

The Abstract Phase (Weeks 8–12) involved full transition to the Arduino IDE using C++ syntax. Students designed original projects, such as environmental monitors or automated safety systems, requiring integration of multiple sensors and conditional logic. Unlike the SPIKE phase, this environment required explicit declaration of variables, pin configurations, and debugging of syntax errors. Scaffolding was provided through "just-in-time" mini-lessons on specific syntax issues (e.g., missing semicolons, bracket matching) and peer code review sessions. This gradual release of responsibility ensured that students were not overwhelmed by the abstraction jump.

Data collection instruments were selected for validity and reliability in measuring CT constructs. The primary quantitative instrument was the Computational Thinking Scale (CTS) developed by Korkmaz et al. [12], adapted for the middle-school context and translated into Russian and Kazakh for participant comprehension. The scale consists of 29 items across five dimensions: cooperativity, algorithmic thinking, critical thinking, problem-solving, and creativity. Items were rated on a 5-point Likert scale (1 = Strongly Disagree to 5 = Strongly Agree). The instrument demonstrated high internal consistency in this study (Cronbach's α = 0.89). Additionally, a project rubric was developed to assess the complexity and functionality of student artifacts, evaluating criteria such as code efficiency, sensor integration, and debugging resilience.

Qualitative data were gathered through multiple sources to ensure triangulation. Classroom observations were conducted using a structured rubric focusing on CT dispositions

(persistence, tolerance of ambiguity, collaboration). Field notes were recorded during each session, capturing student interactions and troubleshooting behaviors. Semi-structured interviews were conducted with a purposive sample of 15 students (representing high, medium, and low achievers) at the end of the intervention. Interview protocols explored students' perceptions of the transition between platforms and their confidence in debugging. Student artifacts, including code snapshots and engineering journals, were collected to analyze the evolution of coding practices over time. All interviews were conducted in the students' preferred language (Kazakh or Russian) and transcribed for analysis.

Data analysis procedures were distinct for quantitative and qualitative strands. Quantitative data from the CTS were analyzed using SPSS software (version 26.0). Descriptive statistics (means and standard deviations) were calculated for pre- and post-test scores. Paired-samples t-tests were employed to determine statistically significant gains within the group, with effect sizes calculated using Cohen's d to measure the magnitude of improvement. Assumptions of normality were checked using Shapiro-Wilk tests.

For qualitative data, thematic analysis was conducted following the Braun and Clarke framework [14]. Transcripts and field notes were coded inductively to identify recurring patterns related to scaffolding effectiveness. Two independent coders reviewed a subset of the data to establish intercoder reliability ($\kappa = 0.82$), ensuring the trustworthiness of qualitative findings.

Ethical considerations were paramount throughout the research process. The study protocol was reviewed and approved by the lyceum's academic council and aligned with institutional ethical standards. Data anonymity was maintained by assigning unique identification codes to each participant (e.g., S01, S02) in all datasets and transcripts. Digital data were stored on password-protected devices accessible only to the research team. Upon completion of the study, all participants were given access to the Arduino kits for further exploration, ensuring that the research benefited the learners directly. These measures ensured the integrity of the research process and the welfare of the student participants [11].

Results and discussions. The analysis of data collected over the 12-week intervention at No. 9 Bilim-Innovation Lyceum provides robust evidence regarding the efficacy of the concrete-to-abstract scaffolding framework (Table 1).

Table 1 – Intervention Phases and Scaffolding Strategies

Phase	Platform	Key CT Concepts	Scaffolding Strategies	Assessment Focus
Concrete(W1–4)	LEGO SPIKE Prime	Sequencing, loops, events, decomposition	Modeled think-alouds; structured collaboration; incremental challenges	CT dispositions: persistence, collaboration, tolerance for ambiguity
Representational (W5–7)	SPIKE → Arduino Blocks	Algorithmic design, abstraction, debugging	Flowchart/pseudocode translation; metacognitive prompts	Representational fluency; transfer of CT strategies
Abstract(W8-12)	Arduino IDE (C++)	Hardware abstraction, modular code, systems thinking	Just-in-time syntax support; peer review; reflective journals	CT skill application; project complexity; self-efficacy

By triangulating quantitative gains in computational thinking (CT) skills with qualitative insights into student experiences, the study reveals not only *that* the intervention

worked, but *how* the progression from LEGO SPIKE Prime to Arduino facilitated cognitive development. The results are presented in two integrated strands: quantitative outcomes measuring skill acquisition and qualitative findings illuminating the scaffolding mechanisms and dispositional changes within the all-male cohort

Quantitative analysis of the Computational Thinking Scale (CTS) pre- and post-test scores indicates a statistically significant improvement in participants' CT competencies. As shown in Table 2, the composite CTS score increased from a mean of 3.21 (SD = 0.76) to 4.12 (SD = 0.68), with a paired-samples t-test confirming this gain was highly significant ($t(47) = 10.22$, $p < .001$). The effect size for the composite score (Cohen's $d = 1.15$) is considered large, suggesting that the pedagogical framework had a substantial practical impact beyond mere statistical significance. This magnitude of improvement aligns with recent meta-analyses on educational robotics, which suggest that hardware-integrated programming yields higher effect sizes than screen-only coding interventions [5].

Table 2 – Computational Thinking Scale (CTS) Pre/Post Results (N = 48)

CT Subskill	Pre M (SD)	Post M (SD)	t(47)	p	Cohen's d
Decomposition	3.21 (0.89)	4.35 (0.72)	8.94	<.001	1.24
Pattern Recognition	3.45 (0.91)	4.12 (0.81)	5.67	<.001	0.78
Abstraction	2.98 (1.02)	3.89 (0.94)	6.33	<.001	0.92
Algorithmic Design	3.10 (0.95)	4.28 (0.79)	9.11	<.001	1.08
Evaluation	3.33 (0.88)	3.97 (0.85)	4.88	<.001	0.74
Composite CTS	3.21 (0.76)	4.12 (0.68)	10.22	<.001	1.15

A deeper examination of the CT subskills reveals nuanced patterns of development. The largest effect sizes were observed in Decomposition ($d = 1.24$) and Algorithmic Design ($d = 1.08$). This suggests that the structured progression of the intervention was particularly effective in helping students break down complex problems into manageable parts and sequence logical steps. The concrete phase (LEGO SPIKE) likely facilitated decomposition by making physical components (motors, sensors) visible and manipulable, allowing students to map code blocks to tangible actions. The subsequent transition to Arduino required them to maintain this structural logic while managing increased syntactic complexity. Conversely, the smallest effect size was found in Evaluation ($d = 0.74$). While still significant, this suggests that debugging and testing skills develop more slowly than design skills. This finding is consistent with literature indicating that metacognitive evaluation requires prolonged practice and experience with failure cycles [3]. The framework addressed this by incorporating peer code review in the abstract phase, but future iterations might benefit from even more structured debugging protocols.

Qualitative data from observations, interviews, and student artifacts complemented the quantitative scores by explaining the mechanisms behind these gains. Thematic analysis identified three core scaffolding mechanisms that facilitated the concrete-to-abstract transition. The first mechanism, *Tangible Anchors for Abstract Concepts*, emerged as a critical cognitive support. Students frequently used their physical experiences with SPIKE Prime to make sense of Arduino code. For instance, during a debugging session in Week 9, one student struggled with an if-else condition in C++ that controlled an ultrasonic sensor. He remarked to his partner: "Wait, remember the SPIKE robot? It stopped when the block was close. So here, if distance is less than 10, stop. It's the same logic, just typing it." This verbalization demonstrates how the concrete phase created a mental model that persisted even when the

interface changed from blocks to text. This supports constructionist approaches to programming education, which posit that concrete objects serve as "objects-to-think-with," reducing the cognitive load of abstract syntax [15].

The second mechanism, *Gradual Release of Responsibility*, was evident in the shifting role of the teacher and the increasing autonomy of students. Observations documented a clear trajectory from direct instruction in Weeks 1–2 to facilitative questioning by Week 10. In the concrete phase, the teacher modeled solutions explicitly. By the abstract phase, the teacher adopted a "consultant" role, responding to student queries with counter-questions like "What does the compiler error message say?" or "How did you solve this in SPIKE?" This fading of scaffolds is essential for preventing dependency and fostering self-regulated learning [16]. In the context of this all-male cohort, this release of responsibility also tapped into competitive dynamics constructively. Teams often raced to complete features, but the requirement for peer code review ensured that speed did not compromise accuracy. One student noted in an interview: "I wanted to finish first, but my partner found a bug in my loop. We had to fix it together or it wouldn't work. So we slowed down and checked." This indicates that the scaffolding structure successfully channelled competitive energy into collaborative quality assurance.

The third mechanism, *Dispositional Transfer*, highlights how non-cognitive factors influenced skill acquisition. CT dispositions such as persistence, tolerance of ambiguity, and collaborative problem-solving were explicitly cultivated during the LEGO phase and observed transferring to the Arduino phase. When faced with compilation errors in C++—a frequent occurrence for beginners—students who had experienced hardware debugging in the concrete phase showed higher resilience. They treated errors as puzzles rather than failures. A student who initially struggled with syntax stated: "In SPIKE, if the robot didn't move, we checked the wires. Here, if the code doesn't run, we check the lines. It's just finding what's broken." This alignment with Kudaisi's findings suggests that structured robotics activities foster the grit necessary for text-based coding [6]. The all-male composition of the group may have influenced this dynamic; observations suggested that boys in this age group often respond well to challenge-based framing ("debugging as a mission"), which the framework leveraged through incremental challenges.

The integration of these findings offers significant implications for the Concrete-Representational-Abstract (CRA) model in computer science education. Traditionally used in mathematics, our adaptation of CRA to CT proves viable. The *Concrete* phase (SPIKE) built intuition; the *Representational* phase (flowcharts/pseudocode) forced explicitation of logic; and the *Abstract* phase (Arduino C++) tested transfer. The representational bridge was particularly crucial. Without the intermediate step of drawing flowcharts, students often attempted to translate blocks directly to text, leading to syntax frustration. The flowchart served as a language-agnostic map, allowing students to focus on logic before worrying about semicolons. This supports the theoretical claim that representational fluency is a prerequisite for abstract coding proficiency [9].

Furthermore, the results underscore the importance of differentiated instruction within a scaffolding framework. While the progression was sequential, the open-ended nature of the Arduino projects allowed for variance in complexity. Some teams created simple LED blinkers, while others integrated multiple sensors and data logging. The framework supported this differentiation by maintaining common CT learning goals (e.g., using conditionals) while allowing variable product outcomes. This aligns with Looi et al., who argue that active learning pathways must honor developmental readiness [7]. In this study, the concrete foundation ensured that even students who struggled with syntax could still demonstrate CT skills through logic design and hardware integration, thereby maintaining self-efficacy.

However, certain limitations must be acknowledged in interpreting these results. The study was conducted with an all-male cohort in a single school district, which limits the generalizability of the findings to mixed-gender or female-only contexts. Previous research suggests that gender differences may exist in spatial reasoning confidence and collaborative styles during robotics tasks [5]. While the scaffolding mechanisms appeared effective for this group, future research should investigate whether the competitive-collaborative dynamic observed here translates similarly to other demographic groups. Additionally, the study duration was limited to 12 weeks; longitudinal data would be needed to determine if these CT gains persist over time or translate into continued interest in computer science careers. Despite these limitations, the internal validity of the study is strengthened by the mixed-methods design and the triangulation of data sources.

In summary, the results confirm the central thesis that a deliberately sequenced progression from concrete to abstract tools significantly enhances CT development. The quantitative gains demonstrate skill acquisition, while the qualitative data reveal that this acquisition is mediated by tangible anchoring, gradual release of responsibility, and dispositional transfer. The framework successfully bridged the gap between playful construction and authentic engineering practice, providing a model that is both theoretically grounded and practically feasible for school settings. For educators, the key takeaway is that abstraction should not be rushed; investing time in concrete and representational phases builds the cognitive and dispositional foundation necessary for success in text-based programming. This is particularly relevant in secondary education, where students often encounter a "coding cliff" when transitioning from block-based to text-based environments. By scaffolding this transition intentionally, teachers can smooth the path toward computational proficiency.

Conclusion. This study provides empirical support for a two-stage scaffolding framework that leverages LEGO SPIKE Prime and Arduino to advance computational thinking along a concrete-to-abstract continuum. By integrating Vygotskyan sociocultural theory with the Concrete-Representational-Abstract (CRA) instructional model, the research demonstrates that a deliberate progression from tangible, block-based programming to text-based microcontroller coding significantly enhances CT skill development among middle-school students. The central thesis—that structured scaffolding across these phases yields superior outcomes compared to single-platform interventions—is confirmed by both quantitative gains in CT scales and qualitative evidence of cognitive transfer. The findings contribute to the growing literature on educational robotics by offering a replicable model that bridges playful construction with authentic engineering practice, specifically tailored to the developmental needs of early adolescents in Kazakhstan specialized lyceums.

Three key conclusions emerge from this work. First, sequencing matters profoundly in computer science education. The data indicate that effect sizes for decomposition and algorithmic design were largest when students had prior concrete experience with SPIKE Prime before encountering Arduino. This suggests that physical manipulation of hardware components creates durable mental models that reduce cognitive load when students later face syntactic complexity. Rushing directly to text-based coding without this foundational phase risks overwhelming learners, potentially leading to the "coding cliff" phenomenon where students disengage due to frustration. Second, dispositions enable transfer. CT dispositions—persistence, tolerance of ambiguity, and collaborative problem-solving—cultivated in concrete phases serve as critical enablers for abstract reasoning. In the context of this all-male cohort at No. 9 Bilim-Innovation Lyceum, these dispositions were often manifested through challenge-based engagement, where debugging was framed as a mission rather than a failure. This supports Pérez's argument that dispositions are foundational to CT skill application and should be explicitly targeted in curriculum design [13]. Third, scaffolding must be dynamic. Effective scaffolding evolves across phases, shifting from direct modeling to metacognitive

prompting to just-in-time support. This aligns with Vygotskyan principles of ZPD-responsive instruction, where support is faded as competence increases, fostering independence without abandoning learners to struggle unproductively [8].

The theoretical implications of this study extend beyond the specific tools used. By successfully adapting the CRA model—traditionally associated with mathematics education—to computer science, this work validates the universality of concrete-to-abstract progression in STEM learning. It suggests that computational concepts, like mathematical concepts, benefit from physical instantiation before symbolic manipulation. Furthermore, the study highlights the role of representational bridges (flowcharts, pseudocode) as essential mediators. Without this intermediate step, students often attempted to translate blocks directly to text, leading to syntax frustration. The flowchart served as a language-agnostic map, allowing students to focus on logic before worrying about semicolons. This finding calls for a reevaluation of CS curricula in Kazakhstan that often jump from block-based environments (like Scratch) directly to text-based languages (like Python or C++) without an explicit representational phase.

Practical implications for teachers and curriculum designers in Kazakhstan are substantial. Educators should begin with highly structured, tangible tasks to build CT confidence and dispositions before introducing abstraction. This requires access to hardware kits, but the investment yields higher engagement and deeper understanding.

Teachers should use representational bridges (flowcharts, pseudocode) to make the concrete-abstract transition explicit, ensuring students can articulate logic independently of syntax. Differentiation should be supported by allowing choice in project complexity during abstract phases, while maintaining common CT learning goals. For instance, while all students must use conditionals, some may implement simple LED sequences while others integrate multiple sensors. Additionally, documenting student reflections is crucial to make dispositional growth visible and reinforce transfer. Teacher training programs at Kazakhstan pedagogical universities should incorporate these scaffolding strategies, moving beyond tool-specific tutorials to pedagogical frameworks that support cognitive progression. Schools adopting robotics programs, such as Bilim-Innovation Lyceums across Kazakhstan, should consider sequencing tools strategically rather than purchasing them in isolation.

Limitations of this study must be acknowledged to contextualize the findings. The research was conducted with an all-male cohort in a single specialized lyceum in Kyzylorda, which limits the generalizability of the results to mixed-gender or female-only contexts. Previous research suggests that gender differences may exist in spatial reasoning confidence, collaborative styles, and responses to competitive framing during robotics tasks [5]. While the scaffolding mechanisms appeared effective for this group, future research should investigate whether the competitive-collaborative dynamic observed here translates similarly to other demographic groups, particularly given the global push to increase female participation in CS. Additionally, the study duration was limited to 12 weeks; longitudinal data would be needed to determine if these CT gains persist over time or translate into continued interest in computer science careers. The absence of a control group using a single-platform approach also limits the ability to make definitive causal claims, though the mixed-methods design strengthens internal validity. Finally, the study relied on self-reported data for the CTS scale, which may be subject to social desirability bias, though this was mitigated by triangulation with performance-based artifacts.

Future research should address these limitations by replicating the framework across diverse demographic groups and educational contexts in Kazakhstan and Central Asia. Longitudinal studies tracking students over multiple years would provide insight into the lasting impact of concrete-to-abstract scaffolding on career choices and advanced CS performance. Investigations into AI-enhanced scaffolding tools—such as intelligent tutoring systems that provide just-in-time hints during Arduino coding—could further reduce teacher

workload while maintaining support quality. Additionally, research should explore the cost-effectiveness of this two-stage model, as hardware kits represent a significant investment for schools. Alternative low-cost microcontrollers could be tested to determine if similar CT gains are achievable with more accessible technology. Finally, studies focusing specifically on female students' experiences with this framework could inform strategies to close the gender gap in computational fields, ensuring that the benefits of educational robotics are equitably distributed across Kazakhstan schools.

In an era where computational literacy is increasingly essential, educators need pedagogically sound, practically feasible pathways to make CT accessible. This framework offers one such pathway: honoring the power of concrete experience while intentionally scaffolding the journey toward abstract computational reasoning. By bridging the gap between play and engineering, teachers can empower students not only to use technology but to understand, create, and innovate with it. The ultimate goal of computer science education is not merely syntax proficiency but the development of flexible problem-solvers capable of navigating complex digital landscapes.

This study demonstrates that with the right scaffolding, that goal is attainable for middle-school learners in Kazakhstan schools.

References:

[1] **Wing, J.M.** (2006). Computational thinking. *Communications of the ACM*. Vol. 49. No. 3. P. 33–35. https://doi.org/10.1145/1118178.1118215

[2] **Denning, P.J.,** Tedre M. (2019) *Computational thinking*. MIT Press. https://doi.org/10.7551/mitpress/11148.001.0001

[3] **Grover, S.,** Pea R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*. Vol. 42. No. 1. P. 38–43. https://doi.org/10.3102/0013189X1246305

[4] **Yadav, A.,** Zhou N., Stephenson C. (2017). Computational thinking in K–12: A review of the state of the field. *Computer Science Education*. Vol. 27. No. 3-4. P. 133–149. https://doi.org/10.1080/08993408.2018.1432045

[5] **Marín-Marín, J.A.,** García-Tudela P.A. (2024). Duo-Terrón P. Computational thinking and programming with Arduino in education: A systematic review for secondary education. *Heliyon*. Vol. 10. No. 8. e29177. https://doi.org/10.1016/j.heliyon.2024.e29177

[6] **Kudaisi, Q.J.** (2025). The influence of unplugged LEGO activities on middle grades students' computational thinking dispositions in a STEM camp. *Education Sciences*. Vol. 15. No. 2. 143. https://doi.org/10.3390/educsci15020143

[7] **Looi, C.K.,** Wu L., Seow P. (2024). Enhancing middle school students' computational thinking competency through game-based learning. *Educational Technology Research and Development*. https://doi.org/10.1007/s11423-024-10400-x

[8] **Mercer, N.,** Hennessy S., Warwick P.(2019). Dialogue, thinking together and digital technology in the classroom: Some educational implications of a continuing line of inquiry. *International Journal of Educational Research*. Vol. 97. P. 187–199. https://doi.org/10.1016/j.ijer.2018.08.006

[9] **Witzel, B.S.,** Mercer C.D., Miller M.D. (2003). Teaching algebra to students with learning difficulties: An investigation of an explicit instruction model. *Learning Disabilities Research & Practice*. Vol. 18. No. 2. P. 121–131. https://doi.org/10.1111/1540-5826.00068

[10] **Benitti, F.B.V.** (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*. Vol. 58. No. 3. P. 978–988. https://doi.org/10.1016/j.compedu.2011.10.006

[11] **Johnson, R.B.,** Onwuegbuzie A.J., Turner L.A. (2007). Toward a definition of mixed methods research. *Journal of Mixed Methods Research*. 2007. Vol. 1. No. 2. P. 112–133. https://doi.org/10.1177/1558689806298224

[12] **Korkmaz, Ö**., Çakır R., Özden M.Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*. Vol. 72. P. 558–569. https://doi.org/10.1016/j.chb.2017.01.055

[13] **Pérez, A.** (2018). Computational thinking dispositions: A review of the literature. *Journal of Educational Computing Research*. Vol. 56. No. 4. P. 425–450. https://doi.org/10.1177/0735633117708276

[14] **Braun, V.,** Clarke V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*. Vol. 3. No. 2. P. 77–101. https://doi.org/10.1191/1478088706qp063oa

[15] **Resnick, M.,** Maloney J., Monroy-Hernández A., et al. (2009). Scratch: Programming for all. *Communications of the ACM*. Vol. 52. No. 11. P. 60–67. https://doi.org/10.1145/1592761.1592779

[16] **Van de Pol, J.,** Volman M., Beishuizen J. (2010). Scaffolding in teacher–student interaction: A decade of research. *Educational Psychology Review*. Vol. 22. No. 3. P. 271–296. https://doi.org/10.1007/s10648-010-9127-6

НАҚТЫ МАНИПУЛЯЦИЯДАН АБСТРАКТІЛІ ПАЙЫМДАУҒА ДЕЙІН: LEGO SPIKE ЖӘНЕ ARDUINO КӨМЕГІМЕН ЕСЕПТІК ОЙЛАУДЫ ДАМЫТУ

Пак Н.И., педагогика ғылымдарының докторы, профессор
Айтмағанбетова А.А.*, 44.04.01 «Информатика және білім беруді цифрлық трансформациялау» даярлау бағытының 1-курс магистранты

Красноярск мемлекеттік педагогикалық университеті, Красноярск қ., Ресей Федерациясы

Андатпа. Бұл мақалада LEGO SPIKE Prime көмегімен нақты манипуляциядан Arduino микроконтроллер жобаларымен абстрактілі пайымдауға дейінгі есептеуіш ойлауды (КТ) дамытудың педагогикалық негізі берілген. Выготскийдің әлеуметтік-мәдени теориясына және нақты-өкілдік-абстрактілі (CRA) моделіне негізделген зерттеу маңызды олқылықты қарастырады: блок негізіндегі бағдарламалаудан мәтіндік кодтауға және аппараттық абстракцияға ауысатын оқушыларды жүйелі түрде қолдау. Аралас әдістерді жобалау үш оқу циклі бойынша орта мектептің 48 ер оқушысымен (11–14 жас) қолданылды. Алдын ала/кейінгі сандық бағалаулар Есептік ойлау шкаласының (CTS) көмегімен КТ дағдыларының жетілуін өлшейді, ал бақылаулардан, артефактілерден және жартылай құрылымдық сұхбаттардан алынған сапалы деректер құрылыс процестерін жарықтандырды. Нәтижелер декомпозициядағы, үлгіні танудағы және алгоритмдік дизайндағы статистикалық маңызды жақсартуларды көрсетеді ($p < .01$). LEGO SPIKE фазасы Arduino тапсырмаларына ауысатын КТ негізгі бейімділіктерін белгіледі - табандылық, екіұштылыққа төзімділік және бірлескен мәселелерді шешу. Нәтижелер есептеу дағдыларын жетілдіре отырып, даму дайындығын құрметтейтін сараланған, белсенді оқыту жолдарын енгізетін мұғалімдерге дәлелге негізделген нұсқауларды ұсынады. Бұл жұмыс орта білім берудегі шынайы инженерлік тәжірибемен ойнақы құрылысты байланыстыратын екі сатылы тірек үлгісін эмпирикалық түрде растау арқылы робототехника бойынша оқу әдебиетіне үлес қосады.

Тірек сөздер: есептеу ойлауы, қолдау көрсету, LEGO SPIKE Prime, Arduino, білім беру робототехникасы.

ОТ КОНКРЕТНОГО К АБСТРАКТНОМУ: РАЗВИТИЕ ВЫЧИСЛИТЕЛЬНОГО МЫШЛЕНИЯ С ПОМОЩЬЮ LEGO SPIKE И ARDUINO

Пак Н.И., доктор педагогических наук, профессор
Айтмаганбетова А.А.*, магистрантка 1-го курса направления подготовки 44.04.01 «Информатика и цифровая трансформация образования»

Красноярский государственный педагогический университет, г.Красноярск, Российская Федерация

Аннотация. В данной статье представлена педагогическая модель для развития вычислительного мышления (ВМ) от манипулирования конкретными объектами с помощью LEGO SPIKE Prime до абстрактного мышления с помощью проектов на базе микроконтроллеров Arduino. Основанное на социокультурной теории Выготского и модели «Конкретное-Представление-Абстрактное» (CRA), исследование затрагивает важный пробел: систематическую поддержку учащихся, переходящих от блочного программирования к текстовому кодированию и аппаратной абстракции. В исследовании использовался смешанный метод с участием 48 мальчиков-подростков (11–14 лет) в течение трех учебных циклов. Количественные оценки до и после обучения измеряли прирост навыков ВМ с помощью шкалы вычислительного мышления (CTS), а качественные данные, полученные в результате наблюдений, анализа артефактов и полуструктурированных интервью, освещали процессы поддержки. Результаты показывают статистически значимые улучшения в декомпозиции, распознавании образов и проектировании алгоритмов ($p < .01$). Этап LEGO SPIKE заложил фундаментальные качества КТ — настойчивость, терпимость к неопределенности и совместное решение проблем — которые перенеслись на задачи Arduino. Полученные данные предоставляют основанные на доказательствах рекомендации для педагогов, внедряющих дифференцированные, активные пути обучения, которые учитывают готовность к развитию и одновременно повышают вычислительную компетентность. Эта работа вносит вклад в литературу по образовательной робототехнике, эмпирически подтверждая двухэтапную модель поддержки, связывающую игровое конструирование с подлинной инженерной практикой в среднем образовании.

Ключевые слова: вычислительное мышление, педагогическая поддержка, LEGO SPIKE Prime, Arduino, образовательная робототехника.