OmniaScience

Journal of Technology and Science Education

JOTSE, 2026 – 16(1): 95-108 – Online ISSN: 2013-6374 – Print ISSN: 2014-5349

https://doi.org/10.3926/jotse.3402

EXPLORING THE RELATIONSHIP BETWEEN COMPUTATIONAL THINKING DIMENSIONS AND ACHIEVEMENT IN ROBOTICS PROGRAMMING AMONG COMPUTER EDUCATION STUDENTS

Fadip Audu Nannim[1*], Moeketsi Mosia[1], Modesta Ero Ezema[2], Felix Egara[1]

[1]Department of Mathematics, Natural Sciences and Technology Education, University of the Free State, Bloemfontein (South Africa)

[2]Department of Computer Science, University of Nigeria, Nsukka (Nigeria)

[*]*Corresponding author: fadip.nannim@unn.edu.ng, MosiaMS@ufs.ac.za, modesta.ezema@unn.edu.ng, egara.fo@ufs.ac.za*

Received March 2025

Accepted October 2025

Abstract

This study investigates the relationship between computational thinking (CT) dimensions and achievement in robotics programming among computer education students in Southeast Nigeria. A correlational research design was adopted, and data were collected from 105 students and analysed using Pearson correlation and multiple regression techniques. The results show that algorithmic thinking exhibits the strongest positive correlation with achievement ($r = .596$, $p < .001$), followed by evaluation ($r = .449$, $p < .001$), generalisation ($r = .416$, $p < .001$), and abstraction ($r = .374$, $p < .001$). However, decomposition negatively correlated with achievement ($r = -.392$, $p < .001$), indicating difficulties in effectively applying decomposition skills to robotics programming tasks. Collectively, the five CT dimensions significantly predict students' achievement, accounting for 74.5% of the variance ($r^2 = .745$, $F(5,99) = 57.845$, $p < .001$). Regression analysis further identifies algorithmic thinking as the strongest positive predictor ($\beta = 1.116$, $p < .001$), while decomposition ($\beta = -.630$, $p < .001$) and evaluation ($\beta = -.449$, $p < .001$) negatively impact achievement. Additionally, students' self-perceptions of their CT abilities generally align with their performance, except for decomposition, where a negative relationship suggests potential gaps in skill application. These results underscore the critical role of algorithmic thinking in robotics programming success and suggest the need for targeted interventions to address challenges associated with decomposition and evaluation skills. The study offers valuable insights for educators and curriculum developers aiming to enhance computational thinking instruction in robotics education.

Keywords **–** Achievement, Computer education, Computational thinking, Robotics programming, Algorithmic thinking.

To cite this article:

Nannim, F.A., Mosia, M., Ezema, M.E., & Egara, F. (2026). Exploring the relationship between computational thinking dimensions and achievement in robotics programming among computer education students. *Journal of Technology and Science Education,* 16(1), 95-108. https://doi.org/10.3926/jotse.3402

---

1. Introduction

There is a silent revolution going on in the busy classrooms of Southeast Nigeria. Students used to just learn in traditional ways, but today they are learning about robotics programming, which is a fast-paced field. This shift is not just about learning to code; it's about fostering a new way of thinking, computational thinking (CT). As these students navigate the complexities of programming robots, they are developing skills that extend far beyond the classroom, preparing them for a future where technology and innovation are paramount.

Computational Thinking (CT), a term popularised by Jeannette Wing in 2006, refers to a problem-solving process that includes a number of characteristics, such as logically organising and analysing data, breaking down complex problems into manageable parts, and creating algorithms to solve these problems (Palop, Díaz, Rodríguez-Muñiz & Santaengracia, 2025; Wing, 2006). CT is increasingly recognised as a fundamental skill for all students, not just those pursuing careers in computer science (Acevedo-Borrega, Valverde-Berrocoso & Garrido-Arroyo, 2022; Grover & Pea, 2013). It encompasses a range of cognitive skills and processes, including abstraction, algorithmic thinking, decomposition, evaluation and generalisation (Rao & Bhagat, 2024). Abstraction in computational thinking refers to the process of filtering out unnecessary details and focusing on essential features of a problem to develop a general solution (Wing, 2006). This skill is fundamental in programming and robotics, where students must identify key components and relationships while ignoring extraneous elements. Recent studies highlight the importance of abstraction in problem-solving, particularly in robotics, artificial intelligence (AI) and data science applications (Gizzi, Nair, Chernova & Sinapov, 2022; Grover & Pea, 2018). However, despite its significance, many students struggle with abstraction due to difficulties in distinguishing between relevant and irrelevant details (Shute, Sun & Asbell-Clarke, 2017). There is a need for more instructional strategies that explicitly teach abstraction skills, particularly through hands-on activities and real-world applications in robotics programming.

Decomposition, on the other hand, involves breaking down complex problems into smaller, more manageable sub-problems, making them easier to analyse and solve (Selby & Woollard, 2014). This approach is widely used in computational problem-solving, software engineering, and robotics (Brennan & Resnick, 2012). Research has shown that effective decomposition enhances students' ability to tackle large-scale problems systematically (Waite, Curzon, Marsh & Sentance, 2020). However, there is a paucity of studies on the correlation between decomposition and achievement in robotics programming. Since developing decomposition skills in programming is often not easy (Kwon & Cheon, 2019), students may struggle with correctly structuring sub-problems. This suggests that further research into instructional strategies is necessary to help students develop efficient decomposition skills while ensuring that breaking down problems does not interfere with overall problem comprehension and achievement. Algorithmic thinking refers to the ability to design and apply step-by-step procedures to solve problems efficiently (Denning, 2009). It is a fundamental component of computational thinking and is particularly essential in robotics programming, where students must develop precise sequences of instructions for their robots to execute (Grover & Pea, 2018). Research has consistently shown that strong algorithmic thinking skills lead to improved programming performance (Csizmadia, Standl & Waite, 2019). However, many students still struggle with creating optimised and error-free algorithms (Sabuncuoglu & Sezgin, 2022), which can hinder their success in programming. While existing studies emphasise the importance of algorithmic thinking, there is still a paucity of research on the relationship between algorithmic thinking and students' achievement in robotics education.

Evaluation in computational thinking involves assessing the effectiveness and efficiency of solutions, debugging errors, and refining algorithms to improve performance (Shute et al., 2017). In robotics programming, students must analyse their code and test their robots to determine whether their solutions meet the desired objectives. Studies indicate that students who engage in frequent evaluation processes tend to develop stronger problem-solving and debugging skills (Chuang & Chang, 2024; Brennan & Resnick, 2012). Generalisation refers to the ability to apply previously learned concepts and solutions to new problems, thereby enhancing problem-solving efficiency and adaptability (Selby &

Woollard, 2014). In robotics programming, students must recognise patterns and transfer coding principles across different tasks. Research has shown that generalisation enhances computational fluency and fosters creative problem-solving (Aranda & Ferguson, 2022; Waite et al., 2020). Yet, few studies have investigated the relationship between generalisation and achievement in robotics programming.

Robotics programming involves designing, writing, and testing code that controls robots. This field has gained significant traction in educational settings due to its hands-on nature and its ability to make abstract concepts tangible (Talan, 2021). Research has shown that robotics programming can enhance students' engagement, motivation, and understanding of complex STEM concepts (Kert, Erkoç & Yeni, 2020). Moreover, it provides a practical context for applying computational thinking skills, thereby reinforcing and deepening students' understanding (Noh & Lee, 2020; Yıldız & Seferoğlu, 2021).

Despite the growing body of research on computational thinking (CT) and robotics programming, there are still some big gaps. A significant portion of the literature comes from Western contexts, with little focus on African settings, especially Nigeria and Southeast Nigeria (Rovshenov & Sarsar, 2023). Recent regional initiatives have commenced efforts to rectify this deficiency, with few studies illustrating the capacity of educational robots to improve students' programming achievement, task persistence, and computational thinking (Nannim, Ibezim, Mosia & Oguguo, 2025; Nannim, Ibezim, Oguguo & Nwangwu, 2024). Others have investigated alternative ways, including unplugged methods, to facilitate the computational thinking of novice learners (Agbo, Okpanachi, Ocheja, Oyelere & Sani, 2024). Nonetheless, research into the particular aspects of computational thinking most affected by robotics programming is still very scarce, highlighting the necessity for contextualised research that enhances comprehension of how robotics might facilitate computational thinking in African educational settings (Rao & Bhagat, 2024). Furthermore, there is a lack of correlational studies examining the impact of the dimension of CT on students' academic achievement in robotics programming. Therefore, this study aims to address the lack of comprehensive research on the relationship between computational thinking dimensions and achievement in robotics programming among computer education students in Southeast Nigeria. By exploring this relationship, the study seeks to fill the existing research gaps and provide insights that can inform educational practices and policies in the region. The following research questions were raised to guide the study:

1. What is the relationship between each dimension of computational thinking (abstraction, decomposition, algorithmic thinking, evaluation, and generalisation) and achievement in robotics programming among computer education students in Southeast Nigeria?
2. How do the dimensions of computational thinking collectively predict achievement in robotics programming among computer education students?
3. Which dimension of computational thinking is the strongest predictor of achievement in robotics programming among computer education students?
4. How do students' perceptions of their computational thinking abilities correlate with their actual performance in robotics programming?

1.1. Theoretical Framework

1.1.1. Constructionism Theory

Constructionism, developed by Seymour Papert, is a learning theory that emphasises the importance of learners actively constructing their own knowledge through hands-on, meaningful activities. This theory builds on Jean Piaget's constructivist ideas, suggesting that knowledge is best acquired when learners are engaged in creating tangible artefacts that reflect their understanding. Constructionism posits that learners construct knowledge most effectively when they are involved in creating something meaningful (Papert, 1980, 1993). This making process allows learners to experiment, iterate, and refine their understanding. The theory advocates for project-based learning, where students work on projects that are personally

meaningful and relevant. This approach encourages deep engagement and sustained interest. Additionally, constructionism emphasises the social nature of learning. Collaboration with peers provides opportunities for learners to share ideas, receive feedback, and build on each other's knowledge. Reflection is also a critical component of constructionism. Learners are encouraged to think about their learning process, evaluate their progress, and make adjustments as needed.

In the context of this study, constructionism provides a robust framework for understanding how computer education students in Southeast Nigeria develop computational thinking skills through robotics programming. The hands-on, project-based nature of robotics programming aligns well with the principles of constructionism, as students actively engage in creating and programming robots, which reinforces their computational thinking skills. As students engage in robotics programming, they construct knowledge by building and programming robots. This hands-on activity allows them to apply computational thinking skills such as abstraction, decomposition, and algorithmic thinking in a practical context. For example, when students program a robot to navigate a maze, they must break down the problem into smaller tasks, develop algorithms, and test their solutions, thereby reinforcing their understanding of computational concepts.

The 8-week robotics programming course is structured around projects that are meaningful and relevant to the students. These projects provide a context for students to apply their computational thinking skills in a sustained and engaging manner. For instance, students might work on projects such as designing a robot to perform specific tasks, which requires them to apply and integrate various computational thinking dimensions. Robotics programming often involves teamwork, where students collaborate to solve problems and build robots. This collaborative environment fosters the sharing of ideas, peer learning, and collective problem-solving. Through collaboration, students can learn from each other's experiences, receive feedback, and refine their computational thinking skills. Throughout the robotics programming course, students are encouraged to reflect on their learning process, evaluate their progress, and make adjustments as needed. This reflective practice helps students to internalise their learning, identify areas for improvement, and develop a deeper understanding of computational thinking.

By grounding this study in the Constructionism Theory, the researcher can explore how the active, project-based, and collaborative nature of robotics programming influences the development of computational thinking skills among computer education students. This theoretical framework not only provides a basis for understanding the learning process but also highlights the importance of hands-on, meaningful activities in achieving educational outcomes

2. Methodology

2.1. Research Design

This study adopts a correlational research design to investigate the relationship between the dimensions of computational thinking and achievement in robotics programming among computer education students. The correlational research design is a type of non-experimental research method used to measure the relationship between two or more variables without manipulating them (Hassan, 2024). This design involves collecting data on the variables of interest and using statistical techniques to determine the strength and direction of the relationships between them. The design is appropriate for this study because it effectively addresses the research objectives of exploring and quantifying the relationships between computational thinking dimensions and achievement in robotics programming, without the need for experimental manipulation.

2.2. Participants

The participants of the study were computer education students from universities and degree-awarding institutions in southeast Nigeria who had undergone an 8-week robotics programming course using Arduino (CRE 252 – Robotics Programming II). The population comprise year-two computer and

robotics education students from the University of Nigeria, Nsukka, Enugu State (36), Alvan Ikoku Federal University of Education, Owerri, Imo State (41) and Nwafor Orizu College of Education, Nsugbe, Anambra State (28). Three intact classes of 105 students were purposively selected based on their enrolment in the course..

2.3. Instruments for Data Collection

Two instruments were used for data collection thus: the Computational Thinking Assessment Questionnaire (CTAQ) and the Robotics Programming Achievement Test (RPAT). The CTAQ was adapted from (Tsai, Liang & Hsu, 2021). It was a 19-item instrument developed to assess the computational thinking abilities of students across five dimensions: abstraction, decomposition, algorithmic thinking, evaluation, and generalisation. The CTAQ was rated on a 5-point Likert scale ranging from 1=Not Agree at All, 2=Not Agree, 3=Undecided, 4=Agree, to 5=Totally Agree. The initial instrument has a reliability index of 0.91, while the adapted version yields a Cronbach's alpha coefficient of 0.87 through trial testing. The RPAT measured students' achievement in the 8-week robotics programming course. It comprised 25 multiple-choice questions on Arduino programming concepts, problem-solving, and implementation. The validity of the RPAT was ensured through expert review, and its reliability was confirmed with a Kuder-Richardson formula 20 (KR-20) coefficient of 0.82. A test blueprint (Table of Specifications) guided the construction of the items on RPAT.

2.4. Procedure

The study was conducted over 10 weeks. The first 8 weeks were dedicated to teaching robotics programming with Arduino, during which students were engaged in hands-on activities and guided practice sessions. At the end of the course, the CTAQ and RPAT were administered to the students. Data collection was supervised by the researcher and trained assistants to ensure accuracy and consistency.

2.5. Data Analysis

The collected data were analysed using Pearson's Moment correlation coefficient (PPMC) to determine the relationships between each computational thinking dimension and achievement in robotics programming. Multiple regression analysis was employed to assess the extent to which the dimensions of computational thinking collectively predicted students' achievement. For decision-making, a value close to 1 indicates a strong positive correlation, a value close to -1 indicates a strong negative correlation and a value close to 0 indicates no correlation. The correlation is statistically significant if $p < 0.05$.

3. Results

3.1. Research Question One

What is the relationship between each dimension of computational thinking (abstraction, decomposition, algorithmic thinking, evaluation, and generalisation) and achievement in robotics programming among computer education students in Southeast Nigeria?

The correlation matrix in Table 1 shows the relationships between computational thinking (CT) dimensions and achievement. It was found that Algorithmic Thinking has a strong positive correlation with achievement ($r = .596$, $p < .001$), suggesting that students who excel in algorithmic thinking tend to achieve higher in robotics programming. Evaluation also shows a significant positive correlation ($r = .449$, $p < .001$). Abstraction ($r = .374$, $p < .001$) and Generalization ($r = .416$, $p < .001$) have moderate positive correlations with achievement. Decomposition, however, has a negative correlation with achievement ($r = -.392$, $p<.001$), implying that higher decomposition skills may be associated with lower achievement in robotics programming. These findings indicate that most dimensions of computational thinking have a positive influence on achievement, except for decomposition, which appears to have an inverse relationship.

Test	Dimension	Achievement	Abstraction	Decomposition	Algorithmic thinking	Evaluation	Generalisation
Pearson Correlation	Achievement	1.000	.374	-.392	.596	.449	.416
	Abstraction	.374	1.000	.165	.508	.591	.380
	Decomposition	-.392	.165	1.000	.253	.177	.016
	Algorithmic Thinking	.596	.508	.253	1.000	.867	.716
	Evaluation	.449	.591	.177	.867	1.000	.758
	Generalisation	.416	.380	.016	.716	.758	1.000
Sig. (1-tailed)	Achievement	.	<.001	<.001	<.001	<.001	<.001
	Abstraction	.000	.	.047	.000	.000	.000
	Decomposition	.000	.047	.	.005	.036	.435
	Algorithmic Thinking	.000	.000	.005	.	.000	.000
	Evaluation	.000	.000	.036	.000	.	.000
	Generalisation	.000	.000	.435	.000	.000	.

Table 1. Correlation Matrix of the Relationship Between Dimensions of Computational Thinking and Achievement in Robotics Programming (*Number of Respondents [N] = 105*)

3.2. Research Question Two

How do the dimensions of computational thinking collectively predict achievement in robotics programming among computer education students?

Model	r	r square	Adjusted r square	Std. error of the estimate	Change statistics				
					r square change	f change	df1	df2	Sig. f change
1	.863a	.745	.732	8.43863	.745	57.845	5	99	<.001

a. Predictors: (Constant), Generalisation, Decomposition, Abstraction, Algorithmic Thinking, Evaluation

Table 2. Model Summary on How the Dimensions of Computational Thinking Collectively Predict Achievement in Robotics Programming

Model		Sum of squares	df	Mean square	f	Sig.
1	Regression	20595.728	5	4119.146	57.845	<.001b
	Residual	7049.834	99	71.210		
	Total	27645.562	104			

a. Dependent Variable: Achievement

b. Predictors: (Constant), Generalisation, Decomposition, Abstraction, Algorithmic Thinking, Evaluation

Table 3. Analysis of Variance (ANOVA) on How the Dimensions of Computational Thinking Collectively Predict Achievement in Robotics Programming

The model summary in Table 2 and the ANOVA in Table 3 show that the five dimensions of computational thinking collectively explain 74.5% of the variance in students' achievement (r^2 = .745). which is statistically significant (F(5.99)=57.845, p < .001). This indicates that computational thinking dimensions. when considered together, are strong predictors of robotics programming achievement. The standard error of the estimate (8.439) suggests a relatively good fit of the model.

3.3. Research Question Three

Which dimension of computational thinking is the strongest predictor of achievement in robotics programming among computer education students?

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	114.593	9.204		12.450	<.001
	Abstraction	3.836	1.110	.220	3.457	<.001
	Decomposition	-22.573	1.941	-.630	-11.627	<.001
	Algorithmic Thinking	20.332	1.943	1.116	10.467	<.001
	Evaluation	-9.565	2.536	-.449	-3.772	<.001
	Generalisation	-2.345	1.658	-.116	-1.415	.160

Table 4. Summary of Coefficients of Regression on Which Dimension of Computational Thinking is the Strongest Predictor of Achievement in Robotics Programming

Table 4 shows the standardised beta coefficients in the regression output to help identify the strongest predictor of achievement in robotics programming among computer education students. It was found that Algorithmic Thinking has the highest positive standardised beta coefficient ($\beta = 1.116$, $p < .001$), indicating that it is the strongest predictor of achievement. Abstraction is also a significant positive predictor ($\beta = .220$, $p < .001$). Evaluation has a significant negative effect ($\beta = -.449$, $p < .001$), suggesting that higher evaluation skills might unexpectedly contribute to lower achievement. Decomposition shows a strong negative impact on achievement ($\beta = -.630$, $p < .001$), reinforcing its negative correlation. Generalisation is not a significant predictor ($p = .160$), meaning its contribution to achievement is minimal. Overall, Algorithmic Thinking emerges as the most potent positive predictor, while Decomposition negatively impacts achievement.

3.4. Research Question Four

How do students' perceptions of their computational thinking abilities correlate with their actual performance in robotics programming?

To answer this research question, the results in Table 1 were carefully explored. The results suggest that students' self-perceptions of their computational thinking skills (as indicated by abstraction, decomposition, algorithmic thinking, evaluation, and generalisation) correlate with their actual performance in varying degrees: Positive correlations with achievement suggest that students who perceive themselves as proficient in algorithmic thinking, evaluation, abstraction, and generalisation tend to perform well in robotics programming. However, the negative correlation between decomposition and achievement ($r = -.392$, $p < .001$) suggests that students who believe they are good at decomposition may not necessarily achieve high scores, or they may struggle to apply decomposition effectively in robotics programming. These findings indicate that while most self-perceived CT skills align with actual performance, decomposition stands out as an area where perceptions and performance do not align positively.

4. Discussions

In this section, we discussed the findings of this study by suggesting plausible reasons for the findings and comparing these findings with existing literature.

4.1. Relationship Between Dimensions of Computational Thinking and Achievement in Robotics Programming

The results in Table 1 show that algorithmic thinking has the strongest positive correlation with achievement in robotics programming. This suggests that students who excel in algorithmic thinking tend to perform better in robotics programming. A reasonable explanation for this finding is that algorithmic thinking allows students to break problems down into logical steps, come up with efficient solutions, and write code to implement them, which are all important skills for programming robots. This aligns with existing literature, as studies by Román-González, Pérez-González and Jiménez-Fernández (2017) and

Grover and Pea (2018) have also identified algorithmic thinking as a strong predictor of success in programming-related tasks. Since robotics programming requires iterative problem-solving, debugging, and optimisation, students with strong algorithmic thinking skills will likely adapt more effectively and achieve higher outcomes.

Similarly, evaluation shows a significant positive correlation with achievement, indicating that students who can assess and refine their work effectively tend to perform better. This is not surprising, as evaluation skills are essential for debugging, testing, and improving program efficiency. Brennan and Resnick (2012) emphasised that evaluation plays a key role in computational thinking, as it enables students to recognise errors and optimise their solutions. Shute et al. (2017) also highlighted evaluation as a crucial skill in computational problem-solving, reinforcing its importance in robotics programming. The study further found that abstraction and generalisation have moderate positive correlations with achievement. Abstraction allows students to focus on essential details while ignoring irrelevant complexities, which is particularly beneficial in structuring and modularising code. Generalisation, on the other hand, enables students to recognise patterns and apply learned problem-solving approaches to new challenges. Research by Lye and Koh (2014) and Weintrop, Beheshti, Horn, Orton, Jona, Trouille et al. (2016) has similarly indicated that students with strong abstraction and generalisation skills tend to perform well in programming tasks, as these skills support adaptive thinking and code reusability.

However, an unexpected finding was the negative correlation between decomposition and achievement. Decomposition, which involves breaking a problem into smaller, manageable sub-problems, is typically considered a fundamental computational thinking skill. However, in this study, higher decomposition skills were associated with lower achievement in robotics programming. One plausible explanation is that excessive focus on decomposition may lead to difficulty in integrating the smaller components into a coherent whole. When students decompose problems too rigidly, they might struggle to synthesise solutions efficiently, leading to fragmented or overly complex code structures. Grover and Pea (2018) also observed that novice programmers sometimes face challenges in applying decomposition effectively, as they may overcomplicate problems rather than simplifying them. Lye and Koh (2014) further noted that while decomposition is beneficial, its effectiveness depends on the student's ability to reconstruct the broken-down components into a functional solution.

Therefore, these findings align with previous research that emphasises algorithmic thinking as the most critical computational thinking skill for programming success (Román-González et al., 2017; Grover & Pea, 2018). The positive influence of evaluation, abstraction, and generalisation is also supported by existing literature (Weintrop et al., 2016; Shute et al., 2017). However, the negative correlation of decomposition contrasts with conventional computational thinking models (Brennan & Resnick, 2012; Lye & Koh, 2014), suggesting that decomposition may not always be beneficial, particularly if students struggle to integrate fragmented components into a complete program.

4.2. How the Dimensions of Computational Thinking Collectively Predict Achievement in Robotics Programming

The results in Tables 2 and 3 show that the five dimensions of computational thinking such as abstraction, decomposition, algorithmic thinking, evaluation, and generalisation, collectively explain 74.5% of the variance in students' achievement in robotics programming. This indicates that computational thinking strongly predicts success in robotics programming. The relatively low standard error of the estimate further suggests that the model fits the data well, meaning that these dimensions, when considered together, effectively predict students' achievement. One plausible explanation for this finding is that robotics programming inherently requires the simultaneous application of multiple computational thinking skills. Students must abstract key problem components, decompose complex tasks into manageable parts, apply algorithmic thinking to structure logical sequences, use evaluation to debug and optimise solutions, and leverage generalisation to apply learned concepts across different problems. This interconnectedness among the CT dimensions likely contributes to the strong predictive power observed in the study. Another reason for the high variance explained by computational thinking could be that CT

skills collectively enhance cognitive flexibility, a crucial trait in programming. Algorithmic thinking and evaluation, for instance, work together to refine problem-solving strategies, while abstraction and decomposition help simplify intricate coding challenges. When students develop these skills holistically, they become more adept at approaching, analysing, and solving robotics programming tasks efficiently, leading to higher achievement levels.

These findings align with previous research emphasising the integral role of computational thinking in programming success. Román-González et al. (2017) found that computational thinking skills explain a significant portion of students' programming proficiency, particularly when these skills are developed collectively. Similarly, Weintrop et al. (2016) highlighted how students with well-developed computational thinking abilities tend to navigate complex coding challenges more effectively, reinforcing the idea that CT is a crucial determinant of programming achievement. Grover and Pea (2018) further argue that computational thinking should be viewed as a cognitive framework rather than a set of isolated skills. Grover and Pea emphasise that students who develop computational thinking holistically are better equipped to tackle programming challenges, debug code, and optimise solutions. This perspective supports the current study's findings that CT dimensions, when considered together, provide a strong predictive model for robotics programming success.

However, some studies suggest that the predictive power of computational thinking skills may vary depending on instructional methods, prior knowledge, and motivation. Lye and Koh (2014) found that while CT significantly influences programming performance, its effectiveness is sometimes moderated by students' prior experience and the nature of the programming tasks. This implies that while computational thinking is a key predictor, other factors such as teaching approaches, student engagement, and external support systems may also shape students' achievement in robotics programming. While the current study finds that CT dimensions collectively explain 74.5% of the variance in achievement, other researchers argue that non-cognitive factors should not be overlooked (Shute et al., 2017). Shute et al. highlighted the role of self-efficacy, motivation, and collaboration in shaping programming success. These factors, though not explicitly included in the current model, may contribute to the remaining unexplained variance in students' robotics programming achievement.

4.3. The Dimension of Computational Thinking Which is The Strongest Predictor of Achievement in Robotics Programming Among Computer Education Students

The results of this study in Table 4 show that algorithmic thinking is the strongest predictor of achievement in robotics programming, while decomposition and evaluation have significant negative effects. Abstraction is also a significant positive predictor, whereas generalisation does not significantly predict achievement. The dominance of algorithmic thinking as the strongest predictor of achievement can be explained by its direct relevance to programming logic and problem-solving. Robotics programming requires students to construct logical sequences of instructions that a robot can execute efficiently. Students with strong algorithmic thinking skills are better equipped to develop structured and efficient solutions, leading to improved performance. This finding is consistent with Wing's (2006) assertion that algorithmic thinking is a fundamental skill in computational problem-solving, as it enables individuals to break down problems into a sequence of well-defined steps.

Also, the positive predictive power of abstraction can be attributed to its role in simplifying complex problems and focusing on essential details. In robotics programming, students must manage multiple components, including sensor inputs, control logic, and sequential execution of tasks. Those proficient in abstraction can efficiently filter out irrelevant details and concentrate on the core structure of a problem, enhancing their ability to design effective programs. This aligns with Brennan and Resnick's (2012) study, which highlights abstraction as a key computational thinking skill that enables students to handle complexity and design robust programming solutions. On the other hand, the negative impact of decomposition on achievement is unexpected but may suggest that students struggle with reintegrating decomposed components into a cohesive whole. While decomposition is generally considered a beneficial computational thinking skill, its effectiveness depends on how well students can reassemble broken-down

problems into functional solutions. If students focus too much on breaking down a problem without successfully synthesising the components, they may experience difficulty in structuring their programs efficiently. Grover and Pea (2018) similarly note that while decomposition is useful for problem-solving, improper or excessive decomposition without reintegration can lead to cognitive overload and inefficient programming.

Furthermore, the negative relationship between evaluation and achievement may stem from the possibility that students who spend excessive time debugging or overanalysing their code struggle to complete their programming tasks efficiently. Evaluation is crucial for identifying errors and optimising code, but if students become overly focused on debugging without confidence in their corrections, they may become trapped in iterative cycles of problem identification and revision. This finding resonates with Shute et al. (2017), who emphasise that students who lack structured debugging strategies often experience frustration, leading to lower overall performance in programming tasks. While generalisation's non-significant contribution to achievement ($p = .160$) suggests that transferable problem-solving skills may not directly impact robotics programming performance. Generalisation is an essential computational thinking skill that allows students to apply previously learned knowledge to new problems. However, robotics programming often involves highly specific, context-dependent problem-solving approaches, where generalised knowledge may not always be applicable. This finding is consistent with Weintrop et al. (2016), who argue that while generalisation is an important computational thinking skill, its impact varies depending on the similarity between previously encountered problems and new tasks.

The strong predictive power of algorithmic thinking aligns with several studies that emphasise its role in programming success. For instance, Román-González et al. (2017) found that students with strong algorithmic thinking abilities consistently outperform their peers in programming tasks, reinforcing the notion that algorithmic thinking is essential for effective coding. Similarly, Lye and Koh (2014) highlight algorithmic thinking as the backbone of programming education, as it enables students to break problems into logical sequences and implement solutions systematically. However, the negative relationship between decomposition and achievement disagrees with some studies that view decomposition as a critical computational thinking skill. Brennan and Resnick (2012), for example, emphasise decomposition as an essential process in programming. However, this study's findings suggest that if students focus too much on breaking down a problem without properly integrating the components, they may struggle with task completion. Grover and Pea (2018) caution that decomposition can be counterproductive if students are not guided on how to synthesise decomposed parts into complete solutions, which supports the current study's findings. The unexpected negative impact of evaluation also contrasts with studies that highlight its role in debugging and code refinement (Shute et al., 2017). However, it aligns with research suggesting that students who struggle with debugging may experience frustration, leading to decreased motivation and lower achievement (Lye & Koh, 2014). If students lack strategic approaches to debugging, evaluation may become a time-consuming, trial-and-error process rather than a structured skill that enhances performance.

4.4. How Students' Perception of their Computational Thinking Abilities Correlates with their Actual Achievement in Robotics Programming

Results from Table 1 suggest that students' self-perceptions of their computational thinking (CT) abilities generally align with their actual performance in robotics programming, except for decomposition, which exhibits a negative correlation with achievement. Specifically, students who believe they are proficient in algorithmic thinking, evaluation, abstraction, and generalisation tend to perform well in robotics programming, while those who perceive themselves as strong in decomposition do not necessarily achieve high scores. The positive relationship between students' perceived abilities in algorithmic thinking, abstraction, evaluation, and generalisation and their actual performance can be attributed to the confidence-competence loop. When students believe they are skilled in a particular area, they are more likely to engage with the material actively, apply their knowledge effectively, and persist through challenges,

leading to better performance. Bandura and Wessels's (1997) self-efficacy theory supports this, arguing that individuals who have confidence in their abilities are more likely to succeed because they engage more deeply with tasks and demonstrate higher persistence.

The positive correlation between self-perceived evaluation skills and achievement can be attributed to the role of debugging and code refinement in programming. Students who perceive themselves as proficient in evaluating and improving their code are likely to perform better, as effective debugging is a crucial aspect of robotics programming. Shute et al. (2017) found that students with strong debugging self-efficacy were more likely to persist in troubleshooting their code, leading to better programming outcomes. Similarly, students who perceive themselves as good at abstraction and generalisation tend to achieve higher scores. This is likely because abstraction allows them to focus on essential problem components without getting overwhelmed by details, and generalisation helps them apply previous knowledge to new programming challenges. This finding aligns with Grover and Pea (2018), who assert that abstraction and generalisation are critical for effective computational problem-solving, as they help students create reusable code structures and adapt to different programming scenarios.

However, the negative correlation between decomposition and achievement suggests that students who believe they are good at decomposition may struggle with applying it effectively in robotics programming. This may be because decomposition requires not only breaking down a problem into smaller parts but also successfully integrating these parts into a coherent solution. If students overemphasise breaking problems apart without successfully synthesising solutions, their programming performance may suffer. Weintrop et al. (2016) argue that while decomposition is an essential skill, its effectiveness depends on students' ability to recombine decomposed elements into functional programs. Similarly, Grover and Pea (2018) note that decomposition alone does not guarantee programming success; it must be coupled with synthesis and integration skills. Another possible reason for the misalignment between self-perceived decomposition skills and actual achievement could be cognitive overload. When students break down a problem into too many components, they might struggle to track dependencies and relationships between different parts of the program, leading to confusion and reduced performance. Lye and Koh (2014) highlight that students who lack experience in reintegrating decomposed parts often face difficulties in programming, as they may lose sight of the overall goal.

5. Conclusions

This study explored the relationship between computational thinking (CT) dimensions and achievement in robotics programming among computer education students in Southeast Nigeria. The results show that algorithmic thinking is the strongest predictor of achievement, emphasising its crucial role in structuring and implementing logical problem-solving approaches in programming. Other dimensions, such as abstraction, evaluation, and generalisation, also show positive correlations with achievement, reinforcing their importance in developing computational proficiency. However, decomposition exhibits a negative relationship with achievement, suggesting that while breaking down problems is essential, students may struggle with integrating decomposed elements into cohesive solutions. Furthermore, the five CT dimensions collectively explained the variance in robotics programming achievement, demonstrating that computational thinking plays a vital role in programming success. Additionally, students' self-perceptions of their computational thinking abilities align with their actual performance in most dimensions, except for decomposition, where a misalignment exists.

6. Recommendations

The following recommendations were made based on the findings of the study:

1. Since algorithmic thinking has been found to have a strong predictive power, educators should integrate more algorithm-focused exercises, such as structured coding challenges and debugging tasks, into the curriculum to enhance logical problem-solving skills.

2. Decomposition was found to negatively correlate with achievement. Therefore, instructional methods should emphasise effective problem breakdown and reconstruction through guided exercises, scaffolded learning, and project-based tasks.
3. Practical and collaborative learning should be promoted to strengthen abstraction, evaluation, and generalisation. Hands-on projects, peer programming, and real-world applications should be incorporated to reinforce computational thinking in robotics programming.

7. Future Studies

This study offers significant insights into the predictive influence of computational thinking on students' performance in a project-based robotics programming course; however, a number of limitations were acknowledged:

1. The sample size (N=105) is relatively small and comes from just one region (Southeast Nigeria), which may make it hard to generalise the results to other regions of Nigeria or other educational systems.
2. The study utilised a correlational design, which facilitates identification of relationships among variables but does not confirm causation; hence, caution is warranted in interpreting the directionality of the observed interactions.
3. While validated instruments were utilised for measuring computational thinking, some of the assessments depended on how students perceived themselves, which could be affected by biases like wanting to look good or not knowing much about themselves.
4. The findings underscore the negative impact of decomposition difficulties on students' achievement. Subsequent research should investigate whether structured training in decomposition strategies may mitigate this issue and improve the ability of learners to effectively integrate decomposed components into functional robotic systems.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Ethics Statement

This study was approved by the Ethics Committee of the University of Nigeria, Nsukka, and informed consent was obtained from all participants prior to data collection.

Funding

This study was funded by ETDP-SETA researcher Chair in Mathematics Education at the University of the Free State, reference number UFS-AGR22-000053.

References

Acevedo-Borrega, J., Valverde-Berrocoso, J., & Garrido-Arroyo, M.D.C. (2022). Computational thinking and educational technology: A scoping review of the literature. *Education Sciences,* 12(1), 39. https://doi.org/10.3390/educsci12010039

Agbo, F.J., Okpanachi, L.O., Ocheja, P., Oyelere, S.S., & Sani, G. (2024). How can unplugged approach facilitate novice students' understanding of computational thinking? An exploratory study from a Nigerian university. *Thinking Skills and Creativity*, 51, 101458. https://doi.org/10.1016/j.tsc.2023.101458

Aranda, G., & Ferguson, J.P. (2022). The Creative in Computational Thinking. In *Children's Creative Inquiry in STEM* (309-326). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-94724-8_18

Bandura, A., & Wessels, S. (1997). *Self-efficacy* (4-6). Cambridge: Cambridge University Press.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (1, 25). Available at: https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Chuang, Y.T., & Chang, H.Y. (2024). Analyzing novice and competent programmers' problem-solving behaviors using an automated evaluation system. *Science of Computer Programming,* 237, 103138.

Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the constructionist learning theory with computational thinking classroom activities. *Informatics in Education,* 18(1), 41-67. Available at: https://www.ceeol.com/search/article-detail?id=761641

Denning, P.J. (2009). The profession of IT Beyond computational thinking. *Communications of the ACM,* 52(6), 28-30. https://doi.org/10.1145/1516046.1516054

Gizzi, E., Nair, L., Chernova, S., & Sinapov, J. (2022). Creative problem solving in artificially intelligent agents: A survey and framework. *Journal of Artificial Intelligence Research,* 75, 857-911. https://doi.org/10.1613/jair.1.13864

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher,* 42(1), 38-43. https://doi.org/10.3102/0013189X12463051

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. Computer science education: *Perspectives on teaching and learning in school,* 19(1), 19-38.

Hassan, M. (2024). Correlational Research – Methods, Types and Examples. *Strategies, Processes & Techniques utilized in the collection of data.* Available at: https://researchmethod.net/correlational-research/

Kert, S.B., Erkoç, M.F., & Yeni, S. (2020). The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. *Thinking Skills and Creativity,* 38, 100714.

Kwon, K., & Cheon, J. (2019). Exploring Problem Decomposition and Program Development through Block-Based Programs. *International Journal of Computer Science Education in Schools,* 3(1), 1. Available at: https://eric.ed.gov/?id=EJ1214684

Lye, S.Y., & Koh, J.H.L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior,* 41, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Nannim, F.A., Ibezim, N.E., Oguguo, B.C., & Nwangwu, E.C. (2024). Effect of project-based Arduino robot application on trainee teachers computational thinking in robotics programming course. *Education and Information Technologies*, 29(10), 13155-13170. https://doi.org/10.1007/s10639-023-12380-6

Nannim, F.A., Ibezim, N.E., Mosia, M., & Oguguo, B.C. (2025). Project-Based Learning with Arduino Robots: Impact on Undergraduate Students' Achievement and Task Persistence in Robotics Programming. *Frontiers in Robotics and AI*, 12, 1615427. https://doi.org/10.3389/frobt.2025.1615427

Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational technology research and development,* 68(1), 463-484. https://doi.org/10.1007/s11423-019-09708-w

Palop, B., Díaz, I., Rodríguez-Muñiz, L.J., & Santaengracia, J.J. (2025). Redefining computational thinking: A holistic framework and its implications for K-12 education. *Education and Information Technologies*, 30, 13385-13410. https://doi.org/10.1007/s10639-024-13297-4

Papert, S.A. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic books.

Papert, S.A (1993). *The children's machine: Rethinking school in the age of the computer.* Basic Books, Inc.

Rao, T.S.S., & Bhagat, K.K. (2024). Computational thinking for the digital age: a systematic review of tools, pedagogical strategies, and assessment practices. *Educational technology research and development,* 72, 1893-1924. https://doi.org/10.1007/s11423-024-10364-y

Román-González, M., Pérez-González, J.C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior,* 72, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Rovshenov, A., & Sarsar, F. (2023). Research trends in programming education: A systematic review of the articles published between 2012-2020. *Journal of Educational Technology and Online Learning,* 6(1), 48-81. https://doi.org/10.31681/jetol.1201010

Sabuncuoglu, A., & Sezgin, T.M. (2022). Kart-on: An extensible paper programming strategy for affordable early programming education. *Proceedings of the ACM on Human-Computer Interaction,* 6(EICS), 1-18. https://doi.org/10.1145/3534524

Selby, C., & Woollard, J. (2014). *Computational thinking: The developing definition.* University of Southampton. Available at: http://eprints.soton.ac.uk/id/eprint/356481

Shute, V.J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational research review,* 22, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Talan, T. (2021). The Effect of Educational Robotic Applications on Academic Achievement: A Meta-Analysis Study. *International Journal of Technology in Education and Science,* 5(4), 512-526. Available at: https://files.eric.ed.gov/fulltext/EJ1323488.pdf

Tsai, M. J., Liang, J. C., & Hsu, C. Y. (2021). The computational thinking scale for computer literacy education. *Journal of Educational Computing Research,* 59(4), 579-602. https://doi.org/10.1177/0735633120972356

Waite, J., Curzon, P., Marsh, W., & Sentance, S. (2020). Difficulties with design: The challenges of teaching design in K-5 programming. *Computers & education,* 150, 103838. https://doi.org/10.1016/j.compedu.2020.103838

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of science education and technology,* 25, 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J.M. (2006). Computational thinking. *Communications of the ACM,* 49(3), 33-35. https://doi.org/10.1145/1118178.1118215

Yıldız, T., & Seferoğlu, S.S. (2021). The effect of robotic programming on coding attitude and computational thinking skills toward self-efficacy perception. *Journal of Learning and Teaching in Digital Age,* 6(2), 101-116. Available at: https://files.eric.ed.gov/fulltext/EJ1308356.pdf

Published by OmniaScience (www.omniascience.com)

Journal of Technology and Science Education, 2026 (www.jotse.org)