ISSN: 2473-4901
v12 n6832

Teaching Quantum Computing Without Quantum Hardware: A Hands-on Learning Module for Undergraduate Computing Education

John D. Crabtree
jcrabtree@una.edu
University of North Alabama
Florence, AL 35632, USA

Xihui "Paul" Zhang
xzhang6@una.edu
University of North Alabama
Florence, AL 35632, USA

Qiunan Zhang
qzhang@alasu.edu
Alabama State University
Montgomery, AL 36104, USA

Abstract

Quantum computing is an emerging technology with the potential to significantly transform computing, cybersecurity, and data processing. However, limited access to quantum hardware and the complexity of underlying concepts present challenges for integrating quantum computing into undergraduate computing curricula. This study presents a hands-on learning module that introduces quantum computing concepts in a *Software Architecture* course without requiring access to physical quantum hardware. The module combines assigned readings, quizzes, lectures, and guided programming assignments, using a Java-based quantum computing library that runs on simulators and remains compatible with real quantum systems. To evaluate the module's effectiveness, the study collected quantitative and qualitative data from undergraduate students in the Computer Information Systems, Computer Science, and Information Technology majors. A paired-samples t-test (N = 104) revealed a statistically significant improvement in students' understanding of quantum computing concepts from pre-test to post-test. Survey results further indicated substantial gains in students' perceived knowledge of build tools, algorithmic complexity, and quantum computing. Thematic analysis of open-ended responses showed that most students valued the module for its future relevance, introductory exposure, and hands-on experience, despite challenges related to conceptual difficulty, programming, and instructional pacing. These findings suggest that simulator-based, hands-on approaches can effectively introduce quantum computing in undergraduate computing education and help bridge the gap between theory and practice. Implications for curriculum design and instructional strategies are discussed.

Keywords: Quantum computing, Quantum hardware, Computing education, Hands-on learning, Software architecture, Higher education.

ISSN: 2473-4901
v12 n6832

Teaching Quantum Computing Without Quantum Hardware: A Hands-on Learning Module for Undergraduate Computing Education

John D. Crabtree, Xihui "Paul" Zhang, & Qiunan Zhang

1. INTRODUCTION

While quantum computers are not yet practical for everyday use, their potential impact promises to be groundbreaking and disruptive. In the future, when quantum algorithms can be executed at scale, many problems currently considered intractable may become efficiently solvable. On the other hand, algorithms that currently safeguard sensitive information may no longer do so effectively. The National Institute of Standards and Technology (NIST) has recognized this issue and finalized three new quantum-safe encryption standards (i.e., FIPS 203, FIPS 204, and FIPS 205) following an eight-year selection process (NIST, 2024). "These standards specify key establishment and digital signature schemes that are designed to resist future attacks by quantum computers, which threaten the security of current standards" (NIST, 2024). Because of these potential benefits and risks (as well as unforeseen impacts), governments worldwide are investing large sums to "push the boundaries of what is possible with the technology" (Peel, 2025).

Preparing students in computing fields for the upcoming changes brought about by quantum computers will become increasingly important as researchers continue to improve the hardware. Given the interdisciplinary nature of Information Systems, it is especially important to equip students with the background and knowledge necessary to address these changes (Arai et al., 2021). Undergraduate students do not need to wait for this hardware to become widely available. Today, there are many ways to gain hands-on experience with quantum computing systems without the expense of purchasing state-of-the-art infrastructure. Hands-on learning is an instructional approach in which students actively apply knowledge and skills through practical activities, problem solving, and direct engagement with real-world or simulated tasks (Knobloch & Smith, 2024). In IS education, hands-on activities can include database development, programming, data analytics, GIS projects, and other technology-based projects.

We present a learning module in our Software Architecture course that gives our students hands-on experience with basic quantum computing algorithms. Our students implement these algorithms using code libraries that can be configured to overlay both quantum computers and simulated environments. Code developed and executed in the simulator can also run on quantum computing hardware when it becomes available.

Quantum computers are not simply better or faster versions of today's computers; neither will they replace today's hardware. Rather, quantum computers operate on a completely different set of principles, theoretically grounded in quantum mechanics. This fundamental branch of physics, which deals with matter and energy at the smallest scales, is not a skill most computing professionals possess and is not a typical field of study within any undergraduate computing discipline.

Digital computers use integrated circuits that contain transistors. Quantum mechanics plays a crucial role in the operation of transistors and related devices such as flash memory. The key to using this hardware, which has been around for decades, is to create useful abstractions that insulate most users from the complexity at the lowest levels. A similar approach will likely be used with quantum computers to enable broader adoption.

There are new concepts related to quantum computers that everyone will need to learn, but they do not require a deep understanding of the underlying physics. Just as digital computers work with bits, quantum computers work with qubits. Unlike bits, qubits are characterized by three numbers: the probability of measuring 0, the probability of measuring 1, and their phase. Also, in contrast to bits, there is no way to copy qubits since reading (or measuring) a qubit is a destructive, non-repeatable activity. In addition, both probabilities within a qubit may be non-zero, meaning that the qubit is in superposition. When a qubit is measured, it will always produce either 0 or 1, but which value a qubit in superposition will produce cannot be known in advance. Reading or measuring a qubit is not an error-free activity. Avoiding these measurement errors is one of the most difficult remaining problems in quantum computing.

Quantum computers also employ "gates" like the logical NOT. In fact, this gate works just like the digital analog: the probability of a qubit being 1 is "flipped" to the probability of being 0, and vice versa. A Hadamard gate creates equal probabilities of 0 and 1, thus placing the qubit into a state of superposition. Our textbook (i.e., *Software Architecture in Practice*, 4th edition, by Bass et al., 2022) covers many other gates and important aspects of the technology, such as entanglement and quantum teleportation. However, the concepts above are the most important ones for our hands-on labs.

2. CURRENT STATUS OF QUANTUM COMPUTING EDUCATION

A growing body of literature suggests that information technology is fundamentally transforming both social and economic systems. Padmaavathy (2023) argues that emerging technologies such as nanotechnology, biometrics, photonic computing, and quantum computing are reshaping organizational capabilities, particularly in data analysis and business operations. These technologies expand the ways organizations process and interpret information, enabling new forms of efficiency and innovation in business practice.

Extending this broader transformation, Dulababu (2025) highlights that artificial intelligence (AI), quantum computing, and Industry 4.0 technologies are collectively reshaping the job market. These technologies are changing how work is performed, redefining required skill sets, and altering organizational structures. While these technologies enable automation, enhance decision-making, and support data-driven innovation, they also increase pressure on workers and institutions to continuously adapt to rapidly evolving technological environments.

Within higher education, universities are increasingly responding to these technological shifts by developing collaborative ecosystems to support quantum computing research and training. Nott (2018) reports on the University of Sydney–led establishment of the Sydney Quantum Academy, in partnership with Macquarie University, the University of Technology Sydney, and the University of New South Wales, and supported by the New South Wales government. This initiative aims to strengthen postgraduate education, advance interdisciplinary research, and translate quantum computing expertise into workforce development and job creation, reflecting the growing importance of regional innovation clusters in emerging technologies.

At the same time, research on quantum computing education highlights both opportunities and challenges in curriculum integration. Economou and Barnes (2022) emphasize that introductory quantum computing instruction benefits from reducing heavy mathematical prerequisites and adopting more intuitive, conceptually grounded approaches to topics such as superposition and entanglement. Similarly, Kushimo and Thacker (2022) find that students commonly struggle with abstract quantum concepts, particularly entanglement and quantum measurement, suggesting the need for evidence-based instructional strategies and carefully designed learning materials.

Complementing these findings, Farooq and Upadhyay (2025) show that hands-on, lab-based learning environments, such as circuit simulators and IBM Qiskit, can improve student understanding of quantum computing. However, they also note that students continue to face difficulties with probabilistic reasoning and debugging quantum programs, highlighting the importance of scaffolded learning approaches that gradually build both conceptual and computational competence. More broadly, recent mapping studies of quantum computing education indicate that while interest in the field is rapidly increasing, curriculum integration remains fragmented across institutions, underscoring the need for more structured educational frameworks and scalable models of quantum literacy (López-Meneses et al., 2025). Similarly, systematic reviews of early education contexts suggest that quantum computing education is expanding beyond universities but faces significant barriers due to its abstract nature and lack of mature pedagogical models (Yusufa et al., 2025).

Taken together, these studies suggest that quantum computing should be understood as part of a broader technological transformation driven by AI and Industry 4.0, while also representing a distinct educational challenge for higher education institutions. At the macro level, it is reshaping work, skills, and organizational structures (Dulababu, 2025) while also expanding organizational capabilities for data analysis and innovation (Padmaavathy, 2023). At the institutional level, initiatives such as the Sydney Quantum Academy illustrate how universities are building collaborative ecosystems to support training and research (Nott, 2018). At the pedagogical level, however, the literature consistently points to the need for redesigned curricula, more intuitive teaching approaches, and experiential learning environments to support student understanding of quantum computing concepts (Economou & Barnes, 2022; Farooq & Upadhyay, 2025; Kushimo & Thacker, 2022; López-Meneses et al., 2025; Yusufa et al., 2025).

ISSN: 2473-4901
v12 n6832

Overall, the literature indicates that higher education must simultaneously respond to technological disruption, strengthen institutional capacity, and innovate in teaching and learning to prepare students for a quantum-enabled future.

3. TEACHING QUANTUM COMPUTING – THE PEDAGOGY

3.1 Course Background

Our *Software Architecture* course serves three majors: Computer Information Systems (CIS), Computer Science (CS), and Information Technology. CIS and IT students must pass an advanced object-oriented programming course, currently taught in Java, to take *Software Architecture*. Many CS students also enroll in the Java course, but a similar advanced C++ course serves as an alternative prerequisite for CS majors. Given the expectation that most Software Architecture students know Java, we selected an open-source quantum computing platform written in Java.

We employed several instruments and metrics to evaluate the effectiveness of our approach.

3.2 Pre-Test During the Penultimate Week

During the penultimate week of the semester, students are asked to take a pre-test consisting of ten questions (see Appendix A) taken from lecture material planned for the following week, information that should be retained from the hands-on labs the students will complete the following week, as well as material in the final chapter of the textbook. This pre-test is presented as a "survey" to avoid creating the impression that students were being tested on material they should already know at that point in the semester.

3.3 Final Week

Software Architecture in Practice (4th edition) (Bass et al., 2022) is the primary textbook for our Software Architecture course. The final chapter of the book is titled "A Glimpse of the Future: Quantum Computing." This chapter is assigned to be read before our final week of the semester.

The class session begins with a quiz covering the final chapter of the textbook. This ten-question quiz has no overlap with the pre-test (survey) taken the week before. The questions are drawn exclusively from the textbook. Questions that address the same concepts as the pre-test are worded differently and include different answer choices (for multiple-choice or multiple-select questions).

3.4 The First Lecture

The students, who are in their final year of one of our three majors (CIS, CS, or IT), are presented with some popular myths about quantum computing, along with the facts and current state of the technology. A special emphasis is placed on computational complexity through a brief review of Big-O notation and algorithm efficiency. Our computer science students are expected to have more familiarity and experience with this topic than students in other majors. However, all students should have been introduced to the concepts in at least one prerequisite course. We use Shor's algorithm at this point to explain the quantum advantage expected with respect to integer factorization of large numbers. The lecture concludes with a discussion of a possible future architecture (shown in Figure 1) that includes a quantum processing unit (QPU) and cloud-based resources.

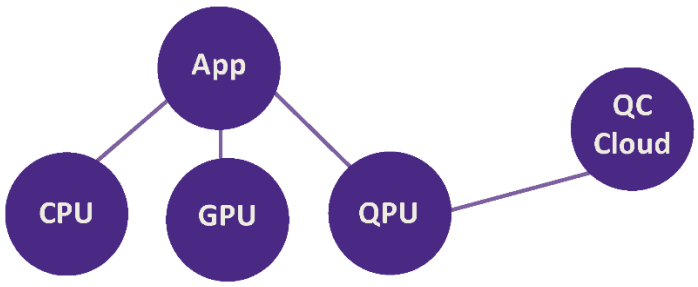

Figure 1. A Possible Future Architecture

ISSN: 2473-4901
v12 n6832

While this module presents an introduction to quantum computing based on our textbook, we also employed open-source software in the form of a Java library named "Strange," which is available via github.com/redfx-quantum/strange. The lead developer of this library, Johan Vos, has also written an excellent book titled *Quantum Computing in Action* (Vos, 2022) that uses this library throughout. In the preface, Vos states that "there was a large gap between the scientific world and the IT world." To do our small part in narrowing that gap, we chose to use his Java library to help us present the textbook's final chapter to our students.

The Strange library has two distinct APIs: a high-level and a low-level (see Figure 2). The first lecture introduces the library and focuses on the high-level API. The underlying implementation could be a localhost simulator (provided by the Strange library) or a cloud-based simulator. In addition, abstraction could allow actual quantum hardware to provide an implementation in the future.

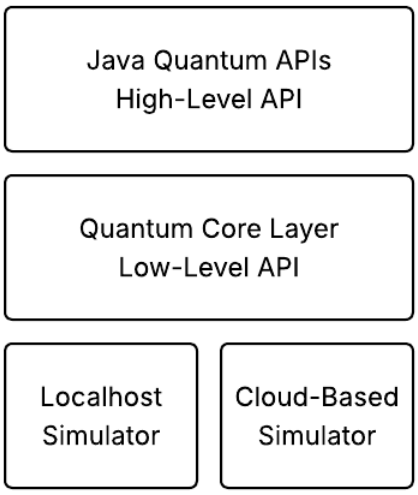

Figure 2. The Architecture of Strange

3.5 High-Level Project Instructions

The first hands-on project requires experience with Git, Maven, Java, and Eclipse. Our students have been using these tools and technologies in the Software Architecture course throughout the semester. This lab tutorial appears in Appendix B. Students must download and install Maven. Once they have added Maven's *bin* directory to the path, they can download Vos' quantum computing equivalent of "hello, world," which, when executed, generates 10,000 random bits and reports the number of ones and zeroes created.

They will then use Git to clone the open-source repository for Strange and build the Java library from the source code, resulting in a jar file using Maven and the pom.xml file included in the Strange source. Next, they will configure Eclipse to use the library they just built, along with the code from the "hello, world" example, to test their library and the environment configuration.

The final step of the lab involves modifying the original code example to produce a bit string of arbitrary length and display (1) the bit string, (2) the corresponding integer value as a decimal number, and (3) the same value in hexadecimal format.

Once the "hello, world" program from the second chapter of the Vos book is modified to meet the requirements above, most students produce a program similar to the example in Listing 1.

ISSN: 2473-4901
v12 n6832

```
import org.redfx.strange.algorithm.Classic;

public class HighLevel {
	public static void main (String[] args) {
		int numBits = 4;
		StringBuffer bitString = new StringBuffer();

		for (int i = 0; i < numBits; i++) {
			if (Classic.randomBit() > 0) {
				bitString.append("1");
			} else {
				bitString.append("0");
			}
		}

		System.out.println(bitString.toString());
		int value = Integer.parseInt(bitString.toString(), 2);
		System.out.println(value);
		System.out.printf("%x\n", value);
	}
}
```

Listing 1. First Quantum Program

As shown in Listing 1, this code is quite simple, as the main goal is to guide students in configuring their development environments. Using an integrated development environment like Eclipse, this process usually does not take long. Therefore, we use this lab to teach students how to generate a random key of arbitrary length. Students must generate a simple four-bit key, represented in binary, decimal, and hexadecimal, as a starting point (see Listing 2). We then challenge them to generate a key of sufficient length for post-quantum cryptographic applications (e.g., 3,072 bits) and display it in hexadecimal.

```
1110
14
e
```

Listing 2. Possible Output of the Code in Listing 1

3.6 The Second Lecture

The second lecture explores several low-level concepts, including qubits, quantum tunneling, quantum hardware, and quantum gates. The Pauli-X gate is presented, along with a simple algorithm for flipping a qubit, as shown in Listing 3.

```
• Create Environment
• Create a Program w/ 1 Qubit (Value is 0 by default)
• Create a Step
• Add an X gate to the step
• Add the step to the program
• Run the program
• Get all Qubits from the Result (only 1 qubit at position 0)
• Measure and display
```

Listing 3. Pseudocode for Bit Flipping Using a Pauli-X Gate

The corresponding code from the low-level Strange API is shown in Listing 4, which explains how the Strange classes and interfaces can be used to implement the algorithm.

ISSN: 2473-4901
v12 n6832

```
public static void main (String[] args) {
        QuantumExecutionEnvironment simulator =
                new SimpleQuantumExecutionEnvironment();

        // create program
        Program program = new Program(1);
        Step step = new Step();
        step.addGate(new X(0));
        program.addStep(step);

        // run program and process result
        Result result = simulator.runProgram(program);
        Qubit[] qubits = result.getQubits();
        System.out.println(qubits[0].measure());
}
```

Listing 4. Bit Flipping Program Implemented With the Low-Level Strange API

The efficiency of Grover's algorithm is also discussed, but the lecture focuses on the Hadamard gate and its role in quantum random number generation. We develop an algorithm for a Quantum Random Number Generator (QRNG) and compare our pseudocode with that from the Pauli-X gate bit-flipping algorithm and example code.

We then emphasize that, while digital computers can only generate pseudorandom numbers, quantum computers, using superposition, can produce truly random numbers that can serve as the basis for new cryptographic algorithms and systems.

At this point, the students should be ready to complete the low-level lab.

3.7 Low-Level Project Instructions

This hands-on project requires the same Eclipse environment that the students built for the previous lab. Depending on the class's ability level, it may be possible to ask students to implement the QRNG algorithm (see Listing 5) we discussed in the lecture. If students need more explicit guidance, the tutorial in Appendix C is provided. The tutorial displays a single random bit. It is emphasized that while classical computers can only generate pseudo-random numbers, this bit would be truly random if the simulator used by the Strange library were replaced with actual quantum hardware.

- Request one Qubit in the "ket-zero" state
- Apply the Hadamard Gate to the Qubit
- The Qubit is now in superposition
- Measure the Qubit

Listing 5. Pseudocode for the Quantum Random Number Generator

As time allows, students could be challenged to combine the features of both programs to produce a program that can generate a truly random number of any size.

3.8 Post-Test

After the second lab, students are asked to take a post-test containing the same 10 questions as those asked during the penultimate week of the semester.

We collected data from the students who took both the pre-test and the post-test and compared the results. A total of 104 students took both quizzes. Both tests were proctored in the classroom. These 104 students represent 94% of the total enrollment. The analysis excluded students who were not present for both quizzes.

A paired-samples t-test indicated that scores increased significantly from pre-test (M = 10.24, SD = 3.58) to post-test (M = 17.09, SD = 6.68), t(103) = 10.07, p < 0.0001. By conventional criteria, this difference is highly statistically significant.

4. THE ONLINE SURVEY

To collect data on students' backgrounds, perceptions of learning effectiveness, and feedback on the learning experience, an online survey was created in Qualtrics. This tool helped ensure anonymity and encouraged students to share their perceptions more openly.

The survey included 13 questions (see Appendix D): five multiple-choice questions on students' backgrounds, six Likert-scale questions on their perceptions of learning effectiveness, and two essay questions requesting feedback on the learning experience.

Question 1 asked students to identify their major. Of the 89 students, 49 (55.06%) chose Computer Information Systems (CIS), 2 (2.25%) chose Computer Science (CS), 38 (42.70%) chose Information Technology (IT), and none chose Other (please specify). Question 2 asked students to identify their academic status. Of the 89 students, 81 (91.01%) were full-time, and 8 (8.99%) were part-time. Question 3 asked students to identify their academic classification. All 89 students (100.00%) were seniors. Question 4 asked students to report on their current overall GPA. Of the 89 students, 1 (1.12%) selected 1.99 or less, 4 (4.49%) selected 2.00–2.49, 16 (17.98%) selected 2.50–2.99, 38 (42.70%) selected 3.00–3.49, and 30 (33.71%) selected 3.50–4.00.

Table 1 presents the survey results of the six Likert-scale questions (Questions 5-10) on students' perceptions of learning effectiveness. Questions 5 and 6 compared students' perceived understanding of build tools such as Ant, Maven, and Gradle before and after completing the course. Prior to the class, a large majority of students reported limited familiarity with these tools: 70.79% strongly disagreed, and 13.48% somewhat disagreed that they had a good understanding. Only a small proportion expressed any level of agreement (6.74% combined). In contrast, after completing the course, responses shifted markedly toward agreement: 61.80% somewhat agreed and 7.87% strongly agreed that they had a good understanding. In comparison, the proportions of disagreement dropped to 4.49% (strongly disagree) and 5.62% (somewhat disagree). Additionally, 20.22% of students selected a neutral response, suggesting some remaining uncertainty. Overall, these results show that the course effectively increased students' perceived understanding of build tools.

Questions 7 and 8 compared students' perceived understanding of algorithmic complexity before and after completing the course. Before taking the class, a majority of students reported limited understanding, with 32.58% strongly disagreeing and 22.47% somewhat disagreeing that they had a good grasp of the topic. Only 29.21% expressed some level of agreement (19.10% somewhat agree and 10.11% strongly agree). After the course, however, responses shifted significantly toward agreement: 58.43% somewhat agreed, and 17.98% strongly agreed, indicating that over three-quarters of the students felt confident in their understanding. Meanwhile, disagreement dropped sharply to just 6.74% combined. The proportion of neutral responses remained relatively stable (15.73% before vs. 16.85% after), suggesting that while some uncertainty persisted, most students gained meaningful confidence in their understanding. Overall, these results indicate that the course effectively enhanced students' comprehension of algorithmic complexity.

Questions 9 and 10 compared students' perceived understanding of quantum computing before and after completing the course. Prior to the class, most students reported little familiarity with the topic: 64.04% strongly disagreed, and 14.61% somewhat disagreed that they had a good understanding. Only 11.23% indicated any level of agreement. After the course, however, responses shifted markedly toward agreement: 55.06% somewhat agreed and 8.99% strongly agreed that they had a good understanding, representing a clear majority of the class. Meanwhile, the proportion of students expressing disagreement dropped sharply to 3.37% (strongly disagree) and 12.36% (somewhat disagree). Neutral responses increased to 20.22%, suggesting that some students remained uncertain despite the overall gains. Overall, these results indicate that the course was highly effective in improving students' perceived understanding of quantum computing.

ISSN: 2473-4901
v12 n6832

	Q5. Before taking this class, I already had a good understanding of build tools such as Ant, Maven, and Gradle.	Q6. After taking this class, I have a good understanding of build tools such as Ant, Maven, and Gradle.	Q7. Before taking this class, I already had a good understanding of algorithmic complexity.	Q8. After taking this class, I have a good understanding of algorithmic complexity.	Q9. Before taking this class, I already had a good understanding of quantum computing.	Q10. After taking this class, I have a good understanding of quantum computing.
Strongly disagree	63 (70.79%)	4 (4.49%)	29 (32.58%)	3 (3.37%)	57 (64.04%)	3 (3.37%)
Somewhat disagree	12 (13.48%)	5 (5.62%)	20 (22.47%)	3 (3.37%)	13 (14.61%)	11 (12.36%)
Neither agree nor disagree	8 (8.99%)	18 (20.22%)	14 (15.73%)	15 (16.85%)	9 (10.11%)	18 (20.22%)
Somewhat agree	5 (5.62%)	55 (61.80%)	17 (19.10%)	52 (58.43%)	9 (10.11%)	49 (55.06%)
Strongly agree	1 (1.12%)	7 (7.87%)	9 (10.11%)	16 (17.98%)	1 (1.12%)	8 (8.99%)

Table 1. Six Likert-Scale Questions on Students' Perceptions of Learning Effectiveness

Question 11 asked students to report their knowledge of quantum computing before they started this class. The survey results indicate that most students had very limited prior knowledge of quantum computing before taking the class. A clear majority (66.29%) reported having only heard of the topic, while 23.60% indicated understanding the basic definitions. Very few students had any meaningful experience: only 4.49% reported some hands-on experience, and another 4.49% had completed at least part of a semester covering the subject. Just one student (1.12%) reported a year or more of exposure through school or personal interest, and none had professional experience in the field. Overall, these findings suggest that the course primarily served students with little to no background in quantum computing, highlighting the module's introductory nature.

Question 12 asked students whether the quantum computing assignments provided valuable learning experiences. Based on 82 responses, we conducted a sentiment-structured thematic analysis. Each response was coded as primarily positive, negative, or mixed based on its dominant stance toward the perceived learning value of the quantum computing assignments. Overall, sentiment was strongly positive, with three-quarters of students (n = 62, 75.6%) perceiving at least some educational value. Eight primary themes emerged, including five positive, two negative, and one mixed. Table 2 shows each theme's title, description, number of responses, and representative examples. Overall, the findings show a strongly positive but future-oriented perception of the quantum computing assignments. Most students valued them as an introductory exposure to an emerging field, even when they struggled to comprehend. Negative feedback focused mainly on difficulty and limited depth rather than lack of interest, suggesting that better scaffolding and pacing could further improve perceived value.

	Themes	Description	Number of Responses	Representative Examples
Positive	Theme 1: Future Relevance and Career Preparation	Many students believe quantum computing is an emerging field that will become important in future careers and technology systems.	28 (34.1%)	• "Quantum computing will probably be the future of computers, and it is good to learn about it." • "It is the future of computing." • "It will be part of future infrastructure and having a basic understanding is valuable."

ISSN: 2473-4901
v12 n6832

	Theme 2: Introductory Exposure and Basic Understanding	Students frequently emphasized that the assignments helped them gain initial exposure to a new and unfamiliar field.	20 (24.4%)	• "Before I completed the assignments, I had no knowledge of quantum computing but now I have a stable idea." • "They were good for an overview and basic understanding." • "Yes, because students can get basic knowledge of quantum computing."
	Theme 3: Interest and Engagement	Some students highlighted curiosity and interest as key reasons for valuing the assignments.	7 (8.5%)	• "It is something different and interesting." • "It was very interesting, and I did enjoy it." • "It's pretty cool to see where computers are heading."
	Theme 4: Hands-on Learning Experience	A smaller group valued the applied, coding-based nature of the assignments.	7 (8.5%)	• "It allowed us to get hands on with the code itself." • "They gave me a demo of how quantum computing functions." • "It showed how quantum computing can outperform classical randomness."
	Theme 5: Self-Reported Learning Gains	Students reported improved understanding and knowledge development.	10 (12.2%)	• "I felt it advanced my knowledge of quantum computing." • "I could understand more than before." • "Yes, they helped me understand the basics."
Negative	Theme 6: Difficulty and Lack of Understanding	The most common negative sentiment was difficulty understanding the material due to math, programming, or pacing issues.	6 (7.3%)	• "No, I'm not familiar with the math that the lectures and assignments refer to." • "No. It's difficult." • "Was kind of rushed and over my head."
	Theme 7: Limited Practical Usefulness	Some students questioned the relevance of the assignments for their careers or expressed low engagement.	4 (4.9%)	• "The likelihood most of us go on to use quantum computing feels low." • "No, I feel like I was just copying code." • "No, not very valuable, but interesting."
Mixed	Theme 8: Conditional or Balanced Views	These students acknowledged both value and limitations, often depending on time, depth, or prior knowledge.	10 (12.2%)	• "Yes and no. There is a lot of information to retain." • "It was decent, but I would need more in-depth study." • "Possibly useful in the future, but difficult right now."

Table 2. Perceived Value of Quantum Computing Assignments

Question 13 asked students to report on the most difficult challenges they encountered while working on the quantum computing assignments. A total of 79 responses were analyzed using thematic analysis to identify key challenges students encountered. Eight primary themes emerged. Table 3 shows each theme's title, description, number of responses, and representative examples. Overall, conceptual difficulty (22.8%) and programming-related issues (20.3%) dominated student challenges, followed by mathematical complexity and technical setup issues. Fewer students emphasized pacing and debugging problems, while approximately 12.7% reported minimal or no difficulties.

Themes	Description	Number of Responses	Representative Examples
Theme 1: Conceptual Difficulty and Lack of Understanding	Many students reported difficulty understanding the meaning of quantum computing concepts, including qubits, gates, and underlying system behavior.	18 (22.8%)	• "I still feel as though I don't have a full understanding of what quantum computing actually does." • "Understanding what it is doing under the hood was difficult." • "I don't get the idea of gates much."
Theme 2: Programming and Java-Related Challenges	A significant number of students struggled with programming tasks, particularly with Java syntax and structure, and with a lack of prior experience.	16 (20.3%)	• "Unfamiliarity with Java. I had not touched it again until this class." • "The code itself. I had trouble with Java because I am not very good at Java code." • "Trying to figure out how to write the code because I had not used the language before."
Theme 3: Mathematical and Algorithmic Complexity	Students reported difficulty understanding the mathematical foundations and algorithms behind quantum computing.	12 (15.2%)	• "Understanding what the mathematics meant for the functions like 2^n / 2 log n." • "The complexity of the algorithms would be the most difficult part." • "Not having much previous knowledge of how math works behind the scenes."
Theme 4: Setup and Technical Environment Issues	Many students experienced challenges related to installation, environment setup, and tool configuration.	10 (12.7%)	• "How to set up environment and use algorithm." • "Installation and using Java have been the most difficult part." • "The most difficult challenge of these projects would mostly be getting the build setup."
Theme 5: Pacing and Time Constraints	Students frequently noted that the pace of instruction and limited time made learning difficult.	9 (11.4%)	• "The speed of going through them in class was too fast." • "Not enough time to try and understand while doing other assignments." • "It is hard to learn all of that in 2 class periods."
Theme 6: Cognitive Overload and	Some students described being overwhelmed by the volume of new	8 (10.1%)	• "There is a lot of knowledge dumped in a short amount of

Learning Curve	information and steep learning curve.		time." • "Mainly everything… I am lost at most quantum components." • "It was a learning curve that was somewhat challenging."
Theme 7: Errors and Debugging Difficulties	A subset of responses focused on frustration with debugging and error resolution.	6 (7.6%)	• "Understanding what errors meant in development." • "Strange errors in the code that were difficult to figure out." • "Getting all the code to work the way I wanted them to."
Theme 8: Minimal or No Difficulty Reported	A minority of students reported little to no difficulty or indicated that challenges were manageable.	10 (12.7%)	• "I didn't really struggle with the assignments." • "None." • "The assignments were pretty straightforward."

Table 3. Challenges Encountered in Quantum Computing Assignments

5. DISCUSSION AND IMPLICATIONS

5.1 Overview of Findings

This study examined the effectiveness of a hands-on learning module designed to introduce quantum computing concepts without requiring access to physical quantum hardware. Findings from both quantitative and qualitative analyses indicate that the module improved students' understanding while also revealing important challenges related to instructional design and student preparedness.

The paired-samples t-test demonstrated a statistically significant improvement in student performance from pre-test to post-test, suggesting that even a short, focused intervention can produce meaningful learning gains. In addition, Likert-scale results showed substantial increases in students' perceived understanding of build tools, algorithmic complexity, and quantum computing. These results are further supported by qualitative findings, in which most students reported that the assignments provided valuable learning experiences, particularly as an introduction to an emerging field.

5.2 Interpretation of Results

A key insight from this study is that students' perceptions of value were strongly future-oriented. Most students viewed quantum computing as important, not because of immediate applicability, but because of its anticipated role in future technologies and careers. This aligns with prior research (e.g., Bond et al., 2020; Lindín et al., 2023) suggesting that emerging technologies motivate learners through perceived long-term relevance rather than short-term utility.

At the same time, the findings highlight a clear tension between perceived importance and cognitive accessibility. While students appreciated exposure to quantum computing, many struggled with conceptual understanding, mathematical abstraction, and programming implementation. This tension is reflected in both the qualitative responses and the distribution of themes, where conceptual difficulty and programming challenges were among the most frequently reported issues.

Another important finding is the role of hands-on, simulator-based learning. Students who engaged with coding assignments and interactive tools reported better understanding and higher engagement. This supports prior literature (e.g., Duchatelet et al., 2024; Kolb, 1984; Prince, 2004; Freeman et al., 2014; Zhang et al., 2023) emphasizing the effectiveness of experiential learning environments in teaching complex and abstract topics. Even when students did not fully grasp the underlying theory, experimenting with code appeared to enhance comprehension and retention.

5.3 Pedagogical Implications

The results of this study suggest several implications for the design and implementation of quantum computing education in undergraduate computing programs:

- *Emphasize Conceptual Scaffolding.* Students frequently reported difficulty understanding fundamental concepts such as qubits, gates, and quantum behavior. Instruction should incorporate scaffolded learning approaches that gradually build conceptual understanding before introducing more complex algorithms or implementations.
- *Address Programming Readiness.* Challenges related to Java programming indicate that prerequisite knowledge plays a critical role in student success. Instructors may need to provide additional support, such as refresher materials, code templates, or guided walkthroughs, particularly for students with limited programming experience.
- *Manage Cognitive Load and Pacing.* Many students cited fast pacing and information overload as barriers to learning. Spreading the module across more instructional time, integrating smaller incremental tasks, and providing opportunities for reflection may improve comprehension.
- *Leverage Hands-on and Simulation-Based Tools.* The positive response to hands-on activities suggests that simulator-based environments can effectively substitute for physical quantum hardware. These tools let students engage with real-world concepts in a controlled, accessible way, making them suitable for broader adoption in undergraduate curricula.
- *Focus on Relevance and Motivation.* Students were highly motivated by the perceived future importance of quantum computing. Instructors should continue to connect course content to real-world applications, industry trends, and career pathways to sustain engagement.

5.4 Implications for Curriculum Design

More broadly, the findings support integrating quantum computing into undergraduate computing education as an introductory, exploratory topic rather than a deeply technical specialization. Given that most students entered the course with little to no prior knowledge, modules like the one presented in this study can serve as an effective entry point.

Institutions may consider:

- Embedding quantum computing concepts into existing courses (e.g., software architecture, algorithms, or cybersecurity)
- Developing interdisciplinary modules that reduce reliance on advanced physics and mathematics
- Expanding exposure through elective courses, workshops, or capstone projects

These approaches can build quantum literacy across computing disciplines without extensive curriculum restructuring.

5.5 Limitations

Several limitations should be acknowledged. First, the study was conducted within a single course at one institution, which may limit generalizability. Second, the module's duration was relatively short, which may have constrained the depth of learning. Third, reliance on self-reported survey data may introduce response bias. Finally, variations in students' prior programming and mathematical backgrounds were not fully controlled, which may have influenced both performance and perceptions.

5.6 Future Research

Future research should explore longer-term implementations of quantum computing modules and examine their impact on knowledge retention and skill development. Comparative studies across different programming languages or platforms (e.g., Python-based frameworks) may also provide insight into how tool selection influences learning outcomes. Additionally, investigating scaffolded instructional designs and adaptive learning approaches could help address the challenges identified in this study.

5.7 Conclusions

This study demonstrates that quantum computing can be effectively integrated into undergraduate computing education without access to quantum hardware through hands-on, simulator-based learning modules. While students recognize the importance of quantum computing and benefit from early exposure, instructional strategies must address challenges related to conceptual difficulty, programming readiness, and cognitive load. By adopting scaffolded, experiential, and accessible approaches, educators can better prepare students for a future in which quantum computing plays an increasingly important role.

6. REFERENCES

Arai, K., Kapoor, S., & Bhatia, R. (2021). Proceedings of the Future Technologies Conference (FTC) 2020 (Vol. 2). Springer Nature. https://doi.org/10.1007/978-3-030-63089-8

Bass, L., Clements, P., & Kazman, R. (2022). *Software architecture in practice* (4th ed.). Addison-Wesley Professional.

Bond, M., Buntins, K., Bedenlier, S., Zawacki-Richter, O., & Kerres, M. (2020). Mapping research in student engagement and educational technology in higher education: A systematic evidence map. *International Journal of Educational Technology in Higher Education, 17*(2). https://doi.org/10.1186/s41239-019-0176-8

Duchatelet, D., Cornelissen, F., & Volman, M. (2024). Features of experiential learning environments in relation to generic learning outcomes in higher education: A scoping review. *Journal of Experiential Education, 47*(3), 400-423. https://doi.org/10.1177/10538259231211537

Dulababu, T. (2025). Business schools at the crossroads: Strategies for job-proof graduates in the AI-quantum age. *IUP Journal of Business Strategy*, *22*(3), 50–64.

Economou, S. E., & Barnes, E. (2022). *Quantum computing pedagogy: Curriculum design and methodologies*. Springer Quantum Technologies. https://link.springer.com/article/10.1007/s44217-025-00400-1

Farooq, U., & Upadhyay, K. (2025). *Teaching quantum computing through lab-integrated learning: Bridging conceptual and computational understanding*. arXiv. https://arxiv.org/abs/2511.02844

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23), 8410-8415. https://doi.org/10.1073/pnas.1319030111

Knobloch, N. A., & Smith, M. (2024). Experiential learning in school-based agricultural education. In R. K. Barrick & A. C. Thoron (Eds.), *Emerging research in agricultural teacher education* (pp. 98–121). IGI Global. https://doi.org/10.4018/979-8-3693-2766-1.ch006

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.

Kushimo, T., & Thacker, B. (2022). *Investigating students' strengths and difficulties in quantum computing*. arXiv. https://arxiv.org/abs/2212.03726

Lindín, C., Engel, A., Gràcia, M., Rivera-Vargas, P., & Rubio, M. J. (2023). Literature review on emerging educational practices mediated by digital technologies in higher education. *SAGE Open, 13*(4). https://doi.org/10.1177/21582440231204677

López-Meneses, E., Cáceres-Tello, J., Galán-Hernández, J. J., & López-Catalán, L. (2025). Quantum computing in data science and STEM education: Mapping academic trends and analyzing practical tools. *Computers*, *14*(6), 235. https://doi.org/10.3390/computers14060235

NIST. (2024, August 13). *Announcing approval of three federal information processing standards (FIPS) for post-quantum cryptography*. NIST | CSRC. https://csrc.nist.gov/News/2024/postquantum-cryptography-fips-approved

Nott, G. (2018, March 7). Four universities set to join forces with "Sydney Quantum Academy." CIO.

Padmaavathy, P. A. (2023). Changing the landscape—Uncover the emerging technology in business. *PRERANA: Journal of Management Thought & Practice*, *15*(2), 46-56.

Peel, M. (2025, June 23). *UK government to invest more than £500mn in quantum computing*. Financial Times. https://www.ft.com/content/3ae0fa88-2884-47e5-93c4-4690f37ceceb?syn-25a6b1a6=1

Prince, M. (2004). Does active learning work? A review of the research. *Journal of Engineering Education, 93*(3), 223-231. https://doi.org/10.1002/j.2168-9830.2004.tb00809.x

Vos, J. (2022). *Quantum computing in action.* Manning Publications.

Yusufa, A., Román-González, M., & Behera, S. K. (2025). *Quantum computing in K-12 education: An early systematic review*. Computer Science Education. https://doi.org/10.1080/08993408.2025.2545915

Zhang, H., Kee, T., & King, R. B. (2023). An empirical study on immersive technology in synchronous hybrid learning in design education. *International Journal of Technology and Design Education, 34*, 1243-1273. https://doi.org/10.1007/s10798-023-09855-5

ISSN: 2473-4901
v12 n6832

Appendices

APPENDIX A

Pre/Post Test Questions

1. What is the best definition of a Quantum Computer?
    a. a very fast type of computer
    b. a "magic bullet" computer that will solve most, if not all, of the most difficult computer problems
    *c. a device that takes data as input and performs operations that can only be described using quantum physics
    d. a computer, with hardware based on quantum physics, that will replace classical computers

2. Which of the following would provide the best performance?
    *a. O(lg n)
    b. O(n^2)
    c. O(2^n)
    d. O(n)

3. Which of the following search algorithms would provide the best performance given unsorted data?
    a. Binary search algorithm
    b. Linear search algorithm
    *c. Grover's algorithm
    d. Shor's algorithm

4. Which of the following courses would provide the best mathematical basis for quantum computing?
    a. calculus
    b. differential equations
    *c. linear algebra
    d. trigonometry

5. Which of the following are gates used by classical computers?
    *a. AND
    *b. OR
    *c. NOT
    d. X
    e. H
    *f. NAND
    g. CNOT

6. Which of the following are gates used by quantum computers?
    a. AND
    b. OR
    c. NOT
    *d. X
    *e. H
    f. NAND
    *g. CNOT

7. Which of the following gates is used to put a qubit into a state of superposition?
    a. AND
    b. OR
    c. NOT
    d. X
    *e. H
    f. NAND
    g. CNOT

8. As soon as quantum computers become widely available, how will they most likely be used?

a. as desktop systems
*b. as cloud services
c. as portable systems
d. as external devices that attach via USB

9. What is the normal operating temperature of most quantum computers?
a. 60 degrees Fahrenheit
b. 32 degrees Fahrenheit (0 degrees Celsius)
c. 60 degrees Celsius (140 degrees Fahrenheit)
*d. near 0 degrees Kelvin (-459.67 Fahrenheit; -273 Celsius)

10. Write the pseudocode for the steps involved in creating a (truly) random number using a quantum computer.
1) Create a Qubit in the "ket-zero" state
2) Apply the Hadamard Gate to the Qubit
3) Measure the Qubit

APPENDIX B

Technical Tutorial: Quantum Computing With Strange and Maven

Strange is a quantum computing platform for Java.
Prerequisites: Maven, JDK 21, Git, Eclipse, and some Java experience.
Last update: April 30, 2026

Use a Maven Project to Build and Run a Quantum Program

1) Navigate to https://www.manning.com/books/quantum-computing-in-action.
2) Download the source code.
3) Extract the contents of the archive to your Desktop.
4) Using a PowerShell terminal, navigate to the project in Chapter 2.
5) After reviewing the files, build and run the project using Maven: mvn clean javafx:run
6) Save the file with the Java code (Main.java) to your Desktop for use in a later step.

Build the Strange Library From Source

1) cd ~\Desktop
2) Download the source from GitHub.
   git clone https://github.com/redfx-quantum/strange.git
3) Build the JAR file: mvn install
4) Copy the JAR file to your Desktop.
5) Remove or comment out the package statement from the Main.java file on your Desktop.
6) Compile and run the program:
   java -cp strange-0.2.1-SNAPSHOT.jar Main.java

Run the Quantum Program With Eclipse

1) Create a new Java Project: File; New; Java Project and name it "random."
2) There is no need for a Module.
3) Create a "lib" folder within your project (at the same level as "src").
4) Copy the Strange JAR created above into your new lib folder.
5) Right click on the JAR file and add it to your Build Path.
6) Copy the Main.java file from your Desktop to the project's src folder.
7) Build and run your project.

Use the Quantum High-Level API to Create a Random int

1) Modify your working program, from above, that creates a random bit in order to create a random integer of any size.
2) Rename Main.java as HighLevel.java and change the class name to match.
3) Make it easy to specify the size of the int in bits (e.g., use a hard-coded int to specify the number of bits). Alternatively, you could allow the end-user to specify the number of bits from the command line.
4) Use a Quantum Computer (or simulator) to generate that number of quantum bits.
5) Convert those random quantum bits to a single integer. For example:
   a. Given 4 bits as input, call randomBit four times.
   b. Assume that the QC generated: 1, 1, 0, 1
   c. Display 13 (since 1101 in binary is 13 in decimal).
6) Finally, display the random value as a hexadecimal number instead of decimal.

Note: Cryptographic systems, like RSA, often require very large integers (e.g., 3072 bits) to make their keys more secure. These large integers are commonly displayed in hex since the string of digits is shorter.

APPENDIX C

Technical Tutorial: Quantum Computing With the Low-Level Strange API

Strange is a quantum computing platform for Java.
Prerequisites: JDK 21 and Java experience. Familiarity with Strange and QC concepts is preferable.
Last update: April 30, 2026

Use the Strange Low-Level API to Create a Random Bit

1) Create a variable using the QuantumExecutionEnvironment interface and assign that variable an object from the concrete class: SimpleQuantumExecutionEnvironment (using the default, no-arg constructor).
2) Create a Program object by passing 1 (one qubit) to the constructor.
3) Create a new Step using the default, no-arg constructor.
4) Call the step's addGate method, passing a new Hadamard object created with a constructor argument of 0 (it will use qubit 0 – the only one we have; qubits are assigned a default value of zero upon creation).
5) Call the Program object's addStep method with an argument of the step you created above.
6) The object that you created at the top (Step 1) has a runProgram method. Call it passing the Program object as the argument. It will return a Result object, so create an object to hold that result.
7) Using the Result object, call the getQubits() method, which will return an array of Qubit objects (the array size here will be one).
8) Using the first element of the array, call its measure() method, which returns an integer. This is our truly random bit.
9) Display the bit's value.

Tip: If you are using Eclipse, you can use Ctrl-space as you are typing to ask Eclipse to find all matching class names. Alternatively, you can use "organize imports" (Ctrl-Shift O) to find the proper Strange classes to use as you are writing your code.

APPENDIX D

Quantum Computing Survey

This survey includes several questions. The last two are essay questions which are about your perceptions of the Quantum Computing assignments. Please respond to them with as much detail as possible in a manner that most accurately reflects your opinion. We value your honest opinions and appreciate your time and effort in sharing them with us. Thank you.

1. What is your major?
    a. Computer Information Systems (CIS)
    b. Computer Science (CS)
    c. Information Technology (IT)
    d. Other (please specify)
2. What is your academic status?
    a. Full-time
    b. Part-time
3. What is your academic classification?
    a. Freshman
    b. Sophomore
    c. Junior
    d. Senior
    e. Other than any of the above
4. What is your current overall GPA?
    a. 1.99 or less
    b. 2.00 – 2.49
    c. 2.50 – 2.99
    d. 3.00 – 3.49
    e. 3.50 – 4.00
5. Before taking this class, I already had a good understanding of build tools such as Ant, Maven, and Gradle.
    a. Strongly disagree
    b. Somewhat disagree
    c. Neither agree nor disagree
    d. Somewhat agree
    e. Strongly agree
6. After taking this class, I have a good understanding of build tools such as Ant, Maven, and Gradle.
    a. Strongly disagree
    b. Somewhat disagree
    c. Neither agree nor disagree
    d. Somewhat agree
    e. Strongly agree
7. Before taking this class, I already had a good understanding of algorithmic complexity.
    a. Strongly disagree
    b. Somewhat disagree
    c. Neither agree nor disagree
    d. Somewhat agree
    e. Strongly agree
8. After taking this class, I have a good understanding of algorithmic complexity.
    a. Strongly disagree
    b. Somewhat disagree
    c. Neither agree nor disagree
    d. Somewhat agree
    e. Strongly agree
9. Before taking this class, I already had a good understanding of quantum computing.
    a. Strongly disagree
    b. Somewhat disagree
    c. Neither agree nor disagree
    d. Somewhat agree

    e. Strongly agree
10. After taking this class, I have a good understanding of quantum computing.
    a. Strongly disagree
    b. Somewhat disagree
    c. Neither agree nor disagree
    d. Somewhat agree
    e. Strongly agree
11. What was your knowledge of quantum computing before you started this class?
    a. I had heard of it.
    b. I understood the basic definition(s) of quantum computing.
    c. Some experience with quantum computing.
    d. One semester or part of a semester (e.g., in Physics and/or Computer Science).
    e. One year or more, in school or personal hobbies only.
    f. One year or more, some as work outside of school.
12. Do you think that the quantum computing assignments provided valuable learning experiences? Why or why not?
13. What are the most difficult challenges that you encountered while working on these assignments? Please list all the challenges and provide as much detail as possible.