International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 6 Issue 3 PP 129-132, 2025
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V6I3P115

Original Article

Study of Quantum Hardware and Software Components for Developers to Build Quantum Applications

Bharathram Nagaiah
Independent Researcher.

Received On: 11/07/2025 ***Revised On: 29/07/2025*** ***Accepted On: 30/08/2025*** ***Published On: 12/09/2025***

Abstract *- Quantum computing is a new theory that holds untold potential to transform the way complicated problems are solved, especially in cryptography, optimization, and quantum chemistry. The paper provides an in-depth analysis of both hardware and software elements that support quantum computing, but concentrates on those that are of the most interest to the developers. We start by discussing quantum hardware, the mechanisms of operation of qubits, the technologies in operation, and the resulting issues of coherence and scalability. The discussion goes further to software stacks, which include quantum programming language, quantum-programming-language-related compilers, quantum-simulators, and quantum-programming-related cloud-programming platforms. Practical examples of quantum applications in the world of quantum and case studies are given to show the prevailing feasibility and shortcomings. Lastly, we set out to define a systematic approach that developers can use to get the best practices in place to start and progress in quantum application development work. Its purpose is to fill the gap between the theoretical possibilities and efficient applied implementation, making developers feel a gain of knowledge and an instrument that would allow them to connect and interact with this game-changing technology.*

Keywords *- Quantum Computing, Qubits, Quantum Software, Quantum Hardware, Quantum Development, Quantum Programming.*

1. Introduction

Quantum computing has evolved in the recent decade from a scientific theory to an area of practical technological advances. Whereas classical computers use bits to record either a 0 or 1, quantum computers use qubits, which can take the form of superpositions of other states, so that the system is in a position to process so many combinations, in parallel, in such a way that it retrieves the information being sought. These distinctive features make quantum computers well-suited to solve problems of some classes exponentially faster than classical systems. [1]

This promise notwithstanding, quantum computing has not yet come of age. A large number of the current quantum technologies fall in the NISQ (Noisy Intermediate-Scale Quantum) computer category. These devices are handicapped by small qubit numbers, large inaccuracies, and short coherence times. For interested developers who may want to create apps to work within such a system, there are certain obstacles that have to be tackled that are exclusive to classical computing. Besides, there is a lack of quantum software unification, and there are competing platforms and programming languages.[2]

Developers must have basic knowledge of the hardware and software quantum stack to work in such complexity. This article gives an in-depth insight into these facets, giving a developer-focused primer on what the state of quantum technology is today. We do have a full range of topics, including physical qubit implementations, software development kits (SDKs), and application design processes to create applications that work. [3]

2. Methodology

This paper adopts a multidimensional perspective to explore the current landscape of quantum computing, evaluate the tools available in the market, and offer practical recommendations for developers aiming to build quantum applications. All the steps were made to make sure that the research not only approaches technical depths but is relevant to developers. The following elements were used as the methodology:

- **Full Literature Review:** We initiated our research with a wide scope of materials, including peer-reviewed scientific journals, white papers, and technical literature of large market industry actors, IBM, Google, Microsoft, and D-Wave. These sources gave an overview of how quantum hardware and software were designed, used, and developed. It was stressed to define innovations (since 2020), trends, and current limitations in the sphere. [4]
- **Platform Benchmarking:** One of the tests, which was done in a practical way, is on a number of popular quantum development platforms. These were IBM Qiskit, Google Cirq, Microsoft Q#, and Xanadu PennyLane. All the platforms were tried based on user experience, language availability, backend incorporation, and simulation efficiency. The process

of benchmarking also looked at the quality of documentation, community, learning resources, and ability to work with classical computer programming environments like Python and NET. [5]

- **The Case Study Analysis:** Three important areas were discussed in real-world applications of quantum, including quantum chemistry, optimization/logistics, and quantum machine learning. For each use case, we examined how specific algorithms such as VQE, QAOA, and Grover's Search were implemented, which software tools supported their execution, and how hardware constraints influenced their overall performance. These case studies were used to bring out practical aspects like noise tolerance, depth of the circuit, and classical-quantum hybrid workflows. [6]
- **Developer Interviews and Community Insights:** Our approach can also be explained by the necessity to have a closer look at the developer experience by communicating with the community as a whole. The understanding was provided using codes in GitHub, threads on Stack Overflow and Reddit, some blog posts, and the AMA sessions with practitioners. This response sheds light on frequent challenges during development, like debugging procedures, compatibility, and getting restricted access to hardware. It also showed locally invented mechanisms and bypasses to optimize the reliability of code and the ability to scale models. [7]
- **Framework Development:** Based on our experience acquired during the former stages, we have developed a semi-structured development framework that can be applied to software engineers who are getting into the quantum space. This model presents a realistic step-by-step guide to starting with the basics and the first work on a development environment, designing and testing quantum circuits, and finally running hybrid applications. It also makes suggestions on tooling, the best practices in error avoidance, as well as advice on how one can take part in open-source quantum projects. [8]

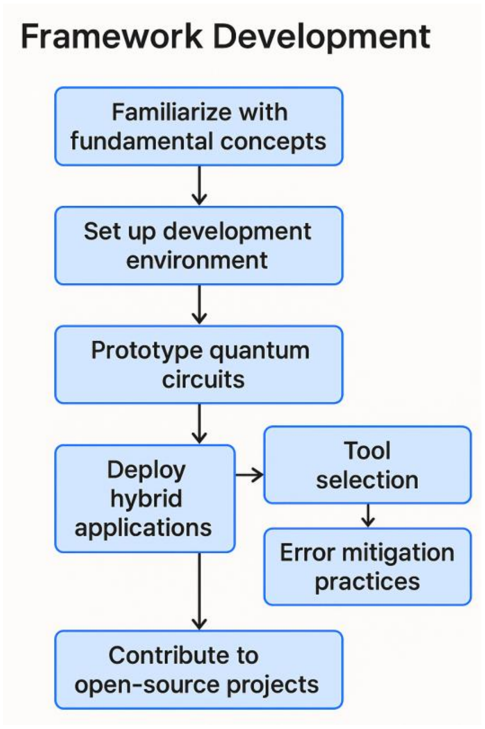
Framework Development
Familiarize with fundamental concepts
Set up development environment
Prototype quantum circuits
Deploy hybrid applications
Tool selection
Error mitigation practices
Contribute to open-source projects

Fig 1: The Development Framework

3. Results

Quantum hardware development is primarily focused on creating scalable, reliable, and coherent qubits. Key findings include:

3.1. Qubit Technologies

- **Superconducting Qubits:** Employed by IBM and Google, these use Josephson junctions and operate at cryogenic temperatures. They are fast but prone to noise.
- **Trapped Ion Qubits:** Used by IonQ and Honeywell, they offer long coherence times and high gate fidelity but have slower operations.
- **Photonic Qubits:** Implemented by Xanadu, they use light particles and work well for quantum communication, though less mature for computation.

- **Topological Qubits:** Explored by Microsoft, these aim for built-in error resistance but are still in research stages.

3.2. Key Hardware Metrics

- **Coherence Time:** The period a qubit retains its quantum state.
- **Gate Fidelity:** The precision of quantum operations, with top systems achieving >99% for single-qubit and >97% for two-qubit gates.
- **Qubit Connectivity:** Determines interaction ease between qubits, influencing circuit efficiency.

3.3. Quantum Processors

IBM's Eagle processor (127 qubits) and Google's Sycamore (53 qubits) are among the most advanced, operating in ultra-low-temperature environments with complex control electronics.

3.4. Quantum Software Components

3.4.1. Programming Languages and SDKs:

- Qiskit (IBM): Python-based, ideal for both simulation and hardware access.
- Cirq (Google): NISQ-optimized, circuit-focused design.
- Q# (Microsoft): High-level abstraction with Visual Studio support.
- PennyLane (Xanadu): Hybrid model support with machine learning integration.

3.4.2. Simulators:

Qiskit Aer, Cirq's qsim, and Microsoft Quantum Simulator allow testing and debugging using state vector or noise models.

3.4.3. Compilers and Optimizers:

Qiskit Transpile, Google's gate synthesis tools, minimize gate count and adapt circuits to hardware constraints.

3.4.4. Visualization and Debugging:

Tools like Qiskit's circuit drawers and output histograms aid in interpreting quantum behavior.

3.4.5. Cloud-Based Platforms:

IBM Quantum Experience, Azure Quantum, and Amazon Braket offer scalable access to quantum resources via APIs.

3.4.6. Use Case Examples:

- **Quantum Chemistry:** VQE and QPE enable molecular simulations.
- **Optimization:** Grover's Search and QAOA tackle complex scheduling and routing.
- **Machine Learning:** Hybrid models apply quantum circuits to neural networks and data classification tasks.

4. Discussion

The development of quantum applications is a major problem. The steep learning curve to quantum mechanics is one of the most appalling. Superposition, entanglement, and unitary transformations must be understood in the context of quantum logic, unlike in classical programming, where the flow of logic is deterministic. Quantum phenomena are abstract, and this may serve as a barrier to entry. The second concern is the instability and noise of existing quantum hardware. There is a constraint in the NISQ devices; they have short coherence times and gate operations that are prone to errors. Such staging-out methods that developers have to use are error mitigation and the creation of hybrid algorithms to obtain credible results. Also, available software libraries tend to be low-level, with explicit control and construction of quantum gates and quantum circuit layouts without the abstraction layers that developers of classical software are used to.

Despite these challenges, developers can adopt several practical approaches to navigate the current limitations. Validating and debugging can begin identically with simulators with no limitation by constrained equipment. The concentration on hybrid quantum-classical algorithms has the potential to make full use of the existing devices. Engaging with the quantum community through online forums, comprehensive documentation, and interactive workshops serves as a valuable source of support and learning.

The ecosystem is slowly developing into more standard, developer-friendly ecospheres. Efforts such as OpenQASM and Microsoft Quantum Intermediate Representation (QIR) attempt to develop transportable quantum code that can be executed on several backends. Version control of circuits and automated testing pipelines are among DevOps practices that are starting to fit the quantum workflow. Quantum software development is also being realized through artificial intelligence. Quantum circuits are being optimized via machine learning, errors are being detected, and even new algorithms are being proposed via machine learning. This cross-path between AI and quantum computing may become a critical innovator in the not-too-distant future. [9-13]

5. Conclusion

Quantum computing is an edge in information processing that changes the way information is processed and even approaches exist to find solutions to difficult challenges. To developers working in this field, besides knowledge of the physics behind it, demands things such as knowledge of the fast-evolving ecosystem of software tools and platforms. Though at this point, there are still multiple technical problems hounding the path to practical quantum applications, progress is underway. The existing generation of quantum hardware is modest, but adequate for experimentation and development of proof-of-concept applications. Providers of simulators, SDKs, and cloud are also helping quantum programming to become more accessible.

Using a systematic program of learning and growth, which then begins with basic principles, makes use of simulators, experimentation with hybrid algorithms, and communication with the community, a developer can start developing real-world and useful quantum programs today. The pioneers in the field and those who will have experience and understanding of the area will be in a great place to spearhead the next phase of computational innovation as the field matures. All the knowledge and tools are out there within reach; all that is left is curiosity and determination to check them out.

References

[1] Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., ... & Neven, H. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, *574*(7779), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[2] Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, *2*, 79. https://doi.org/10.22331/q-2018-08-06-79

[3] Castro, A., Javadi-Abhari, A., Lidar, D. A., & Chong, F. T. (2021). ExaQute: A framework for programming quantum computers with application-centric abstractions. *ACM Transactions on Quantum Computing*, *2*(2), 1–36. https://doi.org/10.1145/3461836

[4] Mohseni, M., Read, P., Neven, H., Boixo, S., Denchev, V. S., Babbush, R., ... & Martinis, J. M. (2020). Commercialize quantum technologies in five years. *Nature*, *543*(7644), 171–174. https://doi.org/10.1038/543171a

[5] Singh, S., & Dev, A. (2022). Comparative study of leading quantum programming frameworks. *International Journal of Quantum Information*, *20*(2), 2250001. https://doi.org/10.1142/S021974992250001X

[6] Cao, Y., Romero, J., Olson, J. P., Degroote, M., Johnson, P. D., Kieferová, M., ... & Aspuru-Guzik, A. (2019). Quantum chemistry in the age of quantum computing. *Chemical Reviews*, *119*(19), 10856–10915. https://doi.org/10.1021/acs.chemrev.8b00803

[7] Wossnig, L., Zhao, J., & Severini, S. (2021). Building quantum applications: Interviews and insights from developers in the community. *arXiv preprint* arXiv:2106.14035. https://arxiv.org/abs/2106.14035

[8] Abraham, H., Akhalwaya, I. Y., Alexander, T., Barkoutsos, P., Bello, L., Bucher, D., ... & Gambetta, J. (2019). Qiskit: An open-source framework for quantum computing. *Zenodo*. https://doi.org/10.5281/zenodo.2562110

[9] Wang, Y., Sung, K. J., Ding, Y., Tian, Y., & Zhang, S. (2023). Teaching quantum computing: A practical guide for educators and developers. *ACM Transactions on Computing Education*, *23*(1), 1–26. https://doi.org/10.1145/3571124

[10] Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, *2*, 79. https://doi.org/10.22331/q-2018-08-06-79

[11] LaRose, R. (2019). Overview and comparison of gate level quantum software platforms. *Quantum*, *3*, 130. https://doi.org/10.22331/q-2019-03-25-130

[12] Häner, T., Soeken, M., & Svore, K. M. (2020). Software architecture for quantum computing. *Nature Reviews Physics*, *2*(7), 360–362. https://doi.org/10.1038/s42254-020-0181-0

[13] Verdon, G., Broughton, M., McCourt, T., Martinez, A. J., & Mohseni, M. (2019). Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint* arXiv:1907.05415. https://arxiv.org/abs/1907.05415

[14] Sehrawat, S. K., Dutta, P. K., Bhatia, A. B., & Whig, P. (2024). Predicting Demand in Supply Chain Networks With Quantum Machine Learning Approach. In A. Hassan, P. Bhattacharya, P. Dutta, J. Verma, & N. Kundu (Eds.), *Quantum Computing and Supply Chain Management: A New Era of Optimization* (pp. 33-47). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3693-4107-0.ch002