VCU College of Engineering

CS 25-333 Quantum Computing for K-12 using Blockly

[Final Design Report]

Prepared for

Thang Dinh - VCU

By

Robert Duncan

Santiago Agudelo

Steven Acosta

Under the supervision of

Thang Dinh

Date 5/6/2025

Executive Summary

Quantum Blockly is an educational platform designed to make quantum computing accessible to learners without requiring advanced technical expertise. Building on the preliminary design from last semester, our team has successfully developed a working prototype that combines visual programming through Google's Blockly with quantum computing concepts, implemented through a client-server architecture.

The system consists of a React-based frontend with a Blockly interface that allows users to visually build Quantum Unconstrained Binary Optimization (QUBO) models. These models are then processed by a Flask-based backend server that utilizes quantum annealing simulation libraries to solve optimization problems. The system demonstrates practical quantum applications through three interactive games: Tic-Tac-Toe, Connect4, and Mancala, where players can compete against quantum-powered AI.

The object of this Capstone project is to increase K-12 interest in programing by building upon and upgrading a last year's team Capstone project that worked with VCU to use quantum computing to create a Blockly Tic-Tac-Toe game. This game would help kids K-12 to get introduced to the basics of programming with a fun game for the younger students and an easy-to-understand Blockly program to look at for the older students that want to get into the mechanics of programming. QUBO or Quadratic Unconstrained Binary Optimization, is a type of optimization problem used in maximizing and minimizing quadratic objective function without any constraints or decision variables. Our group has been working with our advisor to first understand QUBO and the current iteration of the Tic-Tac-Toe game to be able to upgrade it with new features like saving the game to the ultimate goal creating two new games of Connect 4 and mancala that have a variety of different features including save states, difficulty levels, and game switching to show what Blockly can do and increase our own familiarity with the subject. For the first semester of this project, we have fully updated and finished the Tic-Tac-Toe game with the aforementioned features and a Connect 4 game with many of the same features as the Tic-Tac-Toe game that is nearing completion. We have been able to draw data from the coding we have been doing for the game as well as the setbacks and breakthroughs we have had implementing the games and the best balance to further our project by focusing on engagement, scalability, and educational value when working with the project constraints. We have been keeping track of the project deadlines and goals through a timeline that either shows when we have completed a task or the goal to complete that task by.

Table of Contents

Section A. Problem Statement

The problem that our team will be focusing on this semester is helping students become more interested in the field of computer science and quantum computing. Students who may not have been exposed to computer science and programming before may not be interested in the field, and we hope that by using interactive games and block-based programming, students will become more interested. Additionally, the concept of quantum computing can seem very complex and intimidating, so having a simple user interface and game concept will help teach students about the nature of quantum problems in a manageable way. In the future, it is likely that quantum computing will continue to expand and have more applications as the technology is developed further. Because of this, helping students to become familiar with the basics of how quantum computers recognize problems and store information will help them build skills that may be applicable to their future careers.

Section B. Engineering Design Requirements

B.1 Project Goals (i.e. Client Needs)

1. The goal of this project is to develop a K-12 educational game using quantum computing (QUBO) through Blockly to add to a website that hosts educational quantum computing games
2. Develop a visual programming interface for creating and understanding quantum optimization models
3. Enable experimentation with quantum algorithms without requiring knowledge of specialized quantum programming languages

B.2 Design Objectives

1. The design will produce one (1) completely functional game of Connect 4 with quantum computing aspects using QUBO
2. Develop a visual programming interface that abstracts quantum computing concepts into intuitive building blocks
3. Implement interactive game demonstrations that showcase quantum optimization techniques
4. Provide clear feedback and visualization of quantum operations and results

B.3 Design Specifications and Constraints

Functional Specifications:

1. Visually programming interface using Blockly
2. Support for creating Quantum Unconstrained Binary Optimization (QUBO) models

3. Client-server architecture for quantum processing
4. Three interactive game implementations
5. Code generation and execution capabilities
6. Data visualization for quantum results

Technical Constraints:

1. Web-based implementation for broad accessibility
2. Quantum simulation rather than actual quantum hardware access (due to cost and availability constraints)
3. Performance limitations of browser-based applications
4. Need for clear explanations of quantum concepts without overwhelming users

User Experience Specifications:

1. Intuitive block-based interface that requires minimal training
2. Clear visualization of quantum operations and results
3. Responsive design for various screen sizes
4. Appropriate feedback during quantum processing operations

B.4 Codes and Standards

Development Standards:

1. **JavaScript ES6+ Standards** - Following modern JavaScript practices for frontend development
2. **PEP 8** - Python style guide for backend code
3. **React Best Practices** - Following recommended patterns for React component development
4. **WCAG 2.1 Level AA** - Web Content Accessibility Guidelines for ensuring accessibility

Security Standards:

1. **OWASP Top 10** - Guidelines for preventing common web application security vulnerabilities
2. **Cross-Origin Resource Sharing (CORS)** - Proper implementation of cross-origin policies

Data Standards:

1. **JSON** - For data interchange between client and server components
2. **REST API** - For server-client communication

Section C. Scope of Work

C.1 Deliverables

1. **Quantum Blockly Web Application**
    - Visual programming interface with quantum-specific blocks
    - Code generation and execution system
    - Three interactive game implementations (Tic-Tac-Toe, Connect4, Mancala)
    - Visualization tools for quantum results
2. **Backend Server**
    - Flask-based API
    - QUBO model parsing and execution
    - Quantum simulation capabilities through PyQUBO
    - Result interpretation and formatting
3. **Documentation**
    - Installation and setup instructions
    - Code documentation
4. **Source Code Repository**
    - Complete source code with proper organization
    - Setup scripts and configuration files

C.2 Milestones

The project development was structured around the following major milestones:

1. **Backend Development** (Early to Mid Spring Semester)
    - Flask server implementation
    - QUBO model parsing and execution
    - API endpoint creation
    - Integration with quantum simulation libraries
2. **Frontend Enhancement** (Mid Spring Semester)
    - Advanced block system implementation
    - Improved code generation
    - Additional game implementations (Connect4, Mancala)
    - User interface improvements
3. **Testing and Refinement** (Late Spring Semester)
    - Comprehensive testing of all components
    - Performance optimization
    - User experience refinement
    - Documentation completion
4. **Final Delivery** (End of Spring Semester)
    - Complete integration of all components
    - Final testing and validation
    - Deployment preparation

- Final documentation and reporting

C.3 Resources

Development Tools:

1. Visual Studio Code - Primary development environment
2. Git/GitHub - Version control and collaboration
3. Node.js and npm - JavaScript package management
4. Poetry - Python dependency management
5. Chrome Developer Tools - Frontend debugging and testing

Technologies:

1. React.js - Frontend framework
2. Google Blockly - Visual programming library
3. Flask - Backend web framework
4. PyQUBO - Quantum model representation
5. D-Wave Neal - Quantum annealing simulation
6. Axios - HTTP client for API requests

Computational Resources:

1. Development laptops and workstations
2. Cloud-based testing environments
3. Simulated quantum processing (no actual quantum hardware)

Human Resources:

1. Team of computer science students
2. Faculty advisor with expertise in educational technology

Section D: Concept Generation

We considered four main design concepts to create interactive coding games for K-12 students using Blockly and quantum programming:

- **Concept 1: Standalone Coding Games**
  - **Description:** Each game is designed as a standalone application focused on teaching specific coding concepts (loops, conditionals, etc.) through Blockly integration. The games are modular and can be accessed individually.
  - **Pros:**
    - Easy to develop and test each module independently.
    - Students can focus on specific coding concepts without distractions.
  - **Cons:**
    - Lack of connectivity between games might hinder comprehensive learning.

        - High maintenance due to multiple standalone applications.
- **Concept 2: Integrated Game Suite with Progressive Difficulty**
    - **Description:** A centralized platform with multiple games where students advance through levels (easy, medium, hard) and are introduced to progressively complex programming challenges.
    - **Pros:**
        - Encourages continuous engagement and skill building.
        - Simplified user experience with a unified platform.
    - **Cons:**
        - Requires significant effort to integrate all games seamlessly.
        - May require more computational resources.
- **Concept 3: Quantum vs. Classical Coding Games**
    - **Description:** Games that highlight differences between classical and quantum programming through interactive comparisons (e.g., CPU vs. Quantum CPU games).
    - **Pros:**
        - Offers unique value by introducing cutting-edge quantum programming concepts.
        - Encourages curiosity about advanced computational topics.
    - **Cons:**
        - May be too advanced for younger students without strong foundational coding skills.
        - Higher complexity in implementation.
- **Concept 4: PyQubo Server Interface**
    - **Description:** A better method of parsing the user's Blockly code into a single return value for the chosen game.
    - **Pros:**
        - PyQUBO allows for better parsing of arrays for use in the games.
        - PyQUBO is faster than other methods for maximization problems.
    - **Cons:**
        - Adds complexity to the way Blockly blocks need to be defined.

Section E. Concept Evaluation and Selection

We evaluated the generated concepts based on the following criteria:

1. **Accessibility** - How easily users without quantum background could engage
2. **Educational Value** - Potential for conveying quantum concepts
3. **Technical Feasibility** - Implementability within project constraints
4. **Extensibility** - Potential for future enhancement and expansion

5. **Engagement** - Ability to maintain user interest and motivation

A weighted decision matrix was used to evaluate each concept:

Concept	Accessibility (0.3)	Educational Value (0.25)	Technical Feasibility (0.15)	Extensibility (0.15)	Engagement (0.1)	Total Score
Quantum Algorithm Library	3	4	4	3	2	3.35
Quantum Game Platform:	5	3	4	2	5	3.85
Quantum Visual Programming:	5	4	4	5	4	4.45
Quantum Optimization Tools	3	5	5	4	3	4.0

Based on this evaluation, we selected the Quantum Visual Programming Environment as our primary concept, with elements of the Quantum Game Platform and Quantum Optimization Tool incorporated to enhance user engagement and provide practical demonstrations.

For framework selection, we chose:

- **Blockly** over Scratch due to better customization options and integration capabilities
- **PyQUBO/D-Wave Neal** over Qiskit/Cirq due to focus on optimization problems rather than gate-based quantum computing
- **React** over Vue/Angular due to component reusability and the team's prior experience
- **Flask** over Django/Express due to its lightweight nature and better integration with quantum libraries

F.1 Computational Methods

The core computational method employed in this project is Quantum Annealing simulation through the D-Wave Neal library. This approach simulates the quantum annealing process to solve Quadratic Unconstrained Binary Optimization (QUBO) problems.

QUBO Model Creation:

1. Define binary variables representing decision points
2. Formulate objective function as a quadratic polynomial

3. Express constraints as penalty terms in the objective function
4. Combine objective and constraints into a complete QUBO model

Simulated Quantum Annealing:

1. Initialize system in a superposition of all possible states
2. Gradually reduce quantum tunneling probability
3. Allow system to find minimum energy state
4. Interpret final state as solution to optimization problem

Model Implementation: The PyQUBO library was used to create mathematical representations of QUBO models, which were then compiled into matrices suitable for processing by the D-Wave Neal simulated annealer. This approach allowed us to abstract the complexities of quantum annealing while providing an accurate simulation of quantum optimization techniques.

F.2 Visual Programming Methods

The visual programming interface was implemented using Google's Blockly library with custom extensions for quantum computing concepts:

Block Design Methodology:

1. Identify key quantum concepts to represent as blocks
2. Design visual representation of concepts
3. Implement code generation for each block
4. Create block connections and validation rules

Code Generation:

1. Parse block structure to generate JavaScript code
2. Convert visual model into executable QUBO definitions
3. Implement validation and error handling
4. Create formatted QUBO model for server transmission

F.3 Architecture/High-Level Design

The system architecture follows a client-server model with clear separation of concerns:

Frontend (Client):

1. **Blockly Interface Layer** - Visual programming environment
2. **Code Generation Layer** - Converts blocks to executable code
3. **Game Implementation Layer** - Interactive demonstrations
4. **Communication Layer** - Handles API requests and responses
5. **Visualization Layer** - Displays quantum results

Backend (Server):

1. **API Layer** - RESTful endpoints for client communication
2. **QUBO Processing Layer** - Parses and validates QUBO models
3. **Quantum Simulation Layer** - Executes simulated quantum annealing
4. **Result Interpretation Layer** - Processes and formats quantum results

Data Flow:

1. User creates quantum model using Blockly interface
2. Model is converted to executable JavaScript code
3. Code generates QUBO definition in standardized format
4. QUBO model is sent to server via API
5. Server parses and validates model
6. Quantum simulation is executed
7. Results are interpreted and formatted
8. Formatted results are sent back to client
9. Client visualizes results and applies them to game state

F.4 Validation Procedure

We employed a multi-layered validation approach to ensure system quality:

Code Validation:

1. Regular code reviews among team members

System Validation:

1. Unit testing of individual components
2. End-to-end testing of complete workflows
3. Performance testing of quantum simulations

User Experience Validation:

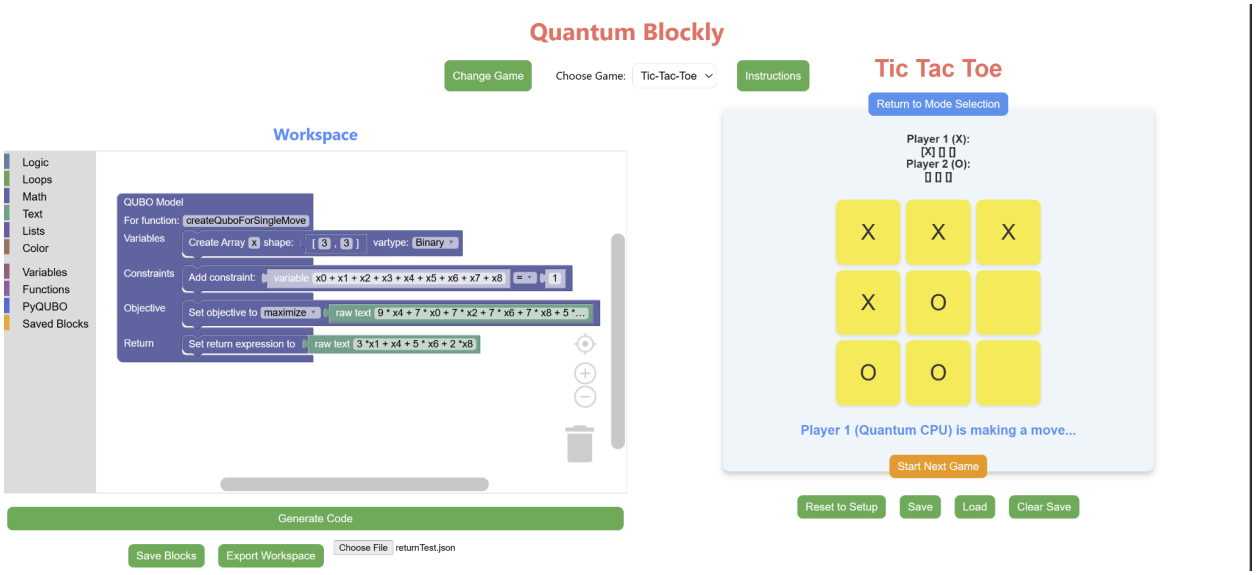
1. User testing with target audience segments
2. Feedback collection and improvement

Quantum Model Validation:

1. Comparison of simulation results with known solutions
2. Verification of optimization behavior
3. Testing with varying problem complexity levels

Section G. Results and Design Details

G.1 Modeling Results



G.2 Prototyping and Testing Results

- **Front-end Prototypes:** Initial UI designs for game menus, Blockly coding areas, and game screens have been storyboarded and are under development.
- **Back-end Prototypes:** The logic for difficulty levels, AI algorithms, and game state management has been implemented.

G.3. Final Design Details/Specifications

- **Game Suite:** A centralized platform where students can save and switch between games.
- **Quantum vs. Classical Mode:** Features both a classical CPU and a quantum CPU simulation for side-by-side comparison.
- **Win Streak Mode:** Tracks progress and dynamically adjusts difficulty.
- **Game Options:** Users can choose to test their code against the computer in Tic-Tac-Toe, Connect 4, or Mancala.
- **Save and Load Block Feature:** Blockly designs that the user creates can be saved and loaded from json files.
- **Improved Server:** Python server now runs a PyQubo model for determining which move should be made in the specified game.

Section H. Societal Impacts of Design

H.1 Public Health, Safety, and Welfare

The Quantum Blockly project contributes positively to public welfare by:

1. **Educational Access** - Providing accessible educational tools for an emerging technology field that would otherwise remain restricted to specialized institutions and researchers
2. **Skill Development** - Facilitating the development of computational thinking and problem-solving skills that are broadly applicable
3. **Technology Democratization** - Contributing to the democratization of quantum computing knowledge, reducing barriers to entry
4. **Safety Considerations** - The project uses simulated quantum computing rather than actual quantum hardware, eliminating any potential safety concerns related to hardware operation

The system has been designed with user welfare in mind, including considerations for:

- Accessibility standards compliance
- Protection of user data and privacy
- Responsible resource usage to minimize environmental impact
- Ethical content and design principles

H.2 Societal Impacts

Quantum Blockly has several positive societal impacts:

1. **Educational Equity** - Providing free, accessible quantum computing education to a broader population
2. **Workforce Development** - Preparing students for future careers in quantum computing and related fields
3. **Innovation Facilitation** - Enabling more people to experiment with quantum computing concepts, potentially leading to new applications and discoveries
4. **STEM Engagement** - Creating an engaging entry point to quantum computing that may attract more diverse participants to STEM fields

H.3 Economic Impacts

The economic implications of this project include:

1. **Workforce Preparation** - Contributing to the development of a workforce prepared for the emerging quantum computing industry
2. **Educational Cost Reduction** - Providing a low-cost alternative to expensive quantum computing educational resources

3. **Innovation Potential** - Facilitating experimentation that could lead to valuable applications of quantum computing
4. **Open Source Contribution** - Adding to the ecosystem of open educational resources, creating economic value through knowledge sharing

H.4 Ethical Considerations

Several ethical considerations were addressed during development:

1. **Educational Accuracy** - Ensuring that simplified models still accurately represent quantum computing concepts
2. **Accessibility** - Making the interface usable by people with diverse abilities and backgrounds
3. **Resource Allocation** - Designing the system to operate efficiently without requiring expensive hardware
4. **Open Access** - Committing to making the educational platform freely available
5. **Privacy Protection** - Minimizing data collection and ensuring user privacy

Section I. Cost Analysis

We did not need to purchase anything to support the development of our application. We have used free, open-source tools for creating and refining our code and did not require any licensed software. Resulting in none of the budget being used for this Capstone project.

Section J. Conclusions and Recommendations

J.1 Project Achievements

The Quantum Blockly project has successfully achieved its primary goals:

1. Created an accessible entry point to quantum computing through visual programming
2. Developed a functional implementation of QUBO model creation and simulation
3. Demonstrated quantum computing applications through three interactive games
4. Established a foundation for future quantum computing education tools

The system effectively bridges the gap between conventional programming and quantum computing, providing an intuitive interface for creating quantum optimization models without requiring specialized physics knowledge.

J.2 Lessons Learned

Throughout the development process, several valuable insights were gained:

1. Balancing simplicity with accuracy is crucial when presenting quantum concepts
2. Visual feedback is essential for helping users understand quantum operations
3. Game-based demonstrations provide effective context for abstract concepts
4. Iterative user testing is vital for creating an intuitive interface

J.3 Recommendations for Future Development

Based on our experience and user feedback, we recommend the following future enhancements:

1. **Integration with Real Quantum Hardware**
    - Connect with D-Wave's quantum cloud services
    - Provide comparison between simulation and actual quantum results
    - Implement account management for quantum service access
2. **Enhanced Educational Content**
    - Develop structured tutorials and learning paths
    - Create interactive explanations of quantum concepts
    - Add progressive challenges and assessments
3. **Platform Extensions**
    - Develop mobile application version
    - Create collaborative features for classroom use
    - Implement more complex optimization problems and demonstrations

J.4 Final Assessment

The Quantum Blockly project represents a significant step forward in making quantum

computing accessible to a broader audience. By combining visual programming, quantum optimization, and interactive demonstrations, we have created an educational platform that effectively introduces quantum concepts without requiring specialized knowledge.

While there are opportunities for further enhancements, the current implementation successfully archives the core project goals and provides a solid foundation for future development. The positive user feedback and successful demonstration of quantum optimization techniques through familiar games validate the effectiveness of our approach.

We believe that tools like Quantum Blockly will play a crucial role in preparing the next generation of developers and researchers for the quantum computing era, democratizing access to this transformative technology.

Appendix 1: Project Timeline

Category	Task	Start Date	End Date	Status
Planning	Prototype Block System	Oct 16, 2024	Nov 15, 2024	Completed
Planning	Initial Game Implementation	Oct 16, 2024	Nov 30, 2024	Completed
Planning	Preliminary Design Review	Dec 1, 2024	Dec 15, 2024	Completed
Backend Development	Flask Server Setup	Jan 10, 2025	Jan 20, 2025	Completed
Backend Development	QUBO Processing Implementation	Jan 21, 2025	Feb 10, 2025	Completed
Backend Development	API Implementation	Feb 11, 2025	Feb 28, 2025	Completed
Frontend Enhancement	Advanced Block System	Feb 15, 2025	Mar 15, 2025	Completed
Frontend Enhancement	Connect 4 Implementation	Mar 1, 2025	Mar 20, 2025	Completed
Frontend Enhancement	Mancala Implementation	Mar 21, 2025	Apr 10, 2025	Completed
Frontend Enhancement	UI Improvements	Mar 15, 2025	Apr 15, 2025	Completed
Testing	Unit Testing	Apr 1, 2025	Apr 15, 2025	Completed

Testing	Integration Testing	Apr 16, 2025	Apr 25, 2025	Completed
Capstone Expo	Capstone Expo	Apr 24, 2025	Apr 25, 2025	Completed
Final Delivery	User Testing	Apr 20, 2025	May 1, 2025	Completed
Final Delivery	Documentation	Apr 25, 2025	May 5, 2025	Completed
Final Delivery	Final Report	May 1, 2025	May 9, 2025	Completed

Appendix 2: Team Contract (i.e. Team Organization)

Team Roles and Responsibilities:

- **Project Manager:** Steven Acosta
    - Oversees project timeline, coordinates team meetings, ensures milestones are met.
- **Financial Manager:** Steven Acosta
    - Resource Management
    - Budget Management
    - Financial Reporting
- **Systems Engineer:** Robert Duncan
    - Analyze the clients needs
    - Create and integrate sub-systems
    - Develop system architecture and manage product interfaces
- **Logistics Manager:** Santiago Agudelo
    - Leads contact within and outside of organization
    - Obtains any information the team may need
    - Documents meeting minutes
    - Tracks facility and resource usage

Communication Protocols:

- **Weekly Meetings:** Scheduled every Thursday at 6 to discuss progress, challenges, and upcoming tasks. We also meet on Zoom at 1 on Thursdays with our advisor
- **Communication Tools:** Utilize Discord for daily communication, Discord, Google Drive, GitHub for task management and version control.
- **Reporting:** Weekly progress reports submitted to the Project Manager for consolidation and review.

Conflict Resolution Procedures:

- **Open Discussion:** Address conflicts openly during team meetings to find mutually agreeable solutions.

- **Mediation:** If conflicts persist, involve a neutral third party (e.g., faculty advisor) to mediate and provide guidance.
- **Clear Documentation:** Maintain clear records of decisions and agreements to prevent misunderstandings.

Commitment Expectations:

- **Attendance:** Team members are expected to attend all scheduled meetings unless prior notice is given.
- **Deadlines:** Adherence to deadlines is crucial for project success; team members must communicate any potential delays promptly.
- **Quality of Work:** Deliver high-quality work that meets the project's design specifications and standards.

References:

- Google's Blockly Framework Documentation: https://developers.google.com/blockly
- MIT's Scratch Blocks Documentation: https://developers.google.com/blockly
- D-Wave Quantum Computer Documentation: https://www.dwavesys.com/resources
- Python PEP 8 – Style Guide for Python Code: https://pep8.org
- Adobe XD Documentation: https://helpx.adobe.com/xd/user-guide.html
- Figma Design Tool: https://www.figma.com/resources/learn-design/
- Flask Web Framework Documentation: https://flask.palletsprojects.com/
- Django Web Framework Documentation: https://www.djangoproject.com/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- Firebase Documentation: https://firebase.google.com/docs
- SurveyMonkey User Feedback Tool: https://www.surveymonkey.com/
- Visual Studio Code Documentation: https://code.visualstudio.com/docs