tiny-qpu

Interactive Quantum Lab

USER MANUAL

Build quantum circuits, visualize quantum states, and explore quantum computing — all in your browser.

Version 2.0 — February 2026

Contents

1. Getting Started

Requirements

tiny-qpu requires Python 3.8 or later and Flask. The dashboard runs entirely locally — no internet connection needed, no cloud accounts, no data leaves your machine.

Installation

```
pip install flask
```

If you have the standalone executable (tiny-qpu.exe), no installation is needed. Simply double-click the file to launch.

Launching the Dashboard

From your terminal or PowerShell, navigate to the tiny-qpu directory and run:

```
python -c "from tiny_qpu.dashboard import launch; launch()"
```

Or if using the executable:

```
tiny-qpu.exe
```

Your default browser will open automatically to http://127.0.0.1:8888 showing the Interactive Quantum Lab dashboard.

2. Interface Overview

The dashboard has three panels arranged left to right, plus a top toolbar:

Header Bar

The top bar contains the tiny-qpu brand, qubit count selector (1-6 qubits), and shot count input (number of measurement samples). Changing the qubit count immediately resizes the circuit canvas.

Left Panel — Gate Palette

Lists all available quantum gates organized by category: single-qubit fixed gates (H, X, Y, Z, S, T), rotation gates (Rx, Ry, Rz, P), multi-qubit gates (CX, CZ, SWAP, CRz, CCX, CSWAP), and measurement (M). Below the gates, you will find preset circuits you can load with one click.

Center Panel — Circuit Builder

The main workspace. Horizontal lines represent qubit wires labeled q0, q1, etc. The toolbar above contains Run, Step, Clear, and Undo buttons, plus a status indicator showing gate count and circuit depth.

Right Panel — Results

Displays simulation results in five sections (top to bottom): Bloch Sphere visualization for each qubit, Probability bars showing the likelihood of each basis state, Measurement Histogram showing sampled counts, Amplitude Table with complex amplitudes and phases, and an OpenQASM editor for import/export.

3. Building Circuits

Placing Gates

There are two ways to add gates to your circuit:

Click method: Click a gate in the left palette to select it (it will be highlighted with a cyan border). Then click on any qubit wire in the circuit area to place it. The gate automatically finds the earliest available column.

Drag and drop: Drag a gate from the palette and drop it onto a qubit wire in the circuit area.

Multi-Qubit Gates

When placing a 2-qubit gate (like CX or SWAP), click on the control qubit — the target is automatically assigned to the adjacent qubit. For 3-qubit gates (CCX/Toffoli), the controls and target span three consecutive qubits from your click position.

Visual rendering: CNOT shows a filled dot (control) connected by a vertical line to a circled-plus (target). CZ shows two filled dots. SWAP shows two X marks.

Rotation Gates

When you place a rotation gate (Rx, Ry, Rz, P, CRz), a parameter dialog appears. Enter the rotation angle in radians, or click one of the quick-select buttons for common values:

Button	Value	Radians
pi	Full rotation	3.14159...
pi/2	Quarter turn	1.5708...
pi/4	Eighth turn (T gate angle)	0.7854...
pi/3	Third rotation	1.0472...
-pi/2	Negative quarter turn	-1.5708...
0	Identity (no rotation)	0

Removing Gates

Click any placed gate in the circuit to remove it. The remaining gates will automatically repack into the earliest available columns. You can also use the Undo button or Ctrl+Z to remove the most recently

placed gate.

4. Running Simulations

Click the **Run** button (or press R) to simulate your circuit. The dashboard sends the circuit to the tiny-qpu statevector simulator, which computes the exact quantum state and samples measurement outcomes.

What Happens When You Click Run

1. The circuit is converted to gate operations and sent to the backend.
2. The statevector simulator applies each gate sequentially to the initial |0...0> state.
3. The final statevector is used to compute probabilities, Bloch coordinates, and amplitudes.
4. The statevector is sampled N times (where N = Shots) to generate measurement counts.
5. All results appear in the right panel.

Shots

The Shots parameter controls how many times the final quantum state is measured. More shots give a more accurate histogram but the probabilities (computed from the statevector) are always exact. Default is 1024. For quick exploration, 100 shots is fine. For publication-quality histograms, use 8192 or higher.

Simulation Limits

The statevector simulator uses 2^n complex numbers (16 bytes each) for n qubits. Memory and time requirements grow exponentially:

Qubits	Statevector Size	Simulation Time
1-4	< 1 KB	Instant
5-10	< 16 KB	< 1 second
10-15	< 512 KB	< 5 seconds
15-20	< 16 MB	Seconds to minutes
20-25	< 512 MB	Minutes

The dashboard supports up to 6 qubits for interactive use, which runs instantly.

5. Understanding Results

Bloch Sphere

Each qubit is visualized on its own Bloch sphere — a 3D representation of a single qubit state. The north pole represents |0> and the south pole represents |1>. Points on the equator represent superposition states.

State	Bloch Position	Coordinates
|0>	North pole	(0, 0, 1)
|1>	South pole	(0, 0, -1)
|+> = (|0>+|1>)/sqrt(2)	Equator (positive x)	(1, 0, 0)
|-> = (|0>-|1>)/sqrt(2)	Equator (negative x)	(-1, 0, 0)
|i> = (|0>+i|1>)/sqrt(2)	Equator (positive y)	(0, 1, 0)
Mixed/Entangled	Near origin (shorter vector)	Purity < 1

When a qubit is entangled with other qubits, its individual Bloch vector shrinks toward the origin. An amber ring around the sphere indicates reduced purity (< 1.0), which means entanglement is present.

Probability Bars

Shows the exact probability of measuring each computational basis state, computed directly from the statevector (not from sampling). States are sorted by probability, with color coding: cyan for high probability (> 40%), amber for moderate (> 10%), and gray for low probability states.

Measurement Histogram

Shows the distribution of measurement outcomes from sampling the quantum state N times (controlled by the Shots parameter). Unlike probabilities, counts include statistical noise — this is what you would see from a real quantum computer. Each bar shows the count above it, with bars colored by hash for visual distinction.

Amplitude Table

Displays the full statevector in Dirac notation. For each basis state |b>, the table shows the complex amplitude (real + imaginary parts), the phase angle in degrees, and the probability ($|\text{amplitude}|^2$). Zero-amplitude states are shown as 0 with no phase.

6. Step-by-Step Mode

Step mode is the most powerful educational feature in tiny-qpu. It lets you watch the quantum state evolve gate by gate, seeing exactly how each operation transforms the statevector, probabilities, and Bloch sphere.

Using Step Mode

1. Build your circuit (or load a preset).
2. Click **Step** (or press S). An amber step indicator appears below the toolbar.
3. Use **Next** (or right arrow key) to advance one gate at a time.
4. Use **Prev** (or left arrow key) to go backward.
5. The right panel updates at each step showing the current quantum state.
6. The current gate is highlighted in the circuit view.
7. Click **Exit** (or press Escape) to return to normal mode.

> *Step 0 always shows the initial state |0...0> before any gates are applied.*

What to Watch For

Hadamard gate (H): Watch the Bloch vector rotate from the north pole to the equator, and the probability split from 100%/0% to 50%/50%.

CNOT after H: The target qubit's Bloch vector collapses toward the origin (entanglement!), and new basis states appear in the probability bars.

Rotation gates: Watch the Bloch vector sweep smoothly. Rx rotates around the x-axis, Ry around y, Rz around z.

7. Preset Circuits

The left panel (below the gate palette) contains preset circuits that demonstrate key quantum computing concepts. Click any preset to load and auto-run it.

Preset	Qubits	Description
Bell State (phi+)	2	Creates maximally entangled pair |00> + |11>. The foundation of quantum information.
GHZ State	3	Three-qubit entanglement: |000> + |111>. Extends Bell state to 3 parties.
Uniform Superposition	3	H gates on all qubits. Every basis state equally likely.
Teleportation	3	Quantum teleportation protocol. Transfers qubit 0's state to qubit 2.
Deutsch-Jozsa	3	Determines if a function is constant or balanced in one query.
QFT (2-qubit)	2	Quantum Fourier Transform — core subroutine of Shor's algorithm.
Grover's Search	2	Searches for |11> in 2-qubit space with amplitude amplification.
Bit-Flip Code	3	3-qubit repetition code protecting |1> against single bit-flip errors.

8. OpenQASM Import/Export

Exporting

Click **Export** in the OpenQASM section (bottom of the right panel) to convert your current circuit into OpenQASM 2.0 format. You can copy this code to use in Qiskit, IBM Quantum Experience, or any other QASM-compatible tool.

Importing

Paste OpenQASM 2.0 code into the text area and click **Import**. The circuit builder will parse the QASM and reconstruct the circuit visually. The qubit count automatically adjusts to match the QASM definition.

Example QASM (Bell State):

```
OPENQASM 2.0; include "qelib1.inc"; qreg q[2]; h q[0]; cx q[0],q[1];
```

9. Keyboard Shortcuts

The dashboard supports keyboard shortcuts for fast interaction. These only work when you are not typing in an input field or text area.

Key	Action
R	Run simulation
S	Enter step mode
C	Clear circuit
Ctrl+Z	Undo last gate
H	Select Hadamard gate
X	Select Pauli-X gate
Y	Select Pauli-Y gate
M	Select Measure
Arrow Right	Next step (in step mode)
Arrow Left	Previous step (in step mode)
Escape	Exit step mode / close dialogs
Enter	Confirm parameter dialog

10. Gate Reference

Single-Qubit Fixed Gates

Gate	Matrix	Effect
H (Hadamard)	1/sqrt(2) [[1,1],[1,-1]]	Creates equal superposition. Maps |0> to |+>, |1> to |->
X (Pauli-X)	[[0,1],[1,0]]	Bit flip (quantum NOT). Maps |0> to |1>, |1> to |0>
Y (Pauli-Y)	[[0,-i],[i,0]]	Bit+phase flip. Maps |0> to i|1>, |1> to -i|0>
Z (Pauli-Z)	[[1,0],[0,-1]]	Phase flip. Maps |1> to -|1>, leaves |0> unchanged
S	[[1,0],[0,i]]	sqrt(Z). Adds pi/2 phase to |1>
T	[[1,0],[0,e^(i*pi/4)]]	pi/8 gate. Adds pi/4 phase to |1>
sqrt(X)	(1/2)[[1+i,1-i],[1-i,1+i]]	Square root of X gate

Rotation Gates

Gate	Definition	Generator
Rx(theta)	exp(-i*theta*X/2)	Pauli X — rotates around x-axis on Bloch sphere
Ry(theta)	exp(-i*theta*Y/2)	Pauli Y — rotates around y-axis on Bloch sphere
Rz(theta)	exp(-i*theta*Z/2)	Pauli Z — rotates around z-axis on Bloch sphere
P(theta)	[[1,0],[0,e^(i*theta)]]	Phase gate — adds relative phase theta to |1>

Multi-Qubit Gates

Gate	Qubits	Description
CX (CNOT)	2	Controlled-X: flips target if control is |1>. Creates entanglement.
CZ	2	Controlled-Z: applies Z to target if control is |1>.
SWAP	2	Exchanges the states of two qubits.
CRz(theta)	2	Controlled Rz: applies Rz(theta) to target if control is |1>.
CCX (Toffoli)	3	Double-controlled X: flips target if both controls are |1>.
CSWAP (Fredkin)	3	Controlled SWAP: swaps targets if control is |1>.

11. Quantum Concepts Quick Reference

Superposition

A qubit can be in a combination of |0> and |1> simultaneously: |psi> = alpha|0> + beta|1>, where $|alpha|^2$ + $|beta|^2$ = 1. The Hadamard gate creates equal superposition from |0>.

Entanglement

Two qubits are entangled when the state of one cannot be described independently of the other. The Bell state (|00> + |11>)/sqrt(2) is the simplest example — measuring one qubit instantly determines the other's outcome, regardless of distance.

Measurement

Measuring a qubit collapses its superposition into a definite |0> or |1> outcome. The probability of each outcome equals $|amplitude|^2$. After measurement, the qubit is no longer in superposition. In the simulator, the statevector gives exact probabilities while the histogram shows statistical sampling of those probabilities.

Quantum Interference

Quantum amplitudes are complex numbers that can add (constructive interference) or cancel (destructive interference). Algorithms like Grover's Search exploit interference to amplify the amplitude of correct answers and suppress wrong ones.

No-Cloning Theorem

An unknown quantum state cannot be perfectly copied. This is why quantum teleportation requires destroying the original state, and why quantum key distribution (BB84) is secure.

12. Troubleshooting

Problem	Solution
Browser doesn't open	Navigate manually to http://127.0.0.1:8888
'ModuleNotFoundError: flask'	Run: pip install flask
Simulation returns error	Check that all multi-qubit gates have valid qubit indices
Bloch sphere empty	Click Run to simulate — results don't auto-update yet
QASM import fails	Ensure format is OPENQASM 2.0 with 'qreg q[N];' declaration
Port 8888 in use	Change port: launch(port=9999)
Gates won't place	Make sure a gate is selected (cyan border in palette)

---

tiny-qpu Interactive Quantum Lab is an open-source project.
GitHub: github.com/SKBiswas1998/tiny-qpu