

Bonus Track (un hello world para los más techies)

No me gustaría terminar el post sin incluir un mínimo guiño técnico a aquellos lectores más techies que quieran probar “algo de verdad” ¿Te gustaría ejecutar código en un ordenador cuántico real desde tu casa? Aquí te guío paso a paso para que en menos de 15 minutos estés manipulando *qubits* en un laboratorio de IBM en Nueva York. Sí, has leído bien - vas a controlar átomos individuales a miles de kilómetros de distancia.

Paso 1: Regístrate en IBM Quantum

1. Ve a <https://quantum.ibm.com/>
2. Haz clic en "Sign up" y crea una cuenta gratuita
3. Confirma tu email
4. Una vez dentro, ve a tu perfil (icono arriba a la derecha) → "Account settings"
5. Copia tu **API Token** (lo necesitarás en el paso 3)

Paso 2: Prepara tu entorno

Abre una terminal o símbolo del sistema y ejecuta:

```
# Instalar Qiskit - el SDK de IBM para computación cuántica
pip install qiskit qiskit-ibm-runtime
# Verificar que se instaló correctamente
python -c "import qiskit; print(f'Qiskit {qiskit.__version__}
instalado correctamente!')"
```

Paso 3: Tu primer programa cuántico

Crea un archivo llamado `mi_primer_quantum.py` y copia este código:

```
"""
Mi primer programa cuántico - Generador cuántico de números
aleatorios
Este programa demuestra la verdadera aleatoriedad cuántica
"""

from qiskit import QuantumCircuit
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2
import matplotlib.pyplot as plt

# =====
# PARTE 1: CONFIGURACIÓN
# =====
```

```
# IMPORTANTE: Sustituye 'TU_TOKEN_AQUI' con tu token de IBM
Quantum
# Solo la primera vez - después se guarda automáticamente
try:
    service = QiskitRuntimeService()
    print("☑ Cuenta ya configurada")
except:
    service = QiskitRuntimeService.save_account(
        channel="ibm_quantum",
        token="TU_TOKEN_AQUI", # <-- PEGA TU TOKEN AQUÍ
        set_as_default=True,
        overwrite=True
    )
    print("☑ Cuenta configurada por primera vez")

# =====
# PARTE 2: CREAR EL CIRCUITO CUÁNTICO
# =====

print("\n Construyendo circuito cuántico...")

# Crear un circuito con 2 qubits (para generar números del 0 al 3)
circuito = QuantumCircuit(2)

# Aplicar puertas Hadamard - crear superposición cuántica
# Cada qubit ahora está 50% en |0> y 50% en |1> simultáneamente
circuito.h(0) # Primer qubit en superposición
circuito.h(1) # Segundo qubit en superposición

# Añadir mediciones para colapsar la superposición
circuito.measure_all()

# Visualizar el circuito
print("\n Tu circuito cuántico:")
print(circuito.draw())

# =====
# PARTE 3: EJECUTAR EN SIMULADOR LOCAL
# =====

print("\n Ejecutando en simulador local...")
```

```
# Usar el simulador local (en tu ordenador)
from qiskit_ibm_runtime.fake_provider import FakeLimaV2
backend_simulado = FakeLimaV2()

# Ejecutar 1000 veces
sampler = SamplerV2(backend_simulado)
job_sim = sampler.run([circuito], shots=1000)
resultado_sim = job_sim.result()

# Mostrar resultados del simulador
print("\n📊 Resultados del simulador:")
counts_sim = resultado_sim[0].data.meas.get_counts()
for estado, cantidad in sorted(counts_sim.items()):
    decimal = int(estado, 2) # Convertir binario a decimal
    porcentaje = (cantidad/1000)*100
    print(f" Estado |{estado}> (número {decimal}): {cantidad} veces ({porcentaje:.1f}%)")

# =====
# PARTE 4: EJECUTAR EN HARDWARE CUÁNTICO REAL
# =====

print("\n Preparando ejecución en ordenador cuántico REAL...")

# Obtener el ordenador cuántico menos ocupado
backend_real = service.least_busy(
    operational=True,
    simulator=False,
    min_num_qubits=2
)

print(f" Seleccionado: {backend_real.name}")
print(f"   - Ubicación: {backend_real.backend_name}")
print(f"   - Número de qubits: {backend_real.num_qubits}")
print(f"   - Cola actual: {backend_real.status().pending_jobs} trabajos esperando")

# Ejecutar en el hardware real
print("\n Enviando tu código al ordenador cuántico...")
print("   (Esto puede tardar 1-5 minutos dependiendo de la cola)")
```

```
sampler_real = SamplerV2(backend_real)
job_real = sampler_real.run([circuito], shots=1000)

# Esperar resultados
print(f"\n Tu trabajo ID: {job_real.job_id()}")
print("    Esperando respuesta del hardware cuántico...")

resultado_real = job_real.result()

# =====
# PARTE 5: COMPARAR RESULTADOS
# =====

print("\n ¡ÉXITO! Resultados del ordenador cuántico REAL:")
counts_real = resultado_real[0].data.meas.get_counts()

# Mostrar comparación
print("\n COMPARACIÓN SIMULADOR vs HARDWARE REAL:")
print("-" * 50)
print("Estado | Simulador | Hardware Real | Diferencia")
print("-" * 50)

for estado in ['00', '01', '10', '11']:
    sim_count = counts_sim.get(estado, 0)
    real_count = counts_real.get(estado, 0)
    diferencia = abs(sim_count - real_count)
    decimal = int(estado, 2)
    print(f"|{estado}|({decimal}) | {sim_count:4d} | {real_count:4d} | {diferencia:3d}")

# =====
# PARTE 6: VISUALIZACIÓN
# =====

# Crear gráfico de comparación
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# Gráfico del simulador
estados = list(counts_sim.keys())
valores_sim = list(counts_sim.values())
ax1.bar(estados, valores_sim, color='blue', alpha=0.7)
ax1.set_title('Simulador (Perfecto)')
```

```
ax1.set_xlabel('Estado Cuántico')
ax1.set_ylabel('Mediciones')
ax1.axhline(y=250, color='r', linestyle='--', label='Esperado (250)')
ax1.legend()
```

Gráfico del hardware real

```
valores_real = [counts_real.get(e, 0) for e in estados]
ax2.bar(estados, valores_real, color='green', alpha=0.7)
ax2.set_title('Hardware Cuántico Real (Con Ruido)')
ax2.set_xlabel('Estado Cuántico')
ax2.set_ylabel('Mediciones')
ax2.axhline(y=250, color='r', linestyle='--', label='Esperado (250)')
ax2.legend()
```

```
plt.suptitle(' Tu Generador Cuántico de Números Aleatorios',
fontsize=16)
plt.tight_layout()
plt.show()
```

```
# =====
# MENSAJE FINAL
# =====
```

```
print("\n" + "="*60)
print(" ¡FELICIDADES!")
print("="*60)
print("""
```

Has ejecutado con éxito tu primer programa en un ordenador cuántico real.

Lo que acabas de hacer:

- ✓ Creaste una superposición cuántica con 2 qubits
- ✓ Generaste verdadera aleatoriedad cuántica (imposible clásicamente)
- ✓ Ejecutaste código en átomos superconductores a -273°C
- ✓ Observaste el ruido cuántico en hardware real

El ruido que ves en los resultados reales no es un error - es la manifestación física de la decoherencia cuántica, uno de los mayores desafíos de esta tecnología.

```
¡Bienvenido/a a la era de la computación cuántica!  
""")
```

Paso 4: ¡Ejecútalo!

```
python mi_primer_quantum.py
```

¿Qué verás?

1. **Primero:** el circuito cuántico visualizado en ASCII
2. **Después:** resultados instantáneos del simulador (perfectos)
3. **Luego:** tu código viajando a un laboratorio de IBM
4. **Finalmente:** los resultados reales con ruido cuántico
5. **Extra:** gráficos comparando simulador vs realidad

Resultado esperado:

Comparación entre simulador vs ordenador cuántico real.

Estado	Simulador	Hardware Real	Diferencia

00>(0)	248	241	7
01>(1)	251	259	8
10>(2)	249	243	6
11>(3)	252	257	5

¿Qué acabas de hacer realmente?

Has creado un **generador cuántico de números aleatorios verdaderos**. A diferencia de los números "aleatorios" de tu ordenador (que son pseudoaleatorios), estos números son fundamentalmente impredecibles según las leyes de la física cuántica. Bancos y sistemas de criptografía pagan mucho dinero por este tipo de aleatoriedad verdadera.

Para los más aventureros:

Modifica el número de *qubits* de 2 a 3 y tendrás números del 0 al 7. Pero cuidado - cada *qubit* adicional duplica el rango de los posibles números generados. Con 10 *qubits* generarías números hasta 1.024. Con 20 *qubits*, hasta más de un millón.

¡Espero que te haya gustado tu primer "hola mundo" en computación cuántica!