Quantum teleportation

Autores:

- Blanca Cano Camarero,
- Iker Villegas Labairu.

Fecha: 11 de Diciembre de 2022.

Introducción

La teleportación cuántica es una técnica para moverse por los estados cuánticos aunque no exista un canal de enlace entre el estado cuántico y su receptor.

Por ejemplo, Alice quiere enviar información cuántica para Bob. Supongamos que se trata del estado qubit

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

Transmitirle la información consistiría en enviarle información a Bob sobre α y β.

Sin embargo el **Teorema de no clonación** establece que

No existe operador unitario U actuando sobre $\mathcal{H}_A \otimes \mathcal{H}_B$ de forma que para todos los estados normalizados $|\phi\rangle_A$ y $|e\rangle_B$ en $\mathcal{H}$ cumplan que

$$U|\phi\rangle_A|e\rangle_B = |\phi\rangle_A|\phi\rangle_B$$

Es decir, no podemos hacer una copia exacta de un estado cuántico desconocido, solo podremos copiar los estados clásicos (no las superposiciones).

Un poco de contexto

El primero de los artículos científicos al respecto fue *Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels* publicado por C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters en 1993.

Se realizó experimentalmente en 1997 por dos grupos de investigadores Sandu Popescu y Anton Zeilinger respectivamente.

Protocolo de Teleportación Cuántica

Requisitos previos

Para transferir un qubit, Alice y Bob usaran un tercer canal (Telamon en la notación de IBM) para enviar un par de qubit entrelazados.

Además, para la correcta comprensión del algoritmo, se requerirá el uso de los estados de Bell. En primer lugar, una puerta de Hadamard nos permite pasar del estado $|00\rangle$ a $\frac{(|0\rangle+|1\rangle)|0\rangle}{\sqrt{2}}$. Y si aplicamos a continuación un CNOT, obtendremos el estado $\frac{(|00\rangle+|11\rangle)}{\sqrt{2}}$. Esta transformación de estados resultante, es la que denominaremos como estado de Bell, y lo denotaremos como $|\beta_{00}\rangle$. De este modo, los estados posibles son:

$$|\beta_{00}\rangle := \frac{(|00\rangle + |11\rangle)}{\sqrt{2}}$$

$$|\beta_{01}\rangle := \frac{(|01\rangle + |10\rangle)}{\sqrt{2}}$$

$$|\beta_{10}\rangle := \frac{(|00\rangle - |11\rangle)}{\sqrt{2}}$$

$$|\beta_{11}\rangle := \frac{(|01\rangle - |10\rangle)}{\sqrt{2}}$$

Desde un punto de vista general los estados de Bell los podemos resumir en la siguiente definición:

$$|\beta_{xy}\rangle = \frac{|0, y\rangle + (-1)^x|1, \overline{y}\rangle}{\sqrt{2}}$$

donde $\overline{y}$ es la negación de y.

Paso generales

1. Un estado de Bell es generado cuando se envía un qubit de A a B.
2. Una medida de Bell del estado cuántico y de el qubit que se va a teletransportar ($|\phi\rangle$) es realizado en A y esto permite que uno de los cuatro estados que se van a medir puedan ser codificados en dos bits clásicos de información.
3. Usando el canal tradicional los dos bits se mandan de A a B.
4. Como resultado de la medida realizada en A, el estado de Bell está en B. Uno de esos cuatro estados es idéntico al estado de $|\phi\rangle$ y los otros tres se le parecen.

Formulación formal

Sea $|\psi\rangle$ el qubit de Alice que quiere enviar a Bob.

El qubit puede escribirse como

$$|\psi\rangle_C = \alpha|0\rangle_C + \beta 1\rangle_C.$$

Donde el subíndice C se usa para distinguir este estado de A a B.

Paso previos de Alice

Puerta CNOT

Alice aplicará una puerta CNOT a sus dos qubits. De esta manera uno de los qubit nos servirá como control mientras que el otro como objetivo.

Recordemos que aplicar una puerta CNOT se puede ver como aplicar un operador de estados, matriz

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

De esta manera

$$|\psi_1\rangle = CNOT|\psi_0\rangle = \frac{1}{2}(\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)).$$

Operador de Hadamard

A continuación Alice aplicará el operador de Hadamard

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Transmisión

El protocolo requiere ahora que Alice y Bob compartan un estado máximo entrelazado. Este será uno de estos cuatro (no importa cual)

$$|\phi^+\rangle_{AB} = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B),$$
$$|\psi^+\rangle_{AB} = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B),$$
$$|\psi^-\rangle_{AB} = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B),$$
$$|\phi^-\rangle_{AB} = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B).$$

Sin pérdida de generalidad supondremos que el estado compartido ha sido $|\phi^+\rangle_{AB}$, donde el subíndice A y B en el estado entrelazado indica si ese estado pertenece a la partícula de Alice o a la de Bob. Recordemos que C era el estado que Alice pretendía teletransportar.

Alice toma una de las partículas del par, entonces tiene dos, la C que es la que quiere teletransportar y ya las ha entrelazado en los dos pasos previos A. Bob tiene una.

El estado total es el siguiente:

$$|\psi\rangle_C \otimes |\phi^+\rangle_{AB} = (\alpha|0\rangle_C + \beta|1\rangle_C) \otimes \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B).$$

Alice hace una medición local en los cuatro estados en las dos partículas en su posesión, que procedemos a escribir para que quede claro:

$$|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|\phi^+\rangle + |\phi^-\rangle)$$
$$|0\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}(|\psi^+\rangle + |\psi^-\rangle)$$
$$|1\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|\psi^+\rangle - |\psi^-\rangle)$$
$$|1\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}(|\phi^+\rangle - |\phi^-\rangle)$$

Después de expandir la expresión para

$$|\psi\rangle_C \otimes |\phi^+\rangle_{AB},$$

se aplican las identidades a los qubits con subíndices A y C.
En nuestro caso particular esto sería:

$$\alpha\frac{1}{\sqrt{2}}|0\rangle_C \otimes |0\rangle_A \otimes |0\rangle_B = \alpha\frac{1}{2}(|\phi^+\rangle_{CA} + |\phi^-\rangle_{CA}).$$

(Para los otros términos se haría de manera similar).

Combinando estos términos, los tres estados de partícula de A, B y C juntos obtenemos la siguiente ecuación del estado de superposición:

$$
\begin{aligned}
|\psi\rangle_C \otimes |\phi^+\rangle_{AB} = \frac{1}{2}(&|\phi^+\rangle_{CA} \otimes (\alpha|0\rangle_B + \beta|1\rangle_B) \\
&+|\phi^-\rangle_{CA} \otimes (\alpha|0\rangle_B - \beta|1\rangle_B) \\
&+|\psi^+\rangle_{CA} \otimes (\alpha|1\rangle_B + \beta|0\rangle_B) \\
&+|\psi^-\rangle_{CA} \otimes (\alpha|1\rangle_B - \beta|0\rangle_B) \\
&).
\end{aligned}
$$

Notemos qe las partículas siguen igual ya que no hemos aplicado ninguna transformación a las operaciones.

La teletrasportación ocurre cuando Alice mide uno de sus qubits A y C en la base de Bell.

$$|\phi^+\rangle_{CA}, |\phi^-\rangle_{CA}, |\psi^+\rangle_{CA}, |\psi^-\rangle_{CA}.$$

Equivalentemente la medida también puede darse en una base computacional con combinaciones

$$\{|0\rangle, |1\rangle\}$$

mapeando cada estado de Bell de manera única con un elemento de

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}.$$

De esta manera los estado de Alice colapsarán con igualdad de probabilidad en uno de los siguientes estados:

$$|\phi^+\rangle_{CA} \otimes (\alpha|0\rangle_B + \beta|1\rangle_B),$$

$$|\phi^-\rangle_{CA} \otimes (\alpha|0\rangle_B - \beta|1\rangle_B),$$

$$|\psi^+\rangle_{CA} \otimes (\alpha|1\rangle_B + \beta|0\rangle_B),$$

$$|\psi^-\rangle_{CA} \otimes (\alpha|1\rangle_B - \beta|0\rangle_B).$$

El entrelazamiento de las partículas de Alice y Bob se ha roto.

Bob toma ahora uno de los cuatro estados posibles que se le parece al estado que va a ser teletrasportado.

Como resultado de la medición de Alice, ella puede enviar el resultado de la medición a Bob por el canal de transmisión tradicional.

De esta forma, habiendo recibido tal información, Bob realizará las siguiente operaciones a su partícula para transformarla en el estado deseado.

- Si Alice indica que su resultado es $|\phi^+\rangle_{CA}$ entonces Bob sabe que su cubit está en el estado deseado.
- Si el mensaje indica que $|\phi^-\rangle_{CA}$ entonces deberá de transformarlo por la puerta cuántica.

$$\sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

para recuperar el estado.

- Si el mensaje de Alice se corresponde a $|\psi^+\rangle_{CA}$, Bob utilizará la puerta:

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

a su qubit.

- Finalmente la puerta apropiada es

$$\sigma_3\sigma_1 = -\sigma_1\sigma_3 = i\sigma_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Después de esta operación, el quibit de Bob estará en el estado

$$|\psi\rangle_B = \alpha|0\rangle_B + \beta|1\rangle_B.$$

Notemos que la teleportación no consiste en copiar los qubits y por tanto es consistente con el teorema de la no clonación.

Esto se puede generalizar, por cada qubit teletransportado Alice necesita enviar a Bob dos bits de información clásicos.

Ejemplo práctico

Ahora vamos a llevar a cabo una prueba práctica del algoritmo de teleportación. Para ello hemos llevado a cabo, a través de un cuaderno de jupyter, un experimento a través del cuál se demuestra como se realiza el paso de la información desde el qubit de Alice al de Bob, desde un punto de vista práctico.

Dicho código puede ejecutarse desde IBM QuantumLab.

In [ ]:

```
from ibm_quantum_widgets import CircuitComposer
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi
from qiskit import QuantumCircuit, execute, Aer, IBMQ
from qiskit.compiler import transpile, assemble
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from qiskit.extensions import Initialize
from qiskit.quantum_info import random_statevector
from qiskit.ignis.verification import marginal_counts
```

Comenzamos almacenando en variables los diferentes métodos de ejecución del circuito para su posterior uso:

In [2]:

```
backend_unitary = Aer.get_backend('unitary_simulator')
backend_qasm = Aer.get_backend('qasm_simulator')
backend_stvec = Aer.get_backend('statevector_simulator')
backend_sim = Aer.get_backend('aer_simulator')
```

Comenzamos construyendo el circuito donde los qbits q_0 y q_1 se corresponden con Alice, mientras que el qbit q_2 será para Bob.

In [3]:

```
qreg_q = QuantumRegister(3, 'q')
creg_c = ClassicalRegister(2, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)
```

A continuación, vamos a inicializar el qubit q_0. Para ello le asignaremos un valor aleatorio como vector de estados. Nuestro objetivo será ver que dicho valor se transporta correctamente al qubit q_2.

In [4]:

```
#Sacamos un vector aleatorio que inicialice el qbit q0
init_value = random_statevector(2)

#Creamos la puerta que inicializará el qbit
init_gate = Initialize(init_value)
init_gate.label = "init_value"

#Añadimos el la puerta que inicializa el qbit q0 a nuestro circuito
circuit.append(init_gate, [0])
circuit.barrier()
```

Out[4]:

```
<qiskit.circuit.instructionset.InstructionSet at 0x1c73b77c8b0>
```

El valor con el cuál inicializamos q_0 será el siguiente

In [5]:

```
display(array_to_latex(init_value, prefix="|\\psi\\rangle ="))
```

$$|\psi\rangle = \begin{bmatrix} 0.41646 + 0.76195i & -0.02158 - 0.49551i \end{bmatrix}$$

Para el segundo y tercer qbit será necesario aplicar en estado de Bell. Para ello introduciremos una puerta de Hadamard y otra CNOT en este orden.

In [6]:
```
circuit.h(qreg_q[1])
circuit.cx(qreg_q[1], qreg_q[2])

circuit.draw()
```

Out[6]:
```
     ┌──────────────────────────────────────────────────┐ ░                 
q_0: ┤ init_value(0.41646+0.76195j,-0.021579-0.49551j) ├─░─────────────────
     └──────────────────────────────────────────────────┘ ░ ┌───┐          
q_1: ─────────────────────────────────────────────────────░─┤ H ├──■───────
                                                          ░ └───┘┌─┴─┐     
q_2: ─────────────────────────────────────────────────────░──────┤ X ├─────
                                                          ░      └───┘     
c_0: ══════════════════════════════════════════════════════════════════════

c_1: ══════════════════════════════════════════════════════════════════════
```

A continuación, para poder llevar a cabo el algoritmo de teleportación, aplicamos en los qbits pertenecientes a Alice una transformación de la siguiente manera: una puerta CNOT para q_0 y q_1, y una de Hadamard para el qbit q_0.

In [7]:
```
circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2])
circuit.cx(qreg_q[0], qreg_q[1])
circuit.h(qreg_q[0])
```

Out[7]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x1c73b77d4e0>
```

El circuito quedará de la siguiente forma:

In [8]:
```
circuit.draw()
```

Out[8]:

```
     ┌──────────────────────────────────────────────┐ ░
q_0: ┤ init_value(0.41646+0.76195j,-0.021579-0.49551j) ├─░──────────
     └──────────────────────────────────────────────┘ ░ ┌───┐
q_1: ───────────────────────────────────────────────────░─┤ H ├──■──
                                                        ░ └───┘┌─┴─┐
q_2: ───────────────────────────────────────────────────░──────┤ X ├
                                                        ░      └───┘
c_0: ════════════════════════════════════════════════════════════════

c_1: ════════════════════════════════════════════════════════════════

«
«     ┌───┐
«q_0: ┤ H ├
«     └───┘
«q_1: ─────
«
«q_2: ─────
«
«c_0: ═════
«
«c_1: ═════
«
```

Veamos la matriz que interviene en nuestro circuito.

In [9]:

```python
unitary= execute(
    circuit,
    backend_unitary,
    shots=1024).result().get_unitary()
array_to_latex(unitary)
```

Out[9]:

$$\begin{bmatrix}
0.20823 + 0.38097i & -0.11894 - 0.21761i & 0.20823 + 0.38097i & -0.11894 \\
0.20823 + 0.38097i & -0.11894 - 0.21761i & 0.20823 + 0.38097i & -0.11894 \\
-0.01079 - 0.24775i & -0.01889 - 0.43376i & -0.01079 - 0.24775i & -0.01889 \\
0.01079 + 0.24775i & 0.01889 + 0.43376i & 0.01079 + 0.24775i & 0.01889 - \\
-0.01079 - 0.24775i & -0.01889 - 0.43376i & 0.01079 + 0.24775i & 0.01889 - \\
0.01079 + 0.24775i & 0.01889 + 0.43376i & -0.01079 - 0.24775i & -0.01889 \\
0.20823 + 0.38097i & -0.11894 - 0.21761i & -0.20823 - 0.38097i & 0.11894 - \\
0.20823 + 0.38097i & -0.11894 - 0.21761i & -0.20823 - 0.38097i & 0.11894 -
\end{bmatrix}$$

Y este será el vector de estados final, tras aplicarse dicha parte del algoritmo:

In [10]:

```python
final_state= execute(
    circuit,
    backend_stvec,
    shots=1024).result().get_statevector()
array_to_latex(final_state)
```

Out[10]:

$$\begin{bmatrix} 0.20823 + 0.38097i & 0.20823 + 0.38097i & -0.01079 - 0.24775i & 0.01079 + 0.24 \end{bmatrix}$$

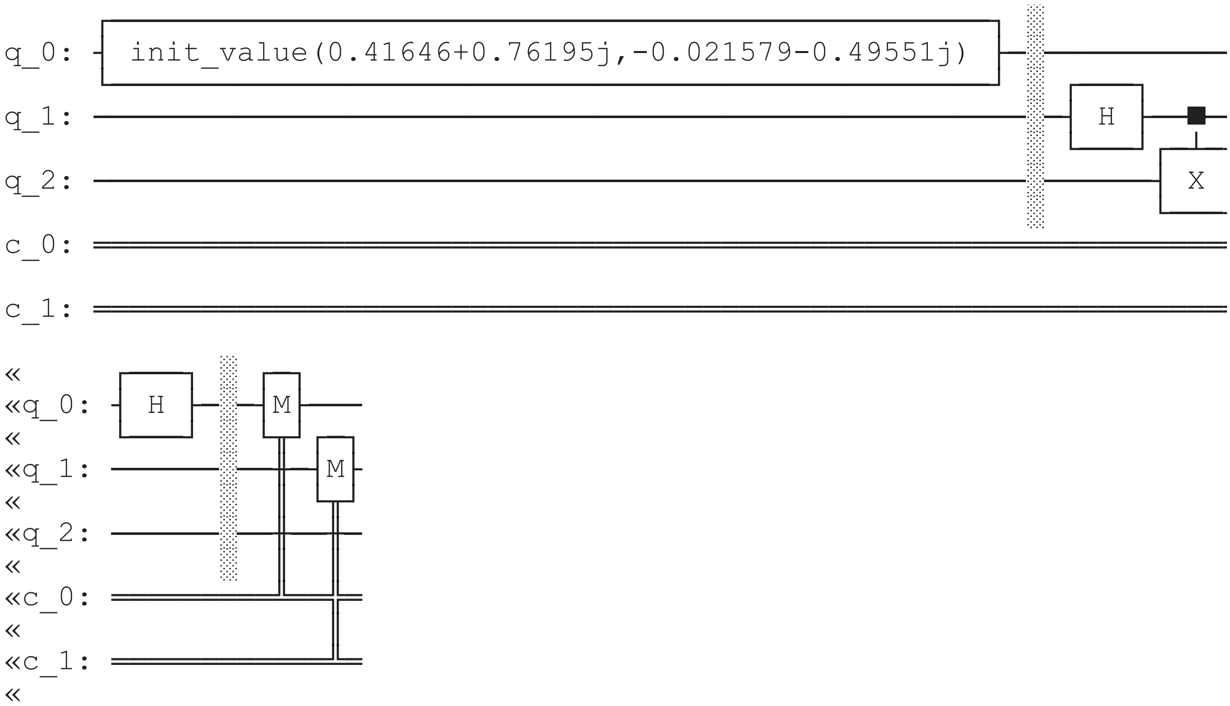
Introducimos dos medidores para q_0 y q_1 y una barrera para separar el circuito de dichos medidores.

In [11]:

```
circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2])
circuit.measure(qreg_q[0], creg_c[0])
circuit.measure(qreg_q[1], creg_c[1])

circuit.draw()
```

Out[11]:



Por último, para descodificar la información transmitida al qbit q_2, aplicaremos dos condicionales a las puertas NOT y Z. De modo que si tienen valor 1 se aplicarán al estado. Así, se seguirá el esquema siguiente:

$$00 \longrightarrow \text{no se aplica ninguna puerta cuántica}$$

$$01 \longrightarrow \text{se aplica la puerta cuántica } X = \sigma_3$$

$$10 \longrightarrow \text{se aplica la puerta cuántica } Z = \sigma_1$$

$$11 \longrightarrow \text{se aplican las puertas cuánticas } XZ = \sigma_3\sigma_1$$

In [12]:

```
circuit.barrier(qreg_q[0], qreg_q[1],qreg_q[2])
circuit.x(2).c_if(creg_c[1], 1)
circuit.z(2).c_if(creg_c[0], 1)

circuit.draw()
```

Out[12]:

```
     ┌──────────────────────────────────────────────┐ ░            
q_0: ┤ init_value(0.41646+0.76195j,-0.021579-0.49551j) ├─░────────────
     └──────────────────────────────────────────────┘ ░ ┌───┐     
q_1: ──────────────────────────────────────────────────░─┤ H ├──■──
                                                       ░ └───┘┌─┴─┐
q_2: ──────────────────────────────────────────────────░──────┤ X ├
                                                       ░      └───┘
c_0: ══════════════════════════════════════════════════════════════

c_1: ══════════════════════════════════════════════════════════════

«     ┌───┐ ░ ┌─┐    ░                
«q_0: ┤ H ├─░─┤M├────░────────────────
«     └───┘ ░ └╥┘┌─┐ ░                
«q_1: ──────░──╫─┤M├─░────────────────
«           ░  ║ └╥┘ ░ ┌───┐ ┌───┐ 
«q_2: ──────░──╫──╫──░─┤ X ├─┤ Z ├─
«           ░  ║  ║  ░ └─╥─┘ └─╥─┘ 
«c_0: ═════════╩══╬══════╬═════■═══
«                 ║      ║         
«c_1: ════════════╩══════■═════════
«                                  
```

Vamos a comprobar si se ha llevado correctamente el experimento. En un ordenador cuántico no es posible mostrar el vector de estados, así que para comprobar si todo ha ido bien utilizaremos otra forma. La puerta *init* que hemos aplicado al qbit q_0 al principio del algoritmo, nos ha permitido pasar del estado inicial de nuestro circuito $|0\rangle$, a un estado aleatorio a partir del cual ibamos a inicializar el algoritmo. De este modo, aplicaremos la inversa de esta puerta *init* al qbit q_2. Si nos devuelve el estado $|0\rangle$ con un 100 de probabilidad, podremos asegurar que toda la información del qubit q_0 se ha transportado correctamente a q_2.

Procederemos del siguiente modo: tras meter una barrera para separar esta nueva parte, añadiremos la puerta inversa a q_2 y un medidor.

In [13]:

```
circuit.barrier(qreg_q[0], qreg_q[1],qreg_q[2]) #barrera de separación

#Añadimos la puerta inversa
inverse_init = init_gate.gates_to_uncompute()
circuit.append(inverse_init, [2])

#Creamos un nuevo registro clásico donde se apoyará el nuevo medidor
c_2 = ClassicalRegister(1)
circuit.add_register(c_2)

#Añadimos el medidor para el qubit q2
circuit.measure(2,2)
```

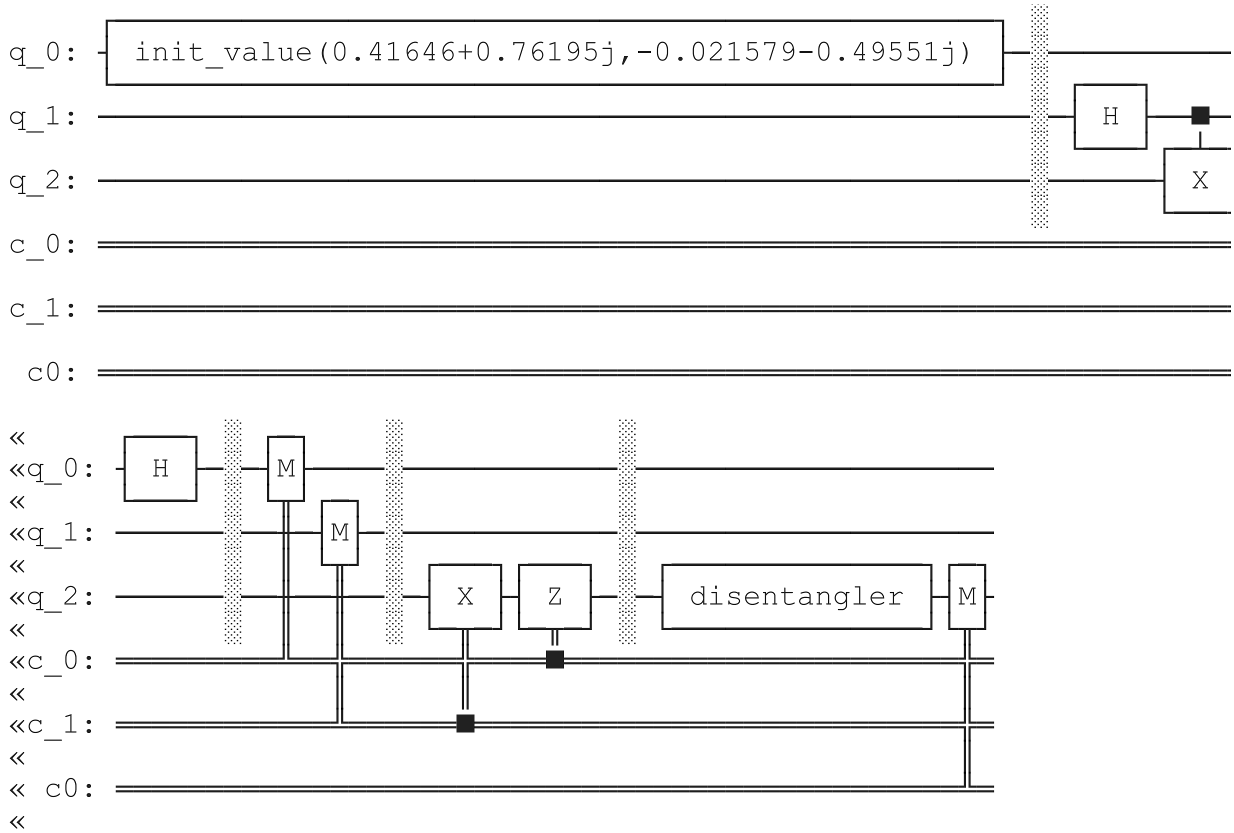
Out[13]:

```
<qiskit.circuit.instructionset.InstructionSet at 0x1c73b77de40>
```

In [14]:

```
circuit.draw()
```

Out[14]:



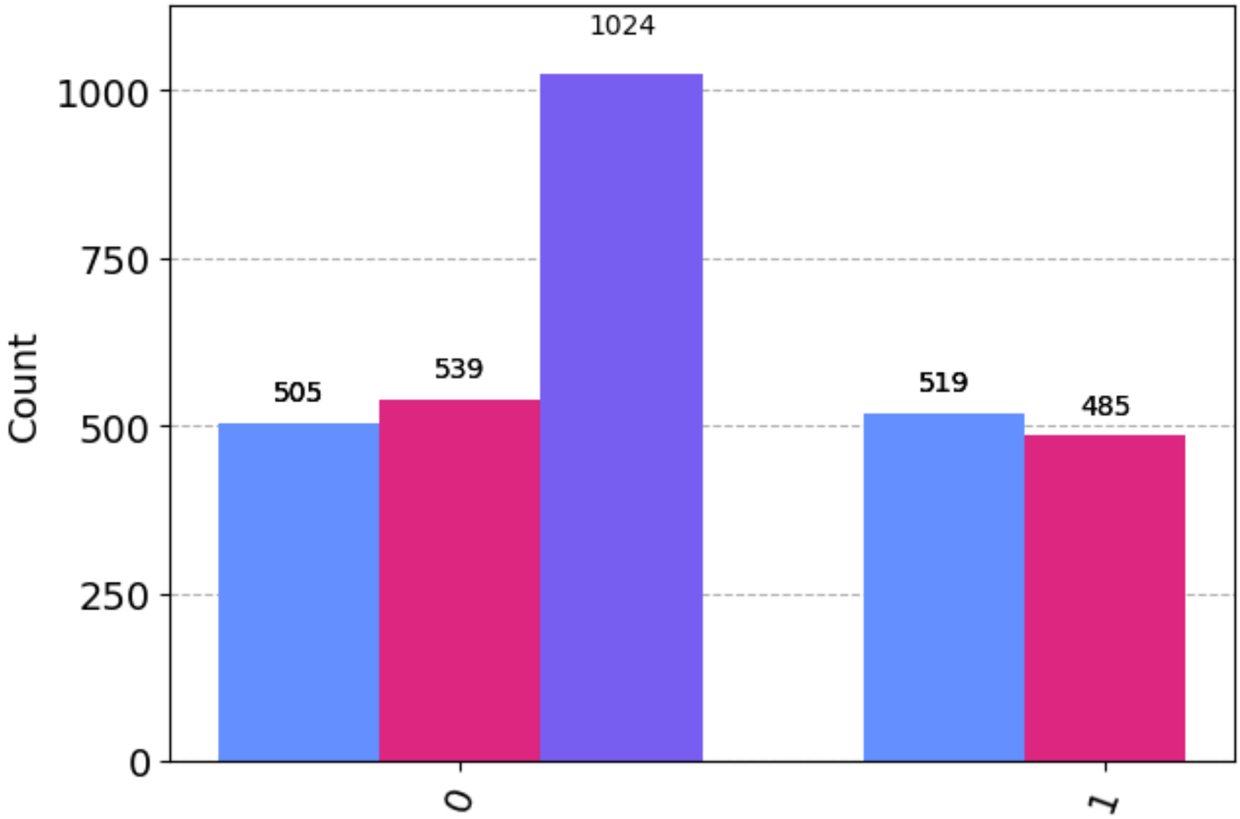
Llevamos a cabo la representación de nuestras mediciones a través de un histograma.

In [15]:

```
t_qc = transpile(circuit, sim)
t_qc.save_statevector()
counts = sim.run(t_qc).result().get_counts()
qubit_counts = [
    marginal_counts(counts, [qubit])
    for qubit in range(3)]
plot_histogram(qubit_counts)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_704504\2833224662.py:2: Deprec
ationWarning: The qiskit.ignis package is deprecated and has been supersced
ed by the qiskit-experiments project. Refer to the migration guide: https:/
/github.com/Qiskit/qiskit-ignis#migration-guide on how to migrate to the ne
w project.
  from qiskit.ignis.verification import marginal_counts
```

Out[15]:



Echando un ojo a nuestro histograma, vemos que todos los disparos se miden finalmente con estado $|0\rangle$ para el caso del qubit q_2 representado con la barra morada. Con esto podemos concluir que con un 100% de probabilidad nuestro experimento habrá transmitido toda la información de q_0 a q_2.

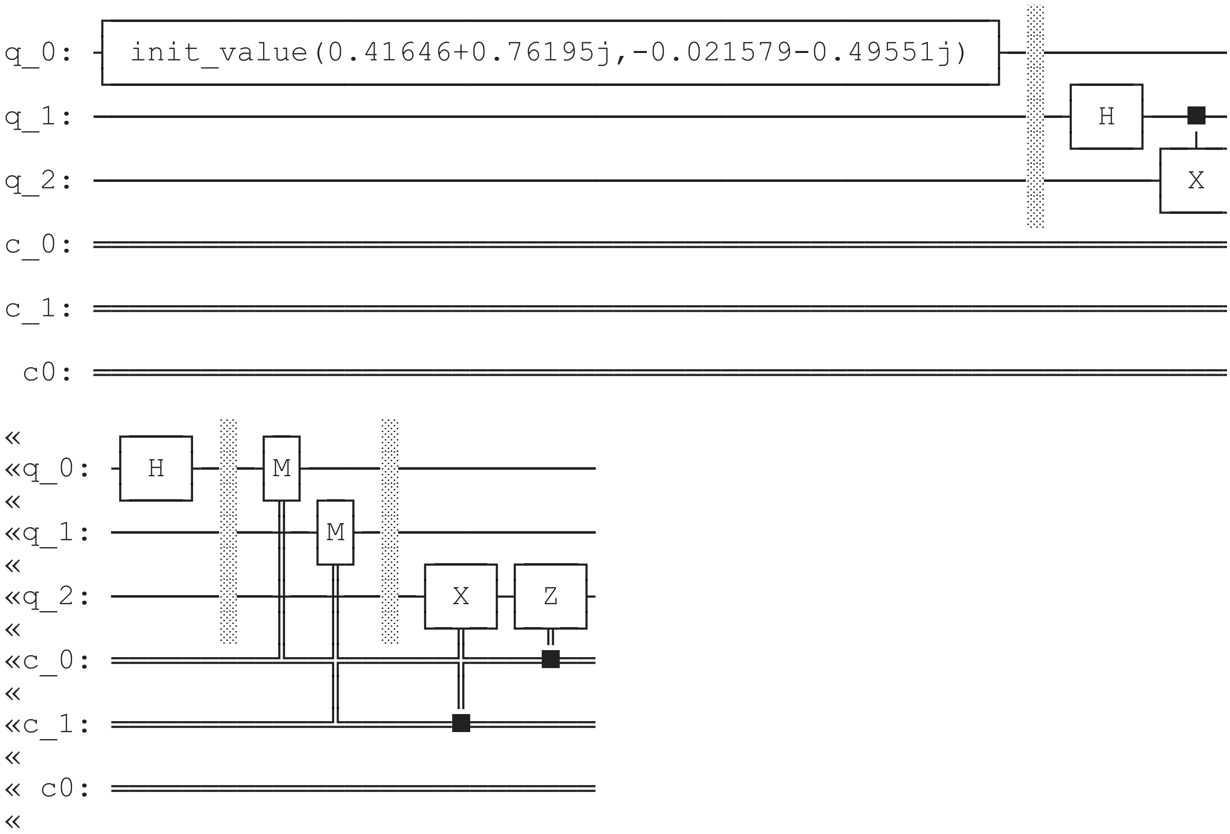
In [16]:

```
for i in range(3):
    circuit.data.pop(13) #borramos el vector de estados almacenado en nuest
```

In [17]:

```
circuit.draw()
```

Out[17]:



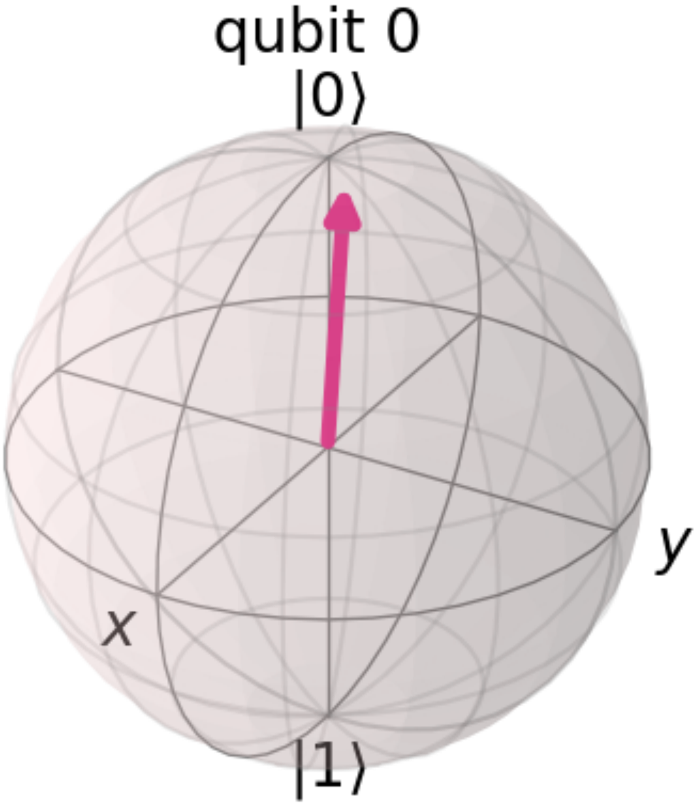
Ahora veamos graficamente si el vector de estados tras el algoritmo de q_2 correspondiente a la información que descodica Bob coincide con la inicialmente dispone Alice en q_0.

El vector inicial correspondiente a la información que quería transmitir Alice era el siguiente:

In [18]:

```
plot_bloch_multivector(init_value)
```

Out[18]:

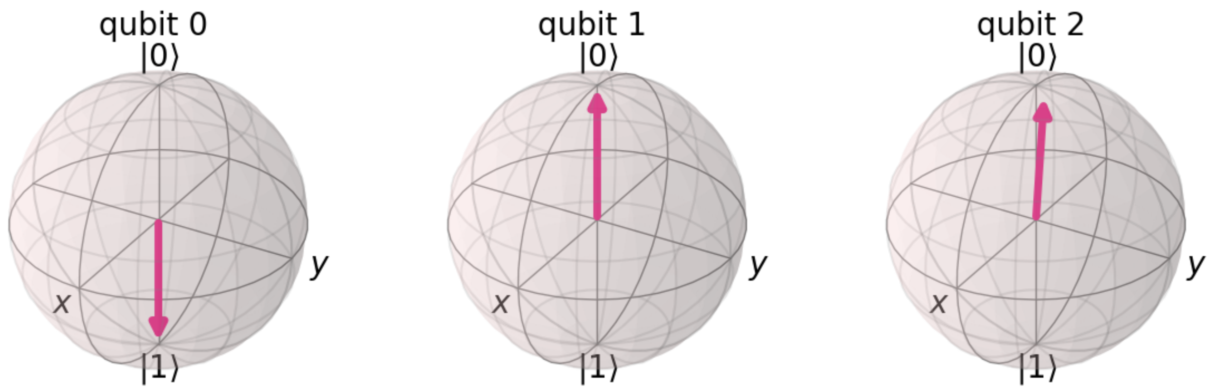


Comparándolo con los estados finales de los tres qbits, vemos que se ha transmitido correctamente al qbit q_2. Pues ambas representaciones coinciden.

In [19]:

```
circuit.save_statevector()
out_vector = backend_sim.run(circuit).result().get_statevector()
plot_bloch_multivector(out_vector)
```

Out[19]:



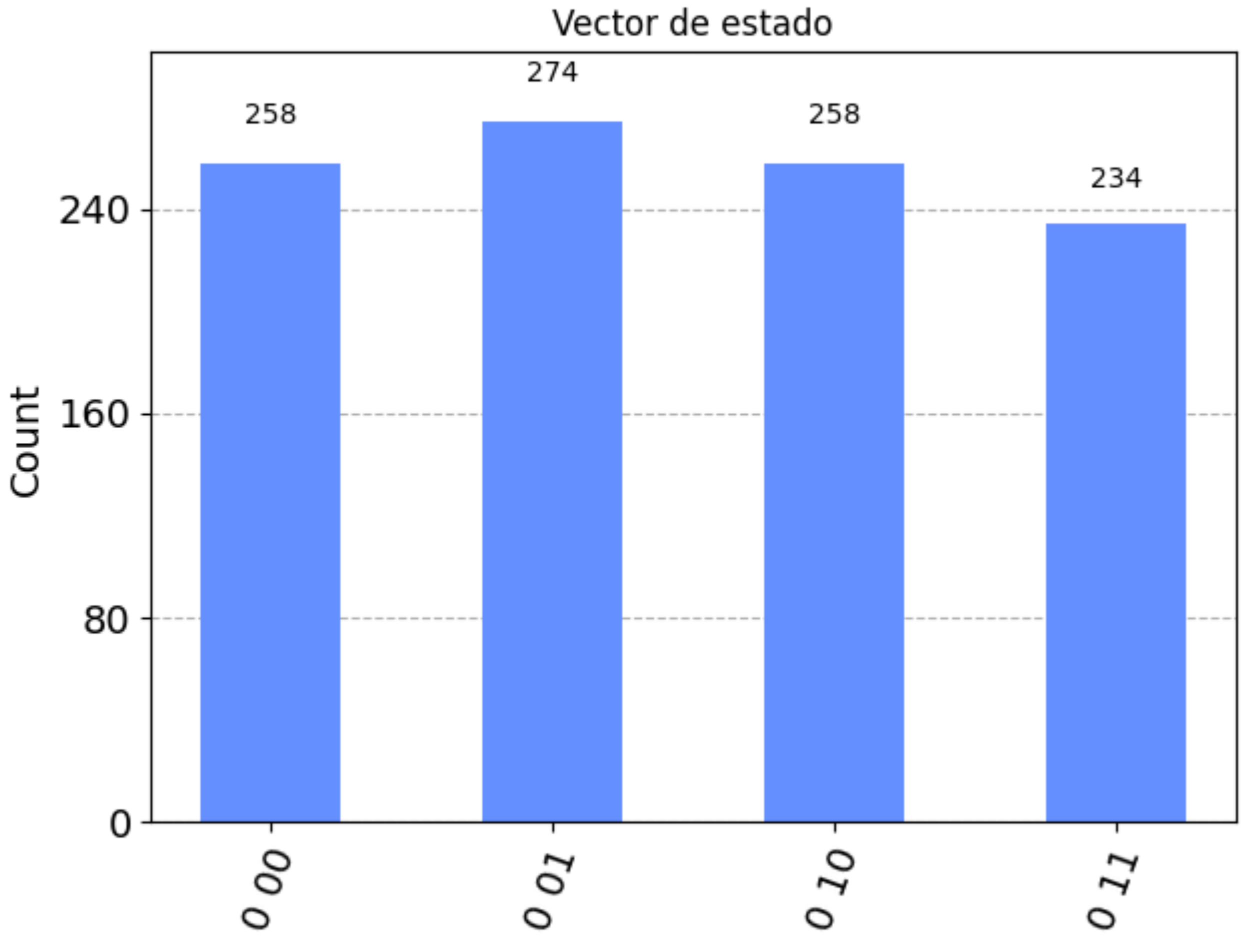
Veamos ahora desde un punto de vista gráfico, el valor que adopta el vector de estados para los tres qbits:

In [20]:

```
job = execute(circuit, backend_qasm, shots=1024)

plot_histogram(job.result().get_counts(), title="Vector de estado")
```

Out[20]:



Tal y como podemos observar, tras lanzar 1024 disparos a nuestro experimento, serán 4 los posibles estados finales, a pesar de tener 3 $qbits$. Esto se debe a lo visto anteriormente, pues el qbit q_2 adoptará siempre el mismo estado final. Y por lo tanto, los estados finales de los qbits restantes oscilarán entre estos cuatro valores con la misma probabilidad, aproximadamente.

Y de esta forma, hemos hecho una muestra práctica de cómo funciona el algoritmo de teleportación.

Fuentes

Todas las referencias web fueron consultadas por última vez con fecha Domingo 11 de Diciembre de 2022.

- Página 26 del libro Quantum Computation and Quantum Information 10th Anniversary Edition Michael A. Nielsen & Isaac L. Chuang
- https://es.wikipedia.org/wiki/Teorema_de_no_clonación
- https://qiskit.org/textbook/ch-algorithms/teleportation.html#3.1-How-Will-We-Test-the-Protocol-on-a-Quantum-Computer?-
- https://en.wikipedia.org/wiki/Quantum_teleportation
- Bennett, Charles H.; Brassard, Gilles; Crépeau, Claude; Jozsa, Richard; Peres, Asher; Wootters, William K. (29 March 1993). "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels". Physical Review Letters. 70 (13): 1895–1899. Bibcode:1993PhRvL..70.1895B.
- D. Boschi; S. Branca; F. De Martini; L. Hardy; S. Popescu (1998). "Experimental Realization of Teleporting an Unknown Pure Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels". Physical Review Letters. 80 (6): 1121–1125. arXiv:quant-ph/9710013. Bibcode:1998PhRvL..80.1121B.