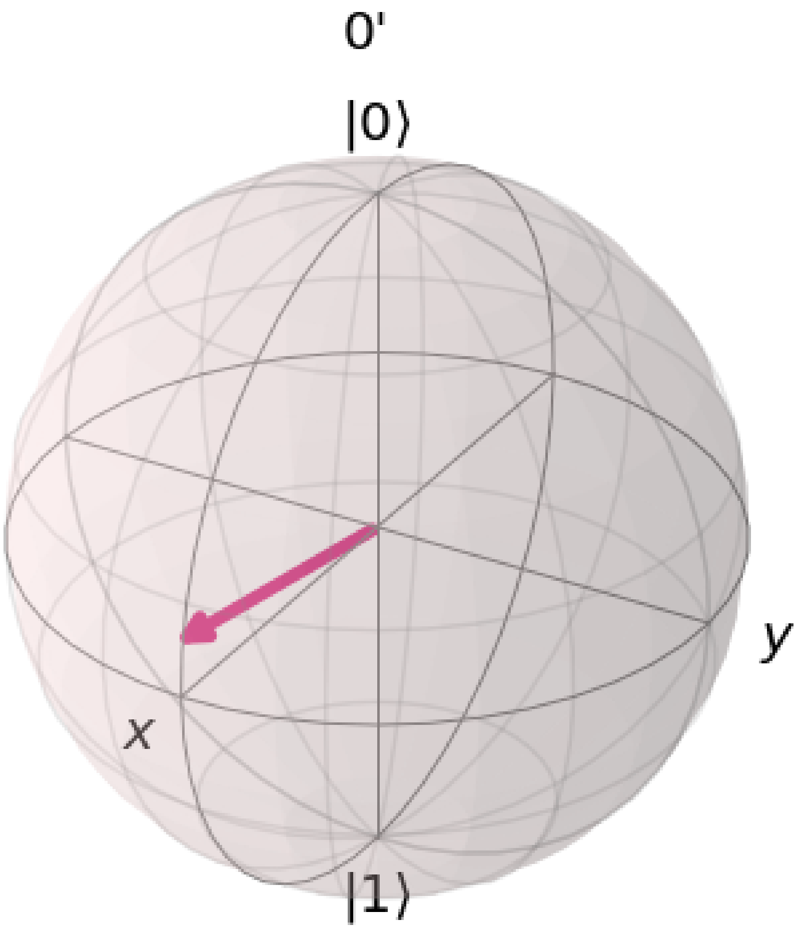
Trabalho_qiskit2

September 11, 2021

[1]:

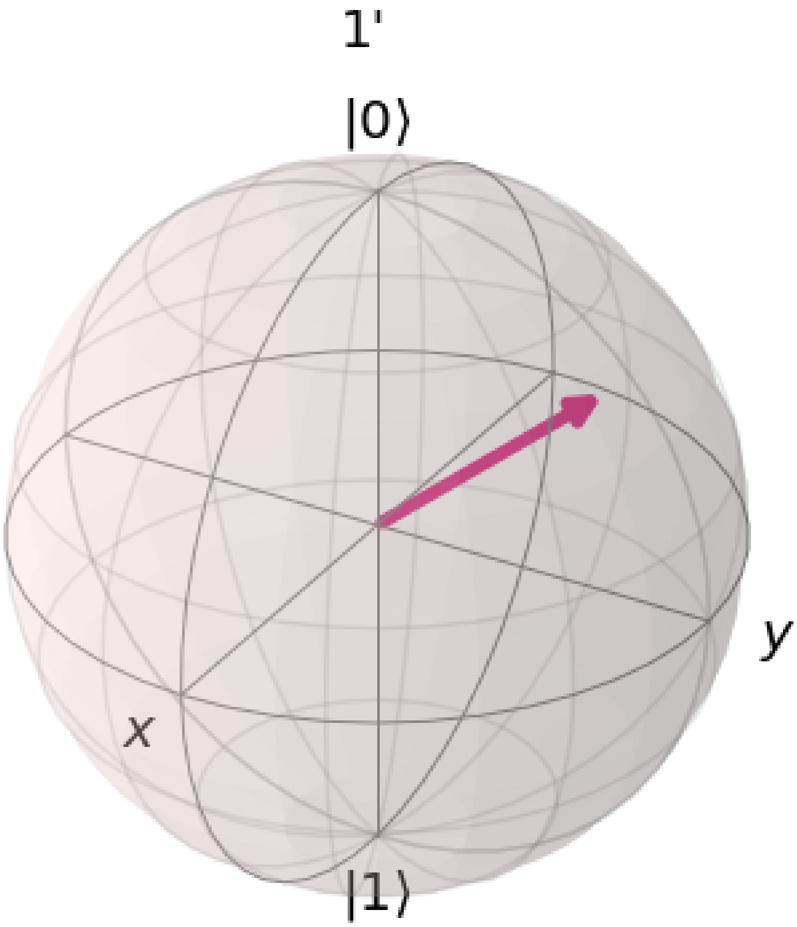
```
import numpy as np
import math
from qiskit import *
from qiskit.tools.visualization import plot_bloch_multivector
from qiskit.visualization import plot_bloch_vector
from qiskit.visualization import plot_histogram
from qiskit.extensions import Initialize


from qiskit import(
  QuantumCircuit,
  execute,
  Aer)

#def base1_measurement(qc,qubit,cbit):

    #qc.rz(math.pi*(16/11),0)
    #qc.ry(math.pi*(0.73),0)

    #qc.measure(qubit,cbit)

    #qc.ry(math.pi*(0.73),0)
    #qc.rz(math.pi*(16/11),0)

    #return qc



def base2_measurement(qc,qubit,cbit):

    qc.rz(math.pi*(16/11),0)
    qc.ry(math.pi*(0.73),0)

    qc.rz(math.pi*(16/11),1)
    qc.ry(math.pi*(0.73),1)
    qc.measure(qubit,cbit)
    qc.measure(qubit+1,cbit+1)

    qc.ry(math.pi*(0.73),0)
    qc.rz(math.pi*(16/11),0)

    qc.ry(math.pi*(0.73),1)
    qc.rz(math.pi*(16/11),1)
    return qc



def base3_measurement(qc,qubit,cbit):
    qc.rz(math.pi*(16/11),1)
    qc.ry(math.pi*(0.73),1)

    qc.measure(qubit+1,cbit+1)

    qc.ry(math.pi*(0.73),1)
    qc.rz(math.pi*(16/11),1)
    return qc

#simulator =Aer.get_backend('statevector_simulator')
#result =execute(qc,simulator).result()
#statevector =result.get_statevector()
#plot_bloch_multivector(statevector)

%matplotlib inline

bloch_vector = [math.sin(math.pi*0.73)*math.cos((16/11)*math.pi), math.sin(math.
 ↪pi*0.73)*math.sin((16/11)*math.pi), math.cos(math.pi*0.73)]
plot_bloch_vector(bloch_vector, title= "0'")
```

[1]:



[2]:
```
bloch_vector2 = [math.sin(-math.pi*0.27)*math.cos((16/11)*math.pi), math.
 ↪sin(-math.pi*0.27)*math.sin((16/11)*math.pi), math.cos(-math.pi*0.27)]
plot_bloch_vector(bloch_vector2, title= "1'")
```

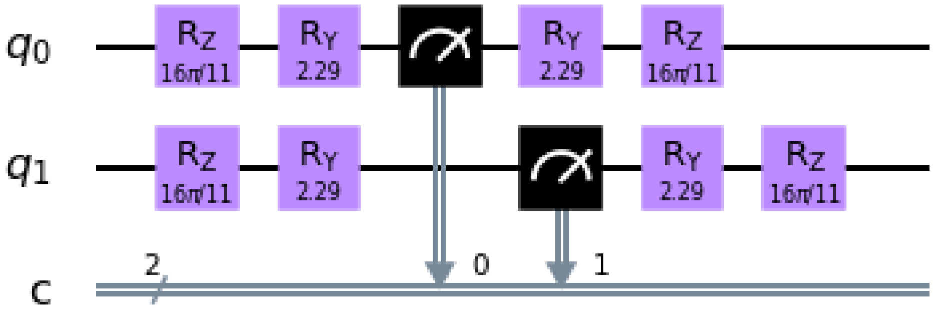
[2]:



[3]:

```
# 1.1
# circuito pra medir na base {0',1'}
qc = QuantumCircuit(2,2)


base2_measurement(qc, 0, 0)

qc.draw(output='mpl')
```

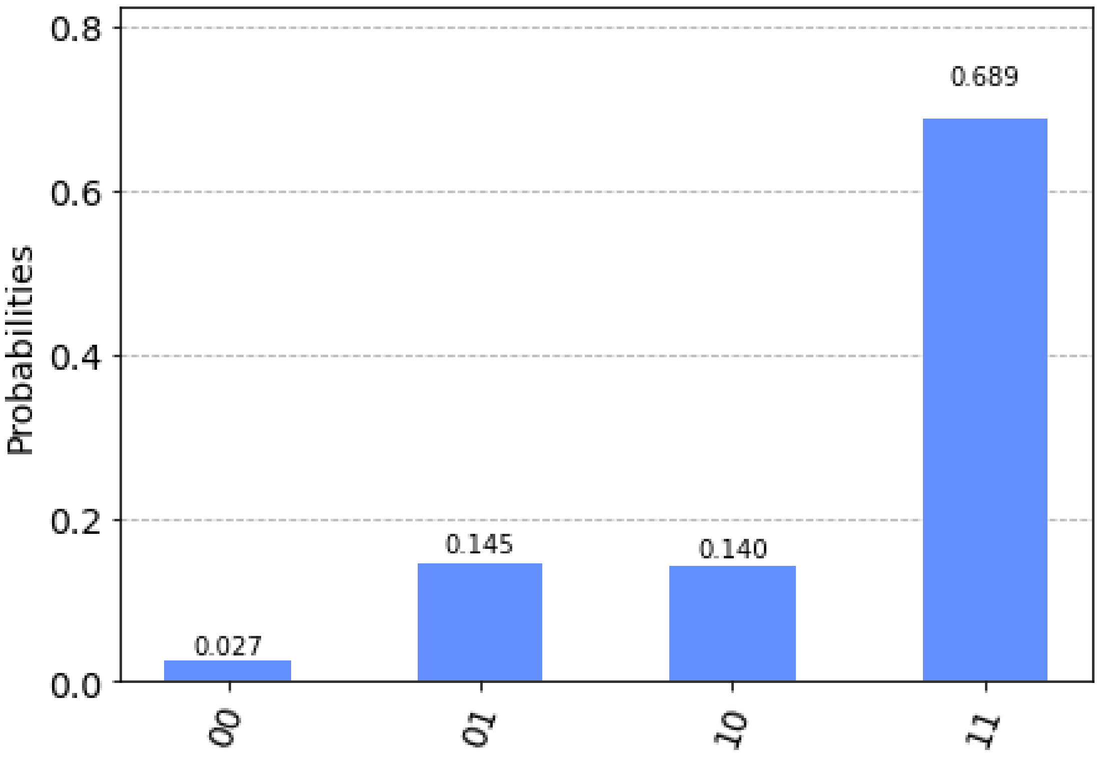
[3]:



[4]:

```
# 1.2
# medidas de 0  na base {0',1'}



backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

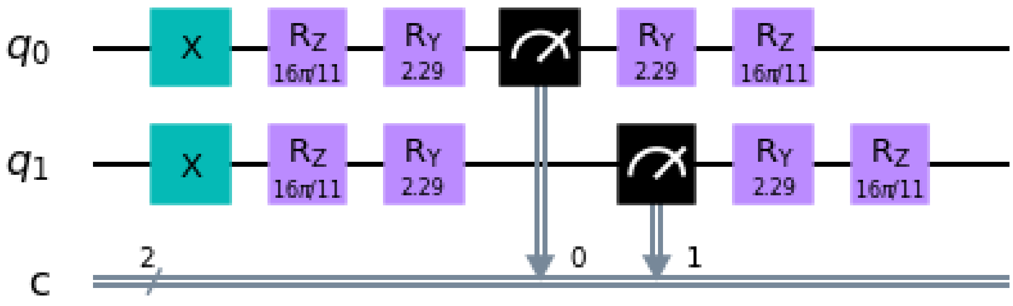
[4]:



[5]:

```
# 1.2
# medidas de 1 na base {0',1'}

qc = QuantumCircuit(2,2)


qc.x(0)
qc.x(1)
base2_measurement(qc, 0, 0)
#qc.x(0)

qc.draw(output='mpl')
```

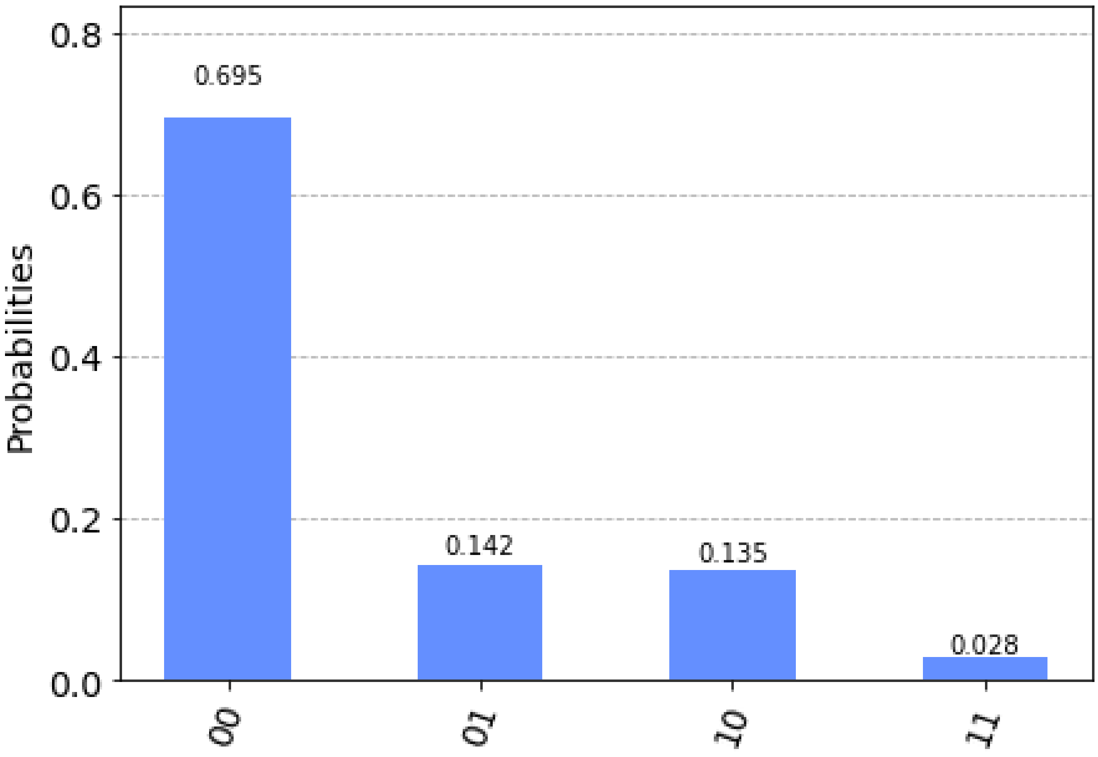
[5]:



[6]:

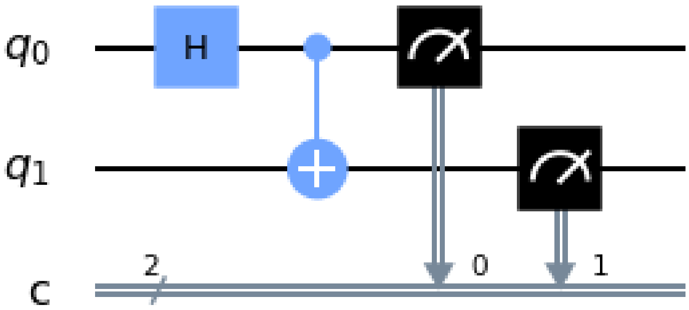
```
# 1.2
# medidas de 1 na base {0',1'}



backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

[6]:



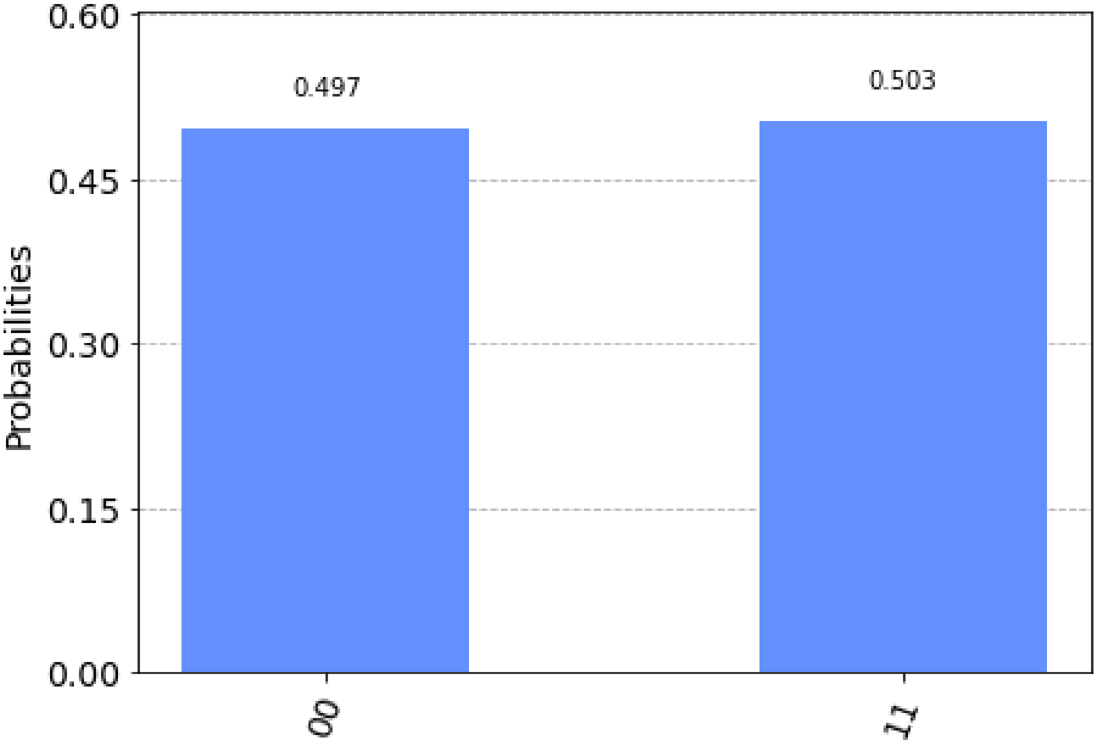
[7]:
```
# 2.1
#estado de bell phi+
qc2 = QuantumCircuit(2,2)

qc2.h(0)
qc2.cx(0,1)


qc2.measure(0,0)
qc2.measure(1,1)

qc2.draw(output='mpl')

#1/raiz2 0 0 1/raiz2
```

[7]:



[8]:
```
# 2.2 i
#medidas phi+ na base computacional
#qc2.measure([0,1], [0,1])

backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc2,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

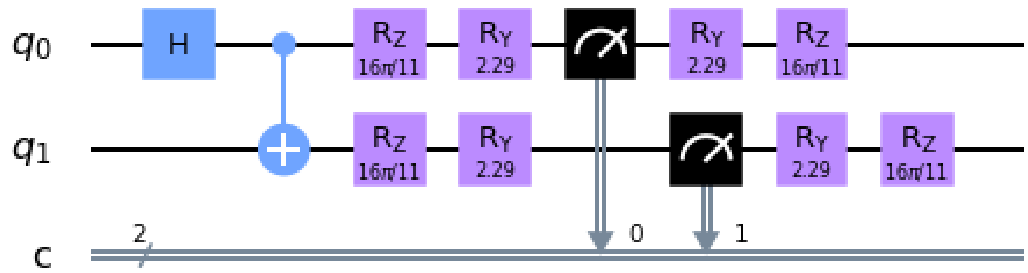
[8]:



[9]:

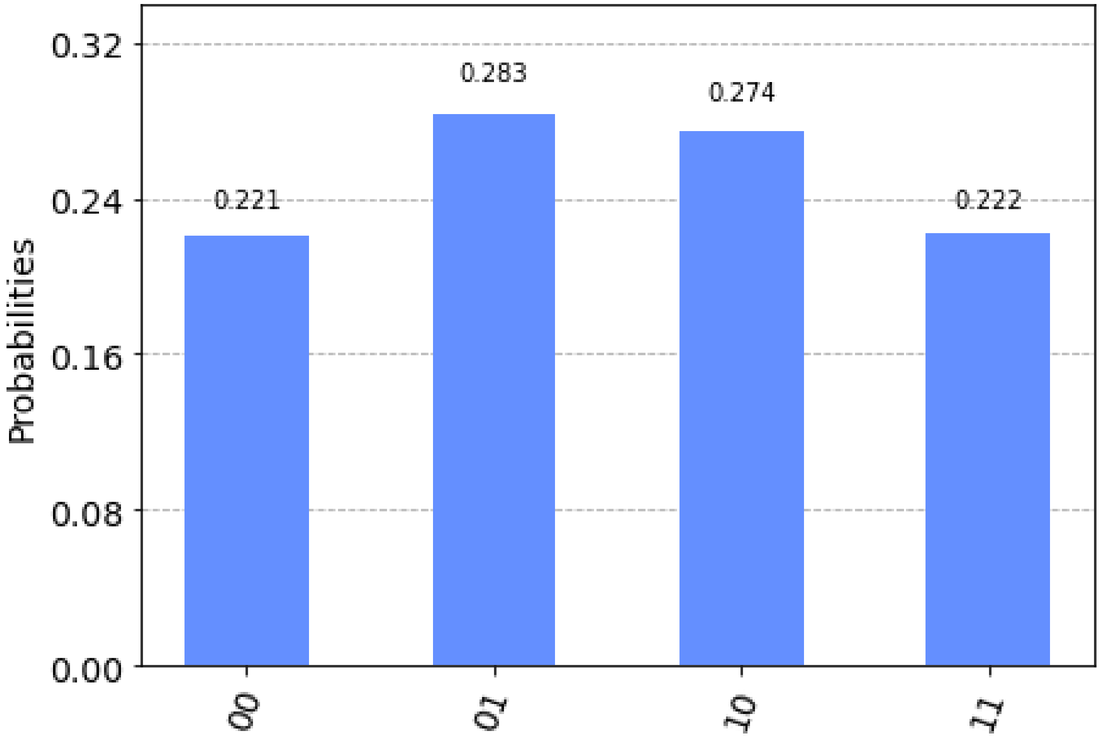
```
#estado de bell phi+ na base {0',1'}

qc3 = QuantumCircuit(2,2)

#base2_measurement(qc3, 0, 0)

qc3.h(0)
qc3.cx(0,1)

base2_measurement(qc3, 0, 0)

qc3.draw(output='mpl')
#1/raiz2 0 0 1/raiz2
```

[9]:



[10]:
```
# 2.2 ii
##medidas phi+ na base {0',1'}
#qc3.measure([0,1], [0,1])


backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc3,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

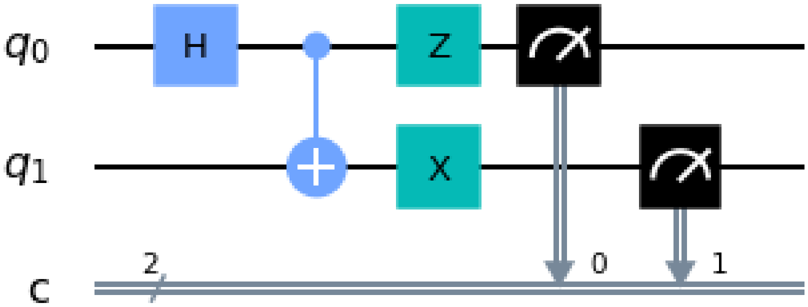
[10]:



[11]:

```
# 2.3
#estado de bell psi-
qc4 = QuantumCircuit(2,2)

qc4.h(0)
qc4.cx(0,1)
qc4.z(0)
qc4.x(1)

qc4.measure(0,0)
qc4.measure(1,1)

qc4.draw(output='mpl')
#0 1/raiz2 1/raiz2  0
```

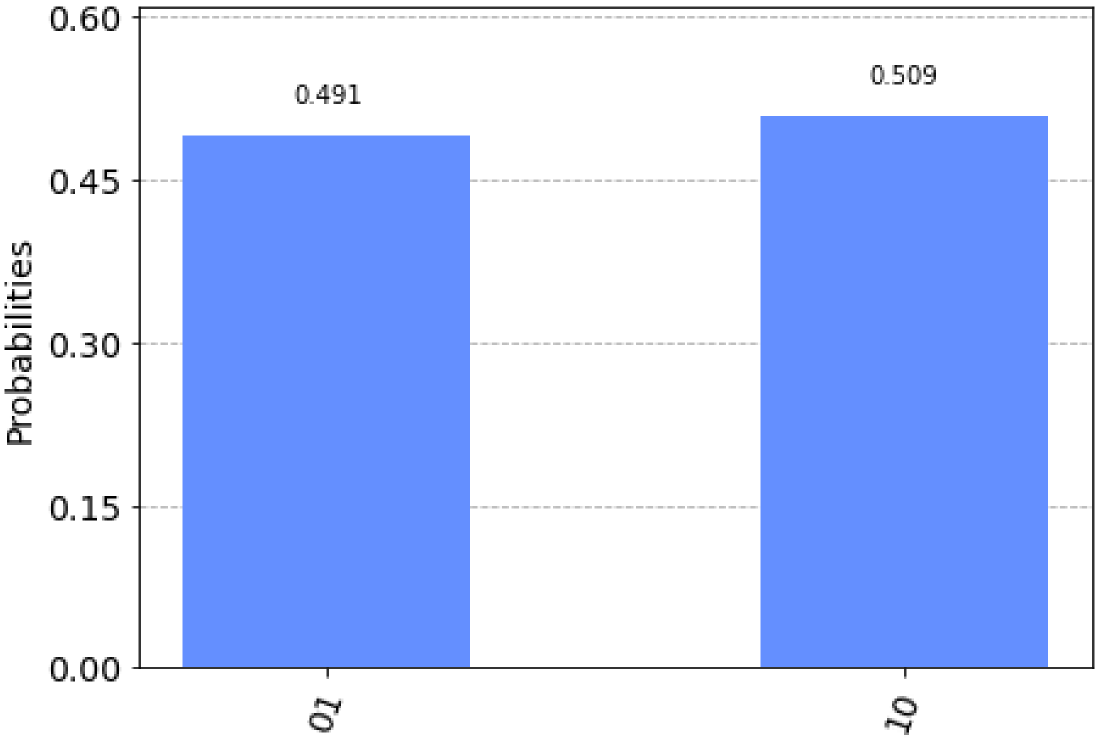
[11]:



[12]:

```
# 2.4 i
##medidas psi- na base computacional
#qc4.measure([0,1], [0,1])

backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc4,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

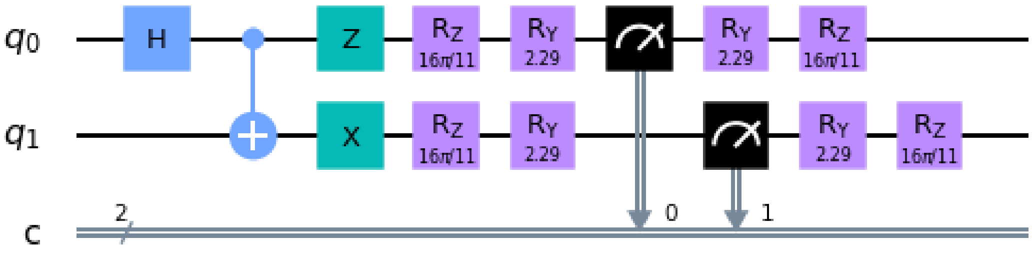
[12]:



[13]:

```
#psi-na base {0',1'}

qc5 = QuantumCircuit(2,2)

#base2_measurement(qc5, 0, 0)

qc5.h(0)
qc5.cx(0,1)
qc5.z(0)
qc5.x(1)

base2_measurement(qc5, 0, 0)

qc5.draw(output='mpl')
#0 1/raiz2 1/raiz2  0
```

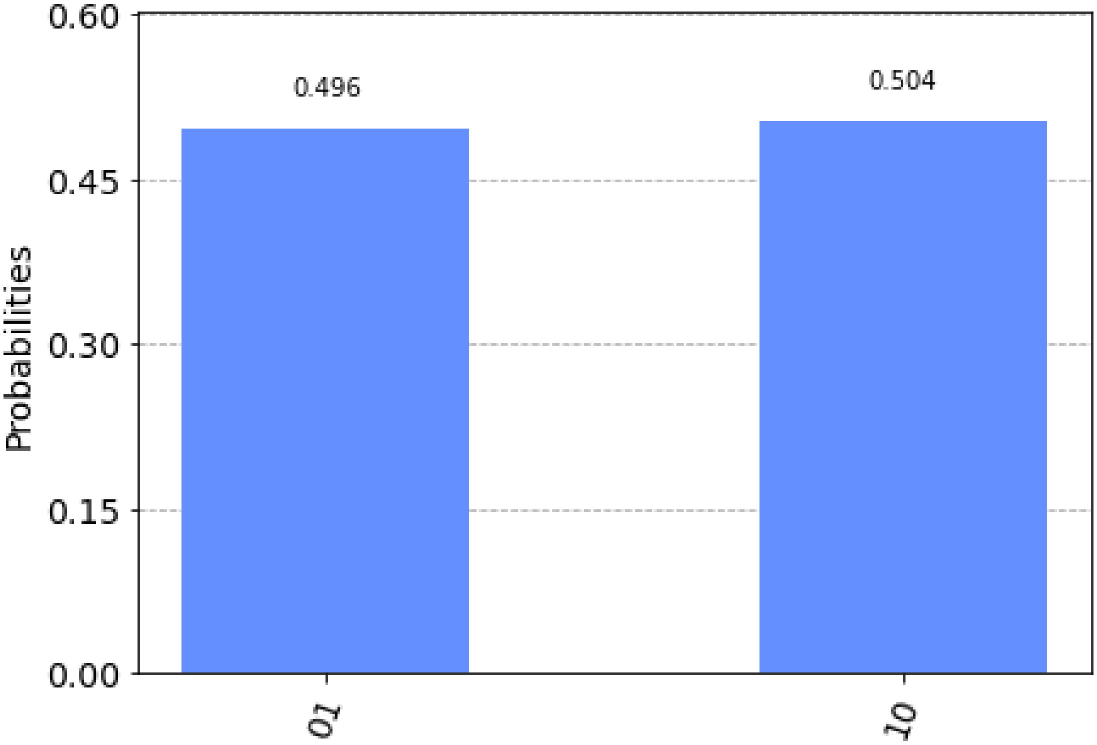
[13]:



[14]:

```
# 2.4 ii
#medidas psi- na base {0',1'}
#qc5.measure([0,1], [0,1])

backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc5,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

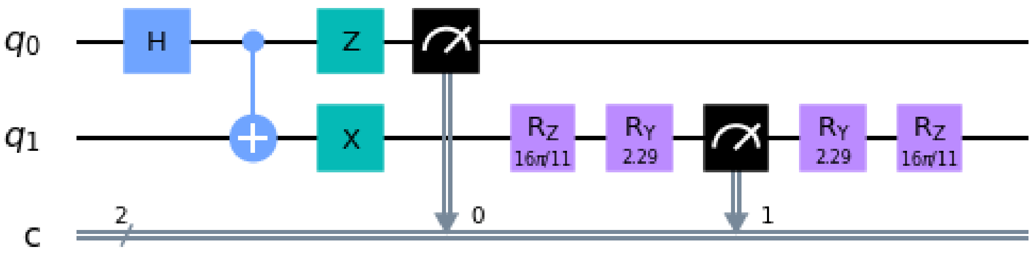
[14]:



[19]:

```
# 2.5
# psi-na base computacional e base {0',1'}
qc6 = QuantumCircuit(2,2)

#base3_measurement(qc6, 0, 0)

qc6.h(0)
qc6.cx(0,1)
qc6.z(0)
qc6.x(1)

qc6.measure(0,0)

base3_measurement(qc6, 0, 0)

#qc6.measure(0,0)

qc6.draw(output='mpl')
#0 1/raiz2 1/raiz2  0
```

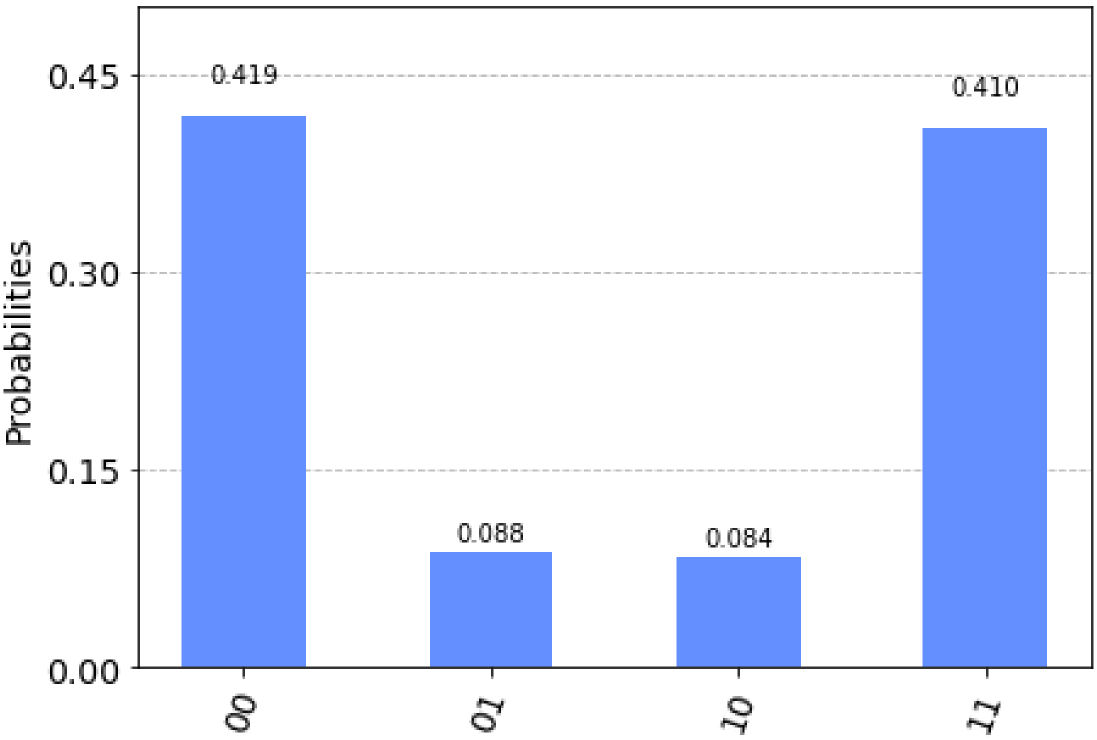
[19]:



[20]:

```
##medidas na base computacional e base {0',1'}
#qc6.measure([0,1], [0,1])

backend = Aer.get_backend('qasm_simulator')
result2 = execute(qc6,backend,shots=10000).result()
counts = result2.get_counts()

plot_histogram(counts)
```

[20]:


Probabilities
0.45
0.30
0.15
0.00
0.419
0.088
0.084
0.410
00
01
10
11

```
[ ]:
```