In [1]:
```
from qiskit import IBMQ
API_TOKEN ='bf24308d4b9628ea5d8f4a213416453f23eb280986828dbe5d3b691a9a9cd0b40b0b40a2e7741cc22abc447b4feb2ba88a4d51f4d2
512c7bca3aa964ef6b1a2c'
IBMQ.save_account(API_TOKEN)

Public_Provider = IBMQ.load_account()
Public_Provider.backends()

backend_publicism = Public_Provider.get_backend('ibmq_qasm_simulator')
backend_publiclon = Public_Provider.get_backend('ibmq_ourense')
print(backend_publicism)
print(backend_publiclon)

from qiskit import QuantumRegister, QuantumCircuit, ClassicalRegister
%matplotlib inline

from qiskit import BasicAer, execute
from qiskit.tools.monitor import job_monitor
from qiskit.tools.visualization import plot_histogram, plot_bloch_multivector
import numpy as np
from qiskit.extensions import Initialize
import math as m
```

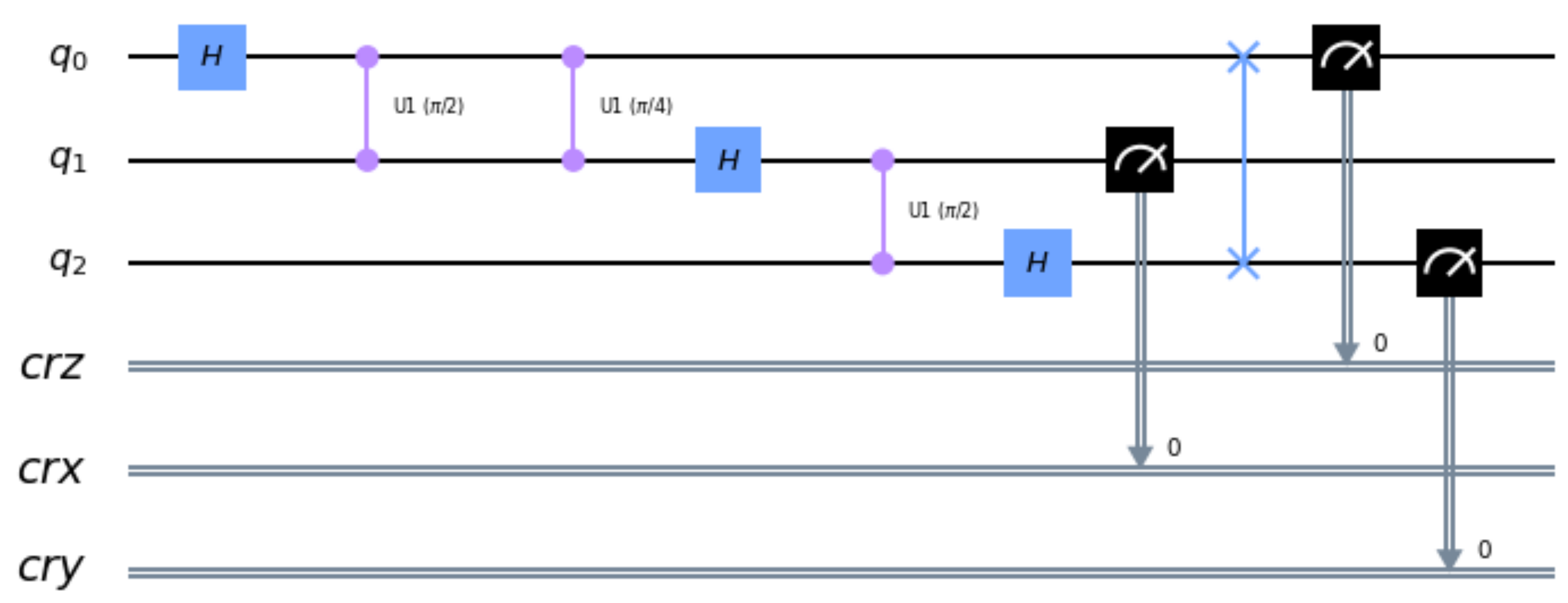
```
configrc.store_credentials:WARNING:2020-09-18 17:50:56,720: Credentials already present. Set overwrite=True to overwr
ite.
```

```
ibmq_qasm_simulator
ibmq_ourense
```

In [2]:
```
qr = QuantumRegister(3, name="q")
crz = ClassicalRegister(1, name="crz")
crx = ClassicalRegister(1, name="crx")
cry = ClassicalRegister(1, name="cry")
FT_circuit = QuantumCircuit(qr, crz, crx,cry)
```

In [3]:
```
FT_circuit.h(qr[0])
FT_circuit.cu1(m.pi/2,qr[1], qr[0])
FT_circuit.cu1(m.pi/4,qr[0],qr[1])
FT_circuit.h(qr[1])
FT_circuit.cu1(m.pi/2,qr[2], qr[1])
FT_circuit.h(qr[2])
FT_circuit.swap(qr[0],qr[2])
FT_circuit.measure(qr[0],crz)
FT_circuit.measure(qr[1],crx)
FT_circuit.measure(qr[2],cry)
FT_circuit.draw(output = 'mpl')
```

Out[3]:



In [4]:
```
job = execute(teleportation_circuit,backend_publiclon)
job_monitor(job)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-5fe2a9e8237a> in <module>
----> 1 job = execute(teleportation_circuit,backend_publiclon)
      2 job_monitor(job)

NameError: name 'teleportation_circuit' is not defined
```

In [5]:
```
job = execute(FT_circuit,backend_publiclon)
job_monitor(job)
```

```
Job Status: job has successfully run
```

In [6]:
```
result = job.result()
```

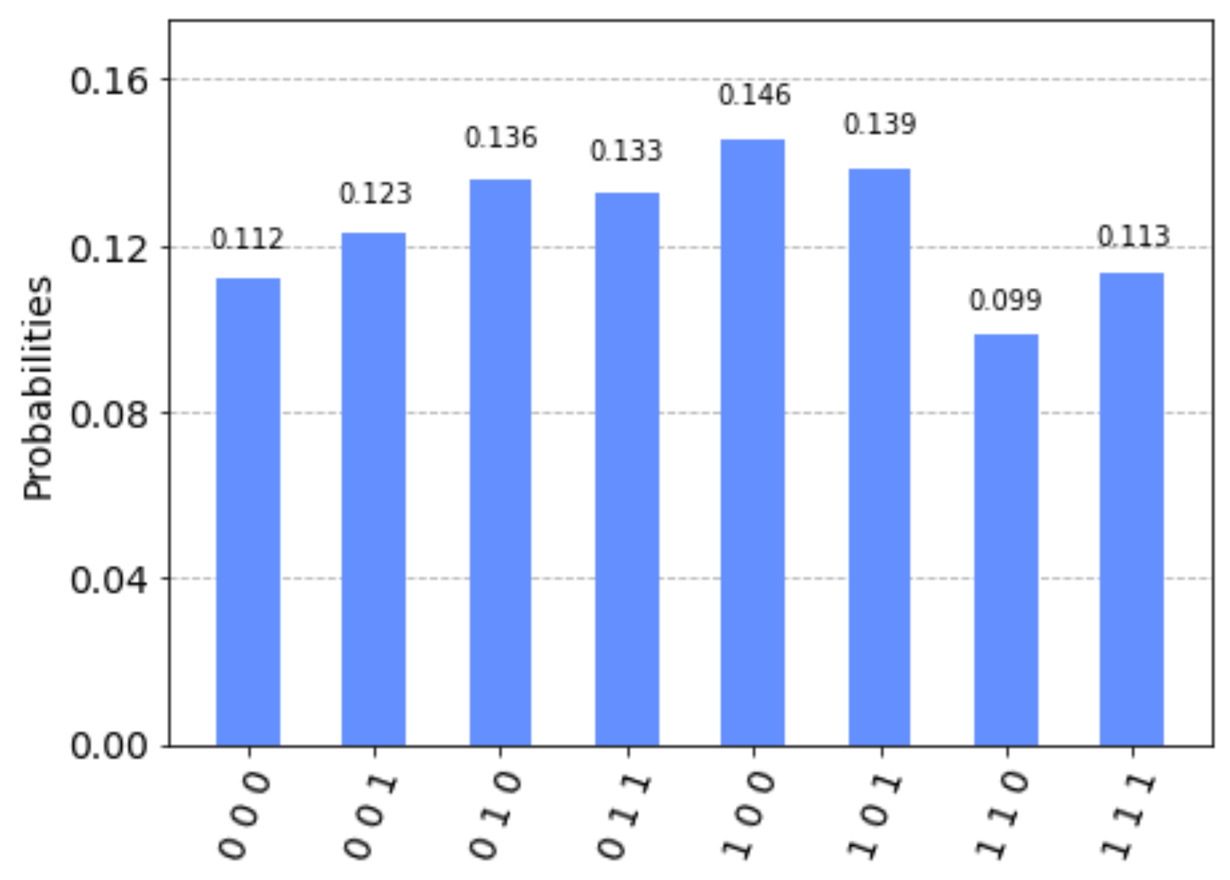
In [7]:
```
count = result.get_counts()
```

In [8]:
```
print(count)
```

```
{'1 1 0': 101, '1 1 1': 116, '0 0 1': 126, '1 0 0': 149, '0 1 1': 136, '0 0 0': 115, '0 1 0': 139, '1 0 1': 142}
```

In [9]:
```
plot_histogram(count)
```

Out[9]:



In [ ]: