PLATAFORMA *IBM Q - QISKIT*: UM RECURSO PEDAGÓGICO PARA A DISCIPLINA DE COMPUTAÇÃO QUÂNTICA NAS ENGENHARIAS

Victória B. S. Barros – victoriabsb@hotmail.com
Universidade Federal do Pará, Faculdade de Engenharia de Computação
Rua Augusto Corrêa, 01 - Guamá
66075-110 – Belém – PA

Wilson R. M. Rabelo – rabelo@ufpa.br
Universidade Federal do Pará, Faculdade de Engenharia de Computação
Rua Augusto Corrêa, 01 - Guamá
66075-110 – Belém – PA

Resumo: *Este trabalho trata da utilização de um processador quântico com 5-qbits, conhecido na literatura como IBM Q e acessível gratuitamente através da internet. Um framework* de desenvolvimento fornecido gratuitamente pela empresa *IBM e chamado de QisKit,* fornece a *ligação entre o usuário e o processador quântico onde se deseja rodar os circuitos ou algoritmos, voltados para o ensino e pesquisa da computação quântica. O objetivo de nosso trabalho consistiu em uma caracterização do QisKit referente a vários aspectos práticos do framework. Consideramos quais seriam as principais condições para o acesso a esse processador quântico via QisKit e, quanto a sua utilização, avaliamos com relação a simplicidade e execução de circuitos básicos da computação quântica. Nosso resultado demonstrou que o QisKit pode ser útil como um recurso didático importante para as disciplinas de computação quântica nas engenharias e áreas afins. Destacamos também que o framework possui mecanismos básicos para o processamento da informação, isto é, criação de circuitos quânticos, com portas quânticas e medidas nas bases computacionais.*

Palavras-chave: *QisKit. IBM Q. Processador Quântico. Computação Quântica.*

1 INTRODUÇÃO

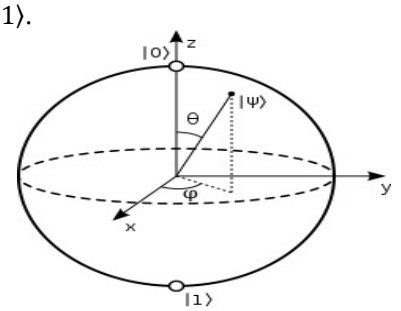
Nas duas últimas décadas, o interesse pela computação quântica vinha da motivação de que alguns resultados teóricos indicavam que certos algoritmos quânticos fornecem ganhos computacionais em relação aos análogos clássicos. Como resultado, atualmente experimentamos avanços tecnológicos que nos permitem começar a contornar alguns problemas práticos para a fabricação escalável dos processadores quânticos. Como exemplos, podemos citar a chegada do primeiro computador quântico da empresa de tecnologia D-Wave (*D-Wave One*), em 2011, e o *IBM Q*, em 2016, da empresa de tecnologia IBM. Abordando mais especificamente o computador da IBM, o *IBM Q* é um processador quântico de 5 q-bits (existe a versão do processador com 14 q-bits) que pode ser acessado remotamente via internet gratuitamente. No campo pedagógico propostas didáticas usando o *IBM Q* vem sendo analisadas e discutidas. Neste contexto, o nosso estudo consistiu em investigar as potencialidades do *IBM Q* usando o framework QisKit como um recurso pedagógico para trabalhar conceitos básicos da computação quântica. O QisKit é um framework de desenvolvimento escrito na linguagem Python e voltado para a computação quântica. Sua utilidade básica é a conexão entre o usuário e o dispositivo desejado (servidores convencionais para a simulação e/ou processador quântico). O QisKit também é subdividido em quatro

projetos: Terra, Aer, Aqua e Ignis, onde cada um desses projetos são destinados a uma dada finalidade. Neste trabalho discutiremos apenas o QisKit Terra e Aer. O QisKit Terra é o framework básico, que permeia todos os outros elementos e fundamental para o processo de criação de portas, circuitos e medidas quânticas em q-bits e, ainda permite a execução dos circuitos quânticos, seja localmente ou nos processadores quânticos. O QisKit Aer é o framework que também interage com todos os outros elementos, entretanto, voltada para a simulação de circuitos quânticos com ruídos intermediários. Este trabalho está dividido da seguinte forma: na seção 2 fornecemos um suporte teórico e, na seção 3, abordamos a instalação do QisKit. Na seção 4, discutimos a utilização do framework e, por fim na seção 5, as conclusões deste trabalho.

2 SUPORTE TEÓRICO: Q-BITS, PORTAS, CIRCUITOS E MEDIDAS QUÂNTICAS

Um conceito importante na computação quântica é o q-bit, um bit de informação codificado nos estados quânticos de um sistema de 2-níveis ($|0\rangle, |1\rangle$) que, matematicamente, é descrito por um vetor bidimensional no espaço Hilbert complexo de dimensão finita, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \binom{\alpha}{\beta}$, onde α e β são as amplitudes de probabilidades dos respectivos estados da base, com $|\alpha|^2 + |\beta|^2 = 1$. Também podemos mapear essas amplitudes de probabilidades α e β em função dos ângulos θ e ϕ, como $|\psi\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle$. Com essa descrição do estado do q-bit podemos agora obter uma visualização de todos os estados possíveis (em função de θ e ϕ) sobre a superfície de uma esfera com raio unitário. Essa descrição geométrica do q-bit é chamada de representação de esfera de Bloch, veja a Figura 1 para mais detalhes.

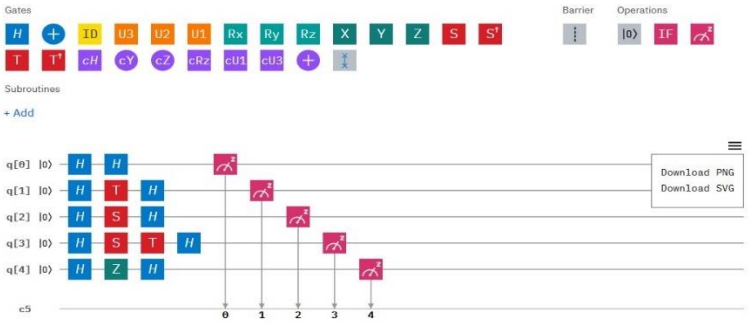
Figura 1: Esfera de Bloch: uma representação geométrica dos estados de 1-qbit. Os polos são os auto estados computacionais $|0\rangle, |1\rangle$.



Fonte: NIELSEN, Michael; CHUANG, Isaac, 2010.

No modelo de circuito padrão da computação quântica um conjunto de *n* q-bits, com uma evolução completamente controlada, é dado por uma sequência de operadores unitários atuando em um, dois ou mais q-bits. Esses operadores unitários são chamados de portas lógicas quânticas, em analogia aos computadores digitais, veja a Figura 2.

Figura 2 - Representação computacional de um circuito na computação quântica. Nessa figura também temos a plataforma de acesso do computador quântico, chamado de *IBM Q*.



Fonte: IBM Quantum Experience

Em termos de simbologia, a evolução temporal dos q-bits é representada pelas linhas horizontais da figura acima e os estágios desta evolução, contendo uma série de portas lógicas atuando nos q-bits. Na figura 2, também temos um diagrama de modelo computacional padrão com 5-qbits, todos no estado inicial $|0\rangle$ e no início de cada linha horizontal. A acima das linhas, as portas quânticas disponíveis para elaboração do circuito. Para compor o exemplo da Figura 2, utilizamos as portas ***H, T e S*** e medidas ao final. A última linha horizontal (*c5*) da Figura 2 é reservada aos resultados das medidas realizadas sobre os q-bits. Para mais detalhes sobre a abordagem matemática na descrição das portas e circuitos quânticos, consulte a referência (NIELSEN, Michael; CHUANG, Isaac, 2010).

3 INSTALAÇÃO DO FRAMEWORK - QISKIT

3.1 Requisitos e Instalação

Neste tópico vamos demonstrar como instalar o framework QisKit (Terra, Aer, Aqua e Ignis). A empresa de tecnologia IBM recomenda a instalação de uma plataforma de distribuição Python, chamada Anaconda, voltada para a computação científica. O Jupyter Notebook, incluído no Anaconda, é recomendado para o desenvolvimento e a utilização dos tutoriais QisKit. Segundo a IBM, o QisKit já foi testado e roda nos seguintes sistemas operacionais de 64-bits: Ubuntu Linux 16.04 ou superior, MacOS 10.12.6 ou posterior e Windows 7 ou posterior. Uma observação a mais é feita para usuários do sistema Windows 7 ou superior, onde a empresa IBM recomenda o *Microsoft Visual C++ Redistributable* para o *Visual Studio 2017* e o *2015*. Para a instalação a IBM recomenda a criação de um ambiente virtual para o QisKit, separadamente de outras aplicações. Portanto, usuário deve rodar alguns comandos em um terminal que, para sistemas Windows é o "Prompt de Comando" e para sistemas Linux e MacOs é simplesmente o "Terminal". Neste terminal o usuário irá especificar alguns comandos do Anaconda, já instalado, para a criação desse ambiente virtual e fixar as bibliotecas necessárias para o framework QisKit. Os comandos a seguir serão em modo terminal em diretório escolhido pelo usuário: *C:\> conda create -n Ambiente_Teste python=3; C:\> source activate Ambiente_Teste; C:\> activate Ambiente_Teste* . Agora com o ambiente do usuário foi criado (neste caso escolhemos o rótulo *Ambiente_Teste*, entretanto, o usuário pode escolher qualquer nome), instala-se o framework QisKit neste ambiente através do comando *pip*:

C:\> pip install qiskit

Se o framework foi instalado corretamente, rode o comando *conda list* no terminal para fazer um checklist dos pacotes instalados no seu ambiente virtual.

3.2 Acessando os Sistemas IBM Q´s

A empresa IBM disponibiliza gratuitamente via internet vários processadores quânticos reais e computadores de alta performance para simulação com o QisKit. Abaixo seguem os passos para o acesso desses processadores clássicos e quânticos:

1) Criar uma conta no *IBM Q Experience*. Segue o link: https://quantum-computing.ibm.com/login
2) Click em "*My Account*" para acessar as configurações da conta.

3) Click em "*Copy token*" para copiar a chave de acesso. Use um editor de texto, de sua escolha, para colar temporariamente. Posteriormente, usaremos no passo 5.

4) Acesse a sua conta no link "*Your accounts*". Posteriormente, click em "*copy url*" e cole temporiamente este no editor de sua escolha.

5) Armazene seus *Copy token* e *copy url* na configuração do arquivo de nome "*qiskitrc*", através dos comandos:

C:\> from qiskit import IBMQ
C:\> IBMQ.save_account('MY_API_TOKEN', 'MY_URL')
Observe que *'MY_API_TOKEN' e 'MY_URL' deverão ser substituídos pelos seus valores copiados no passo 3 e 4.*

Também é importante acrescentar que as simulações e acesso remoto ao *IBM Q* via internet foram realizadas em um computador desktop com as seguintes configurações: processador Intel Core i5, com 8 GB de memória RAM, sistema operacional Windows 7 Professional de 64 bits e HD de 500 GB.

4 UTILIZANDO O QISKIT TERRA/AER

O QisKit Terra/Aer será usado para o desenvolvimento de circuitos e algoritmos quânticos voltados para o ensino da computação quântica. Por simplicidade, a partir de agora denotaremos o QisKit Terra/Aer apenas por QisKit. Para definirmos portas, circuitos e medidas quânticas usando o QisKit, precisaremos de uma interface de trabalho (*IDE- Integrated Development Environment*) que pode ser: o JupyterLab, o Jupyter Notebook ou o Spyder. Todos esses aplicativos estão incluídos na plataforma Anaconda. A IBM recomenda o uso do Jupyter Notebook para desenvolvimento e a utilização de seus tutoriais.

Primeiramente, importaremos algumas bibliotecas da linguagem de programação Python, para fazer gráficos, operações matemáticas e constantes. Figura 3 abaixo demonstra o código em Jupyter Notebook.

Figura 3 – Demonstração do código para importar algumas bibliotecas da linguagem python.

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline
        import numpy as np
        from math import pi
```

Fonte: Jupyter Notebook, plataforma usada para programar utilizando o QisKit.

Logo em seguida, importaremos alguns módulos da biblioteca Qikist, tais como*: QuantumRegister, QuantumCircuit, ClassicalRegister* e o *execute*, que respectivamente, são responsáveis pela declaração de número de q-bits usados, declaração do circuito quântico, declaração dos bits clássicos necessários para armazenar os resultados das leituras e o módulo de execução do circuito. Veja a Figura 4.

Figura 4 – Demonstração do código para importar bibliotecas do Qiskit.

```
In [2]: from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute
```

Fonte: Jupyter Notebook – IBM Q.

Também é importante ressaltarmos que o comando *circuit_drawer* foi retirado do QisKit. No lugar foi incluído o módulo *draw*. Este comando é muito útil, pois demonstra em imagem

formato *text*, *latex* ou *matplotlib*, do circuito quântico após a declaração de todas as portas lógicas. Outro passo importante é importar do Qiskit o *BasicAer* (Figura 5).

Figura 5 – Demonstração do código para importar o simulador de circuitos quânticos.

```
In [3]: from qiskit import BasicAer

        backend = BasicAer.get_backend('statevector_simulator')
```

Fonte: Jupyter Notebook – IBM Q.

Esse comando é importante para realizar três tipos de simulações localmente: (1) *unitary_simulator*, que compreende que os elementos do circuito funcionarão como um operador unitário e a sua saída será dada na forma de uma matriz; (2) *qasm_simulator*, diferente do anterior, este já garante que sejam efetuadas medidas sobre os q-bits, (3) *state_vector* retorna o estado quântico na forma de um vetor complexo de dimensão 2^n, onde n é o número de q-bits. Vale ressaltar a importância de verificar que a quantidade de q-bits afetará diretamente a performance da máquina clássica (desktops ou servidores) que simulará o circuito, podendo ou não rodar o experimento devido ao aumento exponencial da complexidade computacional.

4.1 Portas de 1-qbit

Atuando em um único q-bit, as portas quânticas são matrizes 2 x 2 unitárias, muito utilizadas na computação quântica. Na tabela abaixo, colocamos alguns exemplos.

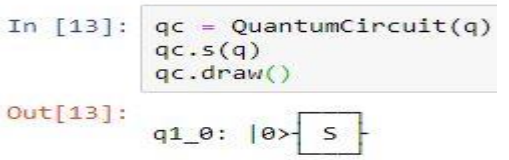
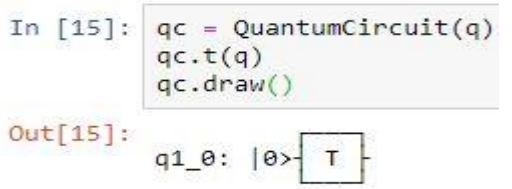
Portas de 1-qbit	**Descrição Matricial**	**Declaração no Jupyter Notebook Utilizando o QisKit**	**ATUAÇÃO NA BASE COMPUTACIONAL**

X
Porta Not
	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$	`In [9]: qc = QuantumCircuit(q)`
		`qc.x(q)`
		`qc.draw()`
		`Out[9]: q1_0: |0>─[ X ]─`
			$\|0\rangle \rightarrow \|1\rangle$
			$\|1\rangle \rightarrow \|0\rangle$

Y
Porta Y
	$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$	`In [10]: qc = QuantumCircuit(q)`
		`qc.y(q)`
		`qc.draw()`
		`Out[10]: q1_0: |0>─[ Y ]─`
			$\|0\rangle \rightarrow i\|1\rangle$
			$\|1\rangle \rightarrow -i\|0\rangle$

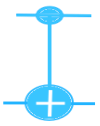
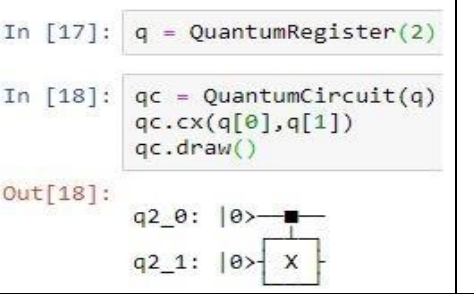
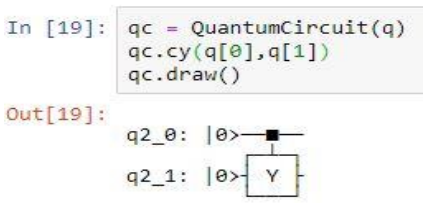
Z
Porta Z
	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$	`In [11]: qc = QuantumCircuit(q)`
		`qc.z(q)`
		`qc.draw()`
		`Out[11]: q1_0: |0>─[ Z ]─`
			$\|0\rangle \rightarrow \|0\rangle$
			$\|1\rangle \rightarrow -\|1\rangle$

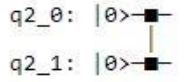
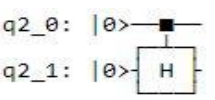
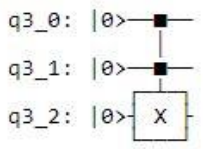
H
Porta Hadamard
	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$	`In [12]: qc = QuantumCircuit(q)`
		`qc.h(q)`
		`qc.draw()`
		`Out[12]: q1_0: |0>─[ H ]─`
			$\|0\rangle \rightarrow \frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$
			$\|1\rangle \rightarrow \frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$

 Porta de Fase	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$		$\|0\rangle \rightarrow \|0\rangle$ $\|1\rangle \rightarrow i\|1\rangle$
 Porta T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$		$\|0\rangle \rightarrow \|0\rangle$ $\|1\rangle \rightarrow e^{i\pi/4}\|1\rangle$

4.2 Portas de múltiplos q-bits

Atuando em dois ou mais q-bits, as portas quânticas são matrizes unitárias, muito utilizadas em circuitos e algoritmos mais complexos. Todas as portas a seguir são habilitadas quando primeiro q-bit, conhecido como controle, está no estado $|1\rangle$. Logo, a porta atua no segundo q-bit, conhecido como alvo. Na tabela abaixo, colocamos alguns exemplos.

Portas de múltiplos q-bits	Descrição Matricial	Declaração no Jupyter Notebook Utilizando o QisKit	Atuação na base computacional
 **Porta CNOT** -Not-Controlado	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$		$\|00\rangle \rightarrow \|00\rangle$ $\|01\rangle \rightarrow \|01\rangle$ $\|10\rangle \rightarrow \|11\rangle$ $\|11\rangle \rightarrow \|10\rangle$
Cy Porta Y-Controlada	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}$		$\|00\rangle \rightarrow \|00\rangle$ $\|01\rangle \rightarrow \|01\rangle$ $\|10\rangle \rightarrow i\|11\rangle$ $\|11\rangle \rightarrow -i\|10\rangle$

Cz Porta Z-Controlada	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	In [20]: `qc = QuantumCircuit(q)` `qc.cz(q[0],q[1])` `qc.draw()` Out[20]: 	$\|00\rangle \rightarrow \|00\rangle$ $\|01\rangle \rightarrow \|01\rangle$ $\|10\rangle \rightarrow \|10\rangle$ $\|11\rangle \rightarrow -\|11\rangle$
C_H Porta H-Controlada	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$	In [21]: `qc = QuantumCircuit(q)` `qc.ch(q[0],q[1])` `qc.draw()` Out[21]: 	$\|00\rangle \rightarrow \|00\rangle$ $\|01\rangle \rightarrow \|01\rangle$ $\|10\rangle \rightarrow \frac{1}{\sqrt{2}}\|1\rangle\otimes\{\|0\rangle + \|1\rangle\}$ $\|11\rangle \rightarrow \frac{1}{\sqrt{2}}\|1\rangle\otimes\{\|0\rangle - \|1\rangle\}$
Porta Toffoli (a porta atua em 3 q-bits. Será habilitada quando os 2 controles forem $\|1\rangle$)	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	In [23]: `q = QuantumRegister(3)` In [24]: `qc = QuantumCircuit(q)` `qc.ccx(q[0],q[1],q[2])` `qc.draw()` Out[24]: 	$\|000\rangle \rightarrow \|000\rangle$ $\|001\rangle \rightarrow \|001\rangle$ $\|010\rangle \rightarrow \|010\rangle$ $\|011\rangle \rightarrow \|011\rangle$ $\|100\rangle \rightarrow \|100\rangle$ $\|101\rangle \rightarrow \|101\rangle$ $\|110\rangle \rightarrow \|111\rangle$ $\|111\rangle \rightarrow \|110\rangle$

4.3 Um circuito quântico na versão QisKit

Uma das operações mais importantes para o estudo da computação quântica é a leitura dos q-bits, ou seja, após o processamento realizamos as medidas. Basicamente, após a realização da medida, o estado do q-bit irá colapsar entre as várias possibilidades de estar ou não em uma série de estados quânticos possíveis, com uma dada amplitude de probabilidade. No QisKit devemos alterar o simulador para o *qasm_simulator*, que possui suporte para medidas quânticas.

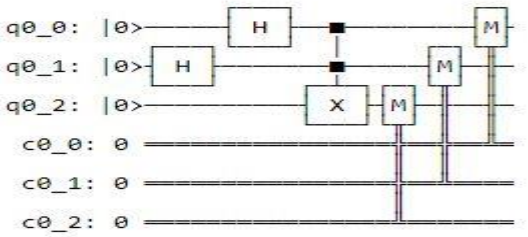
Agora tomaremos um "circuito-exemplo" para fins pedagógicos e, aplicando quando necessário, os códigos QisKit utilizados acima. O circuito-exemplo consiste em apenas três portas quânticas, duas de 1-qbit que são as portas de Hadamard e, posteriormente, utilizamos a porta Toffoli de 3-qbits. E por fim, realizamos as medidas dos q-bits. Observe que os três q-bits estão inicializando no estado padrão $|0\rangle$, e como vamos realizar também três medidas ao final, precisamos de três registros clássicos, c0_0, c0_1, c0_2. Para implementação do circuito, temos:

Figura 6 – Implementação de um circuito com medida e seus respectivos resultados

```
In [26]: q = QuantumRegister(3)
         c = ClassicalRegister(3)

In [27]: qc = QuantumCircuit(q,c)
         qc.h(q[0])
         qc.h(q[1])
         qc.ccx(q[0],q[1],q[2])
         qc.measure(q,c)
         qc.draw()

Out[27]:
```




```
In [28]: job = execute(qc, backend, shots = 1024)
         job.result().get_counts(qc)

Out[28]: {'111': 231, '010': 266, '000': 288, '001': 239}
```

Fonte: Jupyter Notebook – IBM Q.

Acima temos o resultado das medidas, que representa a probabilidade de 1024 rodadas do experimento, 231 deles resultaram em $|111\rangle$, 239 deles em $|001\rangle$, 288 deles em $|000\rangle$ e 266 em $|010\rangle$.

5 CONSIDERAÇÕES FINAIS

Este trabalho introduziu conhecimentos básicos referente a computação quântica, utilizando um framework da IBM, como um recurso didático atual para a disciplina de computação quântica. Neste artigo, abordou-se conceitos básicos de portas e circuitos quânticos e, como esses conceitos poderiam ser explorados utilizando o framework QisKit para projetos de circuitos em simuladores ou processadores quânticos.

É importante ressaltar que as codificações básicas do QisKit foram fornecidas neste trabalho e, portanto, os alunos de engenharia que fazem a disciplina poderão explorar não somente o circuito-exemplo deste artigo, como também criar seus próprios circuitos quânticos.

Por fim, a computação quântica ainda é uma área de estudo em desenvolvimento, que apenas na última década vem ganhando certo destaque dado seus avanços em campos de estudo que agrega a outros, tais como: a segurança da informação (criptografia) e o aprendizado de máquina (machine learning) como também no campo de aplicações escaláveis realísticas. Portanto, encontrar recursos didáticos que, permitam o melhor entendimento de conceitos da disciplina de computação e informação quântica, são fundamentais para o aprendizado dos alunos.

REFERÊNCIAS

Livros:
NIELSEN, Michael A.; CHUANG, Isaac L. **Quantum computation and quantum information**. 10ª ed. São Paulo: Cambridge University Press. 2010.

Internet:

IBM Q Experience Beginners Guides. Disponível em: https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=introduction. Acesso em: 29 abr. 2019.

Qiskit Terra Documentation. Disponível em: https://qiskit.org/documentation/. Acesso em: 29 abr. 2019.
Qiskit Aqua Documentation. Disponível em: https://qiskit.org/documentation/aqua/. Acesso em 29 abr. 2019.

Qiskit Tutorials. Disponível em: https://nbviewer.jupyter.org/github/Qiskit/qiskit-tutorial/blob/master/index.ipynb. Acesso em 29 abr. 2019.

Quantum Logic Gate. Disponível em: https://en.wikipedia.org/wiki/Quantum_logic_gate#Pauli-X_gate. Acesso em 29 abr. 2019.

IBM Q - QISKIT PLATFORM: A PEDAGOGICAL RESOURCE FOR THE QUANTUM COMPUTING DISCIPLINE IN ENGINEERING

Abstract: *This work deals with the use of a quantum processor with 5-qbits, known in the literature as IBM Q and accessible freely through the internet. A development framework provided free by IBM and called QisKit, provides the link between the user and the quantum processor where you want to run the circuits or algorithms, aimed at the teaching and research of quantum computing. The objective of this work was to characterize QisKit regarding several practical aspects of the framework. We considered what would be the main conditions for access to this quantum processor via QisKit and, in terms of its use, we evaluated with respect to the simplicity and execution of basic circuits of quantum computation. Our results demonstrated that QisKit can be useful as an important didactic resource for the quantum computing disciplines in engineering and related fields. We also emphasize that the framework has basic mechanisms for the information processing, that is, creation of quantum circuits, with quantum ports and measures in the computational bases.*

Keywords: *QisKit. IBM Q. Quantum Processor. Quantum Computation.*