Quantum Robot

Learning about Quantum Circuits Using Qiskit
and about a basic Quantum Robot

Prepared By: Aritra Mukhopadhyay
under the supervision of Prof. Prasanta K Panigrahi

August 14, 2022

Abstract

Qiskit is an open-source software development kit widely used for learning quantum computation and working on related topics. It is used to run quantum algorithms both on simulation of quantum computers and on real quantum computers. It translates common programming languages like Python into quantum machine language.

Just like classical computers, quantum computers are also based on gates. They are called Quantum Gates. We can do all sorts of operations with those gates. But the working of those gates is quite different from that of the classical logic gates. They are in the form of matrices (operators) which can act upon vectors to change their states.

Braitenberg Vehicles are very simple devices, yet the resulting behaviour may appear complex or even intelligent. With the aid of Quantum Computers, an excellent future application of Quantum Roots can be at our hands. As a practical example, quantum-controlled Braitenberg vehicles is a mobile quantum system and hence acts as a quantum robot.

I read a paper and tried to simplify the quantum circuits and algorithms. The aforementioned quantum circuit is designed with the help of IBM quantum experience.

Keywords: *Braitenberg Vehicles, Quantum Circuit, Quantum Gates, Quantum Robot, Qiskit, IBM Quantum Experience.*

Acknowledgement

I feel myself being privileged enough for getting so much assistance from Prof. Prasanta K. Panigrahi. I express my gratefulness for his forbearance to my work. I am thankful to Kumar Nilesh for his continuous support in this endeavour. Softwares like Microsoft Powerpoint and Dassault Solidworks and tools like the IBM Quantum Experience assisted me greatly. I thoroughly used the Python programming language throughout this project. Above all, my college seniors boost me greatly in my aim. I express my heartfelt gratitude for extending all your generous hands to me.

Contents

1 Introduction

The project primarily focused on learning and understanding quantum circuits. I thoroughly played around with IBM Quantum Experience and Qiskit to build intuitions about Quantum Gates and their uses in Quantum Circuits.

I read some papers which were provided to me by Prof. Panigrahi. Soon I came across the paper [Mahanti et al., 2019]. While reading the paper, I came up with two easier solutions to the problem addressed in the paper. My report talks about those very solutions.

2 Braitenberg Vehicles

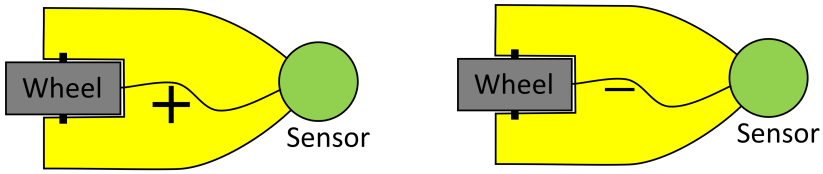
Valentino Braitenberg (1926-2011) was an Italian neuroscientist and a synthetic psychologist. He spent most of his life doing experiments to understand how the animal nervous system works. He was also interested in building synthetic models of human or animal behaviour. He is well known for his book [Braitenberg, 1986]. This book consists of a series of thought experiments about behaviours which can be expected of simple devices. All these thought experiments were inspired by animals and their nervous systems. He said "when we analyse a mechanism, we tend to over-estimate its complexity".

2.1 Working of a Basic Braitenberg Vehicle

These are simple vehicles consisting of actively driven wheels and sensors. The sensors are stimulated by some stimulus (light, temperature, sound, presence of certain chemicals etc). The rotation speed of the wheels is directly controlled by the two sensor data. Depending upon how many sensors and wheels are there and how the sensor data influences the wheel speed (either excitatory or inhibitor), Braitenberg vehicles have been divided into the following categories:

2.1.1 Vehicle 1

These vehicles (in Fig 1) consist of a single sensor and a single wheel. These devices cannot change their directions. They move in a straight line.



(a) Excitatory Signal (b) Inhibitory Signal

Figure 1: Braitenberg Vehicle Model 1

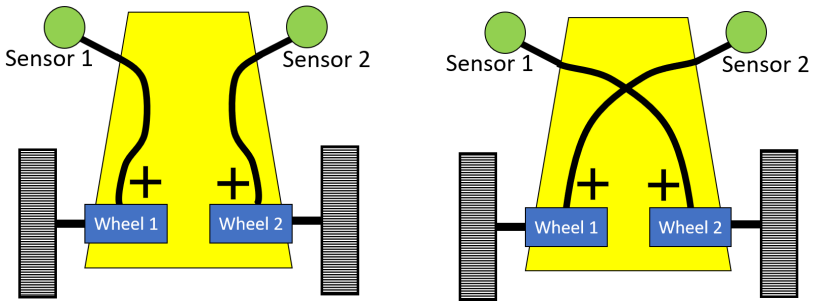
Now the sensor can either give an excitatory or an inhibitory signal to the wheel:

- **Excitatory Signal from Sensor:** Here, wheel speed is directly proportional to the strength of the stimulus. As the sensor senses more stimulus, (the stimulus is near) the wheel rotates faster; the vehicle gains speed. Thus, the speed of the vehicle is more when the vehicle is near the stimulus and the vehicle moves slowly when it is away from the stimulus. As a result, it spends less time near the source of stimulus and more time roaming away from it.
- **Inhibitory Signal from Sensor:** Here, wheel speed is inversely proportional to the strength of the stimulus. As the sensor senses more stimulus, the vehicle slows down. Thus, the devices gather around the source of stimulus.

2.1.2 Vehicle 2

These vehicles (in Fig 2) consist of two sensors and two wheels. Unlike Vehicle 1, they can move freely on a 2D plane. But here, the sensors can always give an excitatory signal to the wheels. They have been divided into three categories depending upon their connections:

- **Uncrossed connections (2a):** Here, the left sensor is attached to the left wheel and the right sensor is attached to the right wheel. Now, suppose a source of stimulus is present on the right side of the vehicle, the right sensor is stimulated more than the left one. Hence the right wheel rotates faster than the left wheel. Thus the vehicle turns left and goes away from the stimulus.



(a) Uncrossed Connection (b) Crossed Connection

Figure 2: Braitenberg Vehicle Model 2

Inference: The vehicle expresses **fear** from the cause of the stimulus. And is termed as **'COWARD'** by Braitenberg.

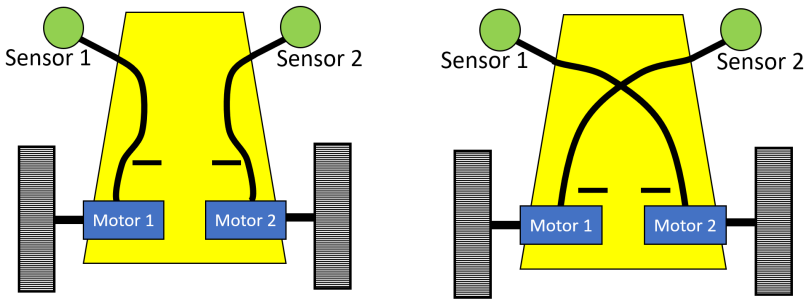
- **Crossed connections (2b):** In this case, the left sensor is attached to the right wheel and the right sensor is attached to the left wheel. Now, to a stimulus on the right, the system responds by rotating the left wheel faster. It moves towards the stimulus. As it moves closer to the stimulus, its speed increases. Soon, the vehicle collides with the stimulus at great speed.
  Inference: The vehicle expresses **aggression/hate** towards the cause of the stimulus.

- **Double connections (2c):** Here, both the sensors are connected to both the wheels. As both the wheels symultaneously recieve the same signal, this vehicle cannot turn (turning is achieved by different speed of the wheels). Thus, despite having a more complex connection, the vehicle is inferior to both 2a and 2b vehicles.
  Inference: More complex connections do not always give us more complex functionality.

In general, all the types of Vehicle 2 dislike the presence of the stimulus in their vicinity.

2.1.3 Vehicle 3

Just like the Vehicle 2 These vehicles (in Fig 3) also have two sensors and two wheels. As a result, they also have full manoeuvring control over the



(a) Uncrossed Connection (b) Crossed Connection

Figure 3: Braitenberg Vehicle Model 3

plane (2D). But here, the sensors can always give an inhibitory signal to the wheels. In absence of a stimulus, the wheels rotate at full speed, and the speed decreases on detection of stimulus. Depending upon their connections, they are of 3 types:

- **Uncrossed connections (3a):** The sensors are connected to their respective wheels. A stimulus on the right side of the vehicle slows down the right wheel. Thus the vehicle turns right and turns toward the stimulus. After turning, it has the stimulus just in front of it. Hence, both the wheels slow down and move at the same speeds. The vehicle heads towards the stimulus. The strength of the stimulus increases. Speed of the vehicle decreases. Finally, it comes to a halt near the stimulus.
  Inference: The vehicle expresses **love** for the cause of the stimulus. It sticks to the first stimulus it sees in its life. It does not care about anything else, nor does it go to other, more powerful stimuli.

- **Crossed connections (3b):** The left wheel is slowed by the right sensor and the left sensor slows down the right wheel. In the presence of a stimulus, although it moves away from it, it slows down. It spends some time near the stimulus but eventually moves away in search of new, stimuli.
  Inference: Braitenberg describes this vehicle as an **explorer**.

Braitenberg talked about a lot of other sorts of vehicles. Some had different types of sensors attached to the same vehicle (maybe one pair of sensors

detects temperature, another detects heat etc.) all of them give either positive or negative responses and may be connected in a crossed or uncrossed fashion. Some vehicles might have a different graph of influence (All this while we were talking about a proportional influence of sensors on wheels. But this might not always be the case. For example, an influence might be drastic, there is no slowing down - if the sensor receives a particular amount of stimulus, it completely turns the wheel on or off.). There are a lot of other types of simple vehicles discussed in detail by Braitenberg in his book. But this much discussion is enough for the purpose of this report.

3 The Problem and Modified Design

3.1 The Problem

We know that the 2b vehicle discussed above crashes into the stimulus with great speed. It shows aggression. That is not a very desirable thing to happen. Our main goal is to fix that and come up with a solution using quantum phenomena. To achieve this we modify the vehicle a bit. Till now, our vehicle was moving on a 2D plane. To avoid the collision we tried to get help from the 3rd dimension.

3.2 Proposed Modified Design

We have seen that the Vehicle 1 has a single wheel and it only moves along a straight line (1 wheel can only give us 1-dimensional motion). As we add another wheel in Vehicle 2 we get full access over the plane (2D). So, it is clear that to enable another degree of freedom, we have to add another actuator of some kind to make our vehicle fly above the stimulus (which acts as an obstacle here).

Now, *'flying'* in the real world is a tough problem in itself. So we won't go into the details of how the vehicle achieves this flying motion. For the scope of this report, we assume that we have done the required arrangements for the vehicle to fly. We say for simplicity that we have attached a propeller on top of the vehicle. It makes the vehicle fly, like a helicopter. Here, we won't discuss problems like the vehicle itself rotating in the opposite direction as a result of the reaction force of the rotating propellers. Let that be a problem for another day.

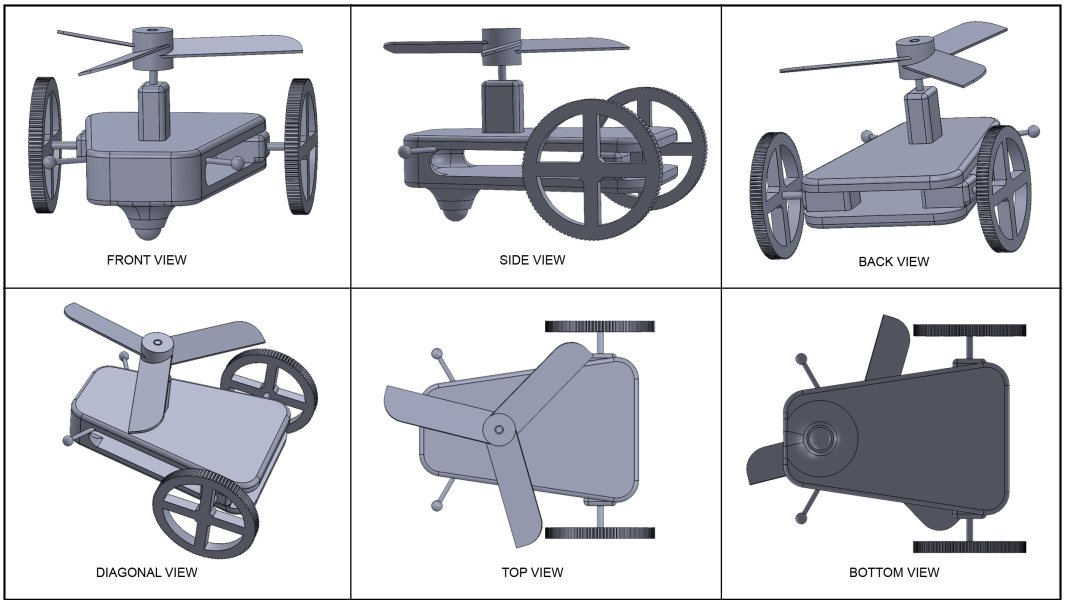


Figure 4: 3D model of the proposed design

So, finally, we are left with a vehicle which looks like Fig 4.

3.3 Robot Configurations

- **Sensors:** 2 photosensors (let's be specific about light as the stimulus from this point onwards) are attached on two sides of the vehicle for detecting the stimulus. They are digital quantum sensors. They return $|1\rangle$ if the intensity of light falling on them crosses a certain (predefined) value. Else, it returns $|0\rangle$.

- **Actuators:** 2 wheels on two sides and 1 propeller on top for taking off the ground.

4 Behaviour of the Vehicle

Our vehicle behaves quite differently from any of the basic Braitenberg Vehicles. The vehicle is equipped with 2 quantum photosensors which return digital output ($|0\rangle$ or $|1\rangle$). Depending upon the sensor values, the wheels either rotate or do not rotate. When both the sensors give $|0\rangle$, both the

Sensor Output		Actuator Input			Behaviour
Left	**Right**	**Left Motor**	**Right Motor**	**Propeller**	
0	0	1	1	0	Moves Forward
0	1	0	1	0	Turns Left
1	0	1	0	0	Turns Right
1	1	0	0	1	Takes off

Table 1: Behaviour of the Vehicle

wheels remain on. If one of the sensors is $|1\rangle$, the wheel of only that side remains on. If both the sensors are $|1\rangle$, and both the wheels are off, the propeller starts and the vehicle takes off. (the propeller remains off the rest of the time).

5 My Solutions to the Problem

The car gets binary inputs ($|0\rangle$ and $|1\rangle$) from two sensors attached to it. We need to build a quantum circuit to give the output to the motors like the *Table 1*.

In the [Mahanti et al., 2019] article, the authors tried to solve the problem by defining a new sort of quantum gate, the **two-control-two-target Toffoli ($\mathbf{C_2(X^3X^4)}$) gate**. They also made a large Quantum Circuit. But, I found some much easier and smaller approaches to the problem.

From the *Table 1*, we can say that:

- M1 = not(S2) = $\sim$ S2
- M2 = not(S1) = $\sim$ S1
- M3 = S1 and S2 = S1 $\wedge$ S2

I came up with two easier solutions:

1. **5 cubit:** This approach uses 5 qubits. We use two Quantum Registers for the sensor inputs, and the rest three for the motor outputs.
2. **3 cubit:** This approach uses 3 qubits. Here, we have two Quantum Registers for taking input from the two sensors. We process those signals and use them as inputs to the first two motor. We use the third Quantum Registers for output to the third motor.

5.1 Approach 1: 5 Cubit Solution

5.1.1 Importing the libraries

```
from qiskit import *
from qiskit.visualization import *
```

5.1.2 Making the Quantum Circuit

Making the Qiskit Classes:

```
# 1. making the sensor registers
sensors = QuantumRegister(2, 'sensor')
# 2. making the motor registers
motors = QuantumRegister(3, 'motor')
# 3. making the classical registers, 1 for each motors
creg_c = ClassicalRegister(3, 'measurements')
# 4. Adding the registers to get the quantum circuit
circuit = QuantumCircuit(sensors, motors, creg_c)
```

Initializing the Circuit: Demonstrating initialization with sensor value as = $|10\rangle$. Please feel free to try out the other states ($|00\rangle$, $|01\rangle$ and $|11\rangle$). This is Similar to the digitalRead function of Arduino.

```
ket_0 = [1, 0]
ket_1 = [0, 1]

# quantumRead = [ket_0, ket_0] # |00>
# quantumRead = [ket_0, ket_1] # |01>
quantumRead = [ket_1, ket_0]   # |10>
# quantumRead = [ket_1, ket_1] # |11>

circuit.initialize(quantumRead[0], [sensors[0]])
circuit.initialize(quantumRead[1], [sensors[1]])
```

Designing the circuit:

```
circuit.ccx(sensors[0], sensors[1], motors[2])
circuit.x(sensors[0])
circuit.x(sensors[1])
circuit.cx(sensors[0], motors[1])
circuit.cx(sensors[1], motors[0])
circuit.barrier()
circuit.measure(motors[0], creg_c[2])
circuit.measure(motors[1], creg_c[1])
```

```
circuit.measure(motors[2], creg_c[0])
circuit.draw(output = 'mpl')
```

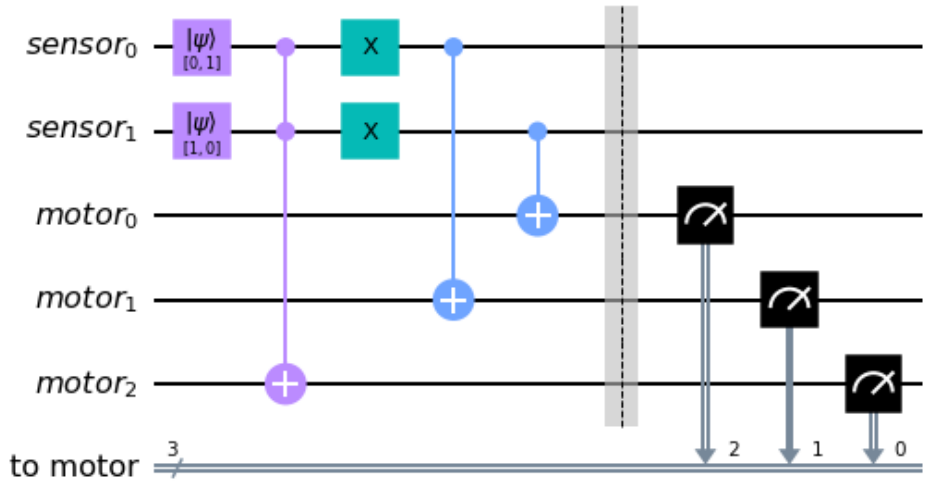


Figure 5: Circuit of the 5 Qubit Design

5.1.3 Executing the Quantum Circuit on a Simulator

```
comp = Aer.get_backend("qasm_simulator")
results = execute(circuit, comp, shots = 1024).result()
print("results:", results.get_counts(circuit))
plot_histogram(results.get_counts(circuit))
```

```
results: {'100': 1024}
```

We got the result $|100\rangle$ with 100% probability as output from the circuit after giving $|10\rangle$ as input. This matches with the *Table 1*. We also checked the other 3 input cases ($|00\rangle$, $|01\rangle$ and $|11\rangle$) and verified that the circuit is working as expected.

Note: The probability came out to be 100%, which is an ideal result. But in real world situations there is always some error involved in the measurement. The probability of getting the other results is not zero in those cases.

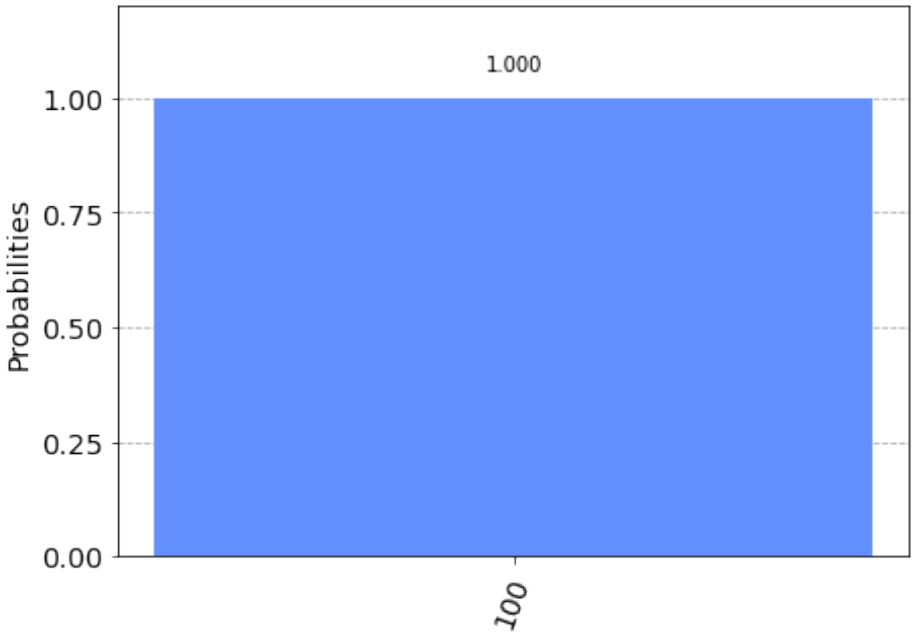


Figure 6: **Histogram:** 5 Qubit circuit executed on a simulator

5.1.4 Executing the Quantum Circuit on an actual Quantum Computer

Importing the relevant libraries

```
from qiskit.providers.ibmq import least_busy
from qiskit.tools.monitor import job_monitor
from qiskit.providers.ibmq.ibmqbackend import IBMQSimulator
# IBMQ.load_account(<API_Token>)
IBMQ.load_account()
```

```
<AccountProvider for IBMQ(hub='ibm-q', group='open',
  project='main')>
```

Choosing the least busy backend

```
provider = IBMQ.get_provider('ibm-q')
q_comp = least_busy(
  [x for x in provider.backends() if not isinstance(x,
   IBMQSimulator)]
  )
```

Executing the circuit on a Quantum Computer

```
job = execute(circuit, q_comp, shots = 16384)
job_monitor(job)
print("results:", job.result().get_counts(circuit))
plot_histogram(job.result().get_counts(circuit))
```

```
Job Status:  job has successfully run
results:
   {'000':  819,
   '001':  438,
   '010':  157,
   '011':  112,
   '100':  12416,
   '101':  1399,
   '110':  877,
   '111':  166}
```

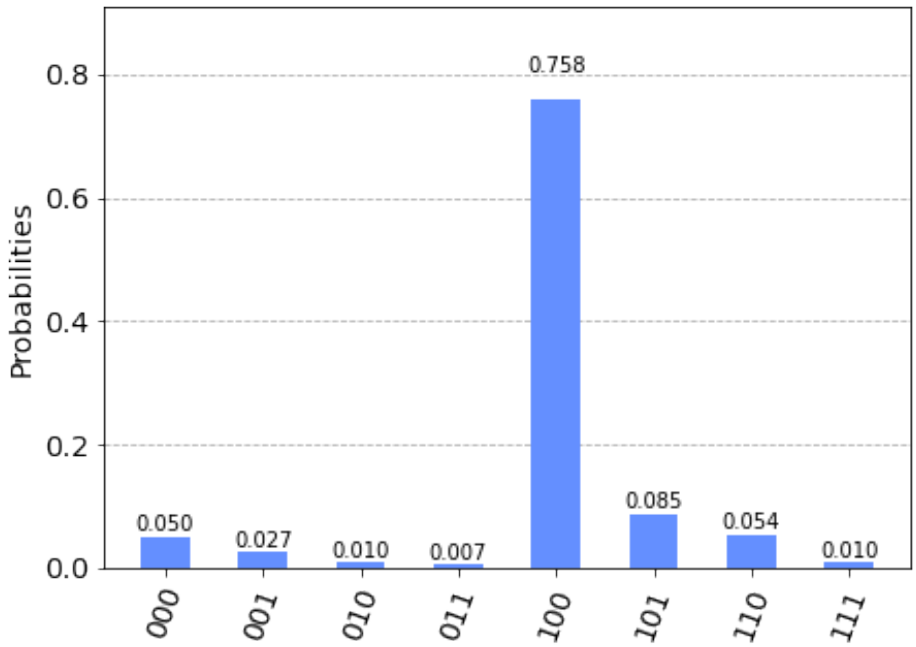


Figure 7: **Histogram:** 5 Qubit circuit executed on an actual Quantum Computer

5.2 Approach 2: 3 Cubit Solution

Importing the libraries: This part is the same as 'Importing the libraries' section from the Approach 1.

Making the Quantum Circuit:

```
# Making the Qiskit Classes:
qreg_s1_to_m2 = QuantumRegister(1, 's1_to_m2')
qreg_s2_to_m1 = QuantumRegister(1, 's2_to_m1')
qreg_to_m3 = QuantumRegister(1, 'to_m3')
creg_measurements = ClassicalRegister(3, 'measurements')
circuit = QuantumCircuit(
  qreg_s1_to_m2,
  qreg_s2_to_m1,
  qreg_to_m3,
  creg_measurements
  )

# Initializing the Circuit:
ket_0 = [1, 0]
ket_1 = [0, 1]
quantumRead = [ket_1, ket_0]   # |10>
circuit.initialize(quantumRead[0], [qreg_s1_to_m2[0]])
circuit.initialize(quantumRead[1], [qreg_s2_to_m1[0]])

# Designing the circuit
circuit.ccx(qreg_s1_to_m2[0],qreg_s2_to_m1[0],qreg_to_m3[0])
circuit.x(qreg_s1_to_m2[0])
circuit.x(qreg_s2_to_m1[0])
circuit.barrier()
circuit.measure(qreg_to_m3[0], creg_measurements[0])
circuit.measure(qreg_s1_to_m2[0], creg_measurements[1])
circuit.measure(qreg_s2_to_m1[0], creg_measurements[2])
circuit.draw(output = 'mpl')
```

Executing the circuits in both simulators and actual quantum computers: The codes are same as 'Executing the Quantum Circuit on a Simulator' and 'Executing the Quantum Circuit on an actual Quantum Computer' section from the Approach 1.

The result on the classical computer was exactly the same as the previous result. That on the Quantum Computer came out to be like Figure 8.

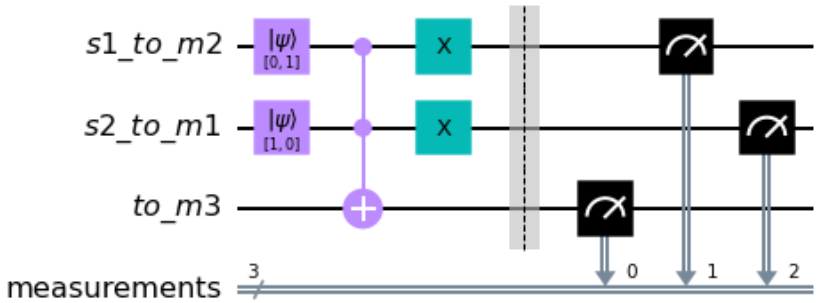


Figure 8: Circuit of the 3 Qubit Design

6 References

[Braitenberg, 1986] Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. Bradford Books. MIT Press. https://books.google.co.in/books?id=7KkUAT_q_sQC.

[Group, 2022] Group, Q. R. (2022). https://sites.google.com/site/quantumroboticsreadinggroup/home.

[Ju et al., 2007] Ju, Y.-L., Tsai, I.-M., and Kuo, S.-Y. (2007). Quantum circuit design and analysis for database search applications. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54:2552 – 2563. https://www.researchgate.net/publication/3451705_Quantum_Circuit_Design_and_Analysis_for_Database_Search_Applications.

[Mahanti et al., 2019] Mahanti, S., Das, S., Behera, B. K., and Panigrahi, P. K. (2019). Quantum robots can fly; play games: an ibm quantum experience. *Quantum Information Processing*, 18(7):1–10. https://arxiv.org/abs/1905.10968.

[Nielsen et al., 2000] Nielsen, M., Chuang, I., Chuang, I., and Chuang, I. (2000). *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press. https://books.google.co.in/books?id=aai-P4V9GJ8C.

[Prescott, 2020] Prescott, T. J. (2020). Braitenberg's vehicles. https://www.youtube.com/watch?v=DFXZc54_4GY.

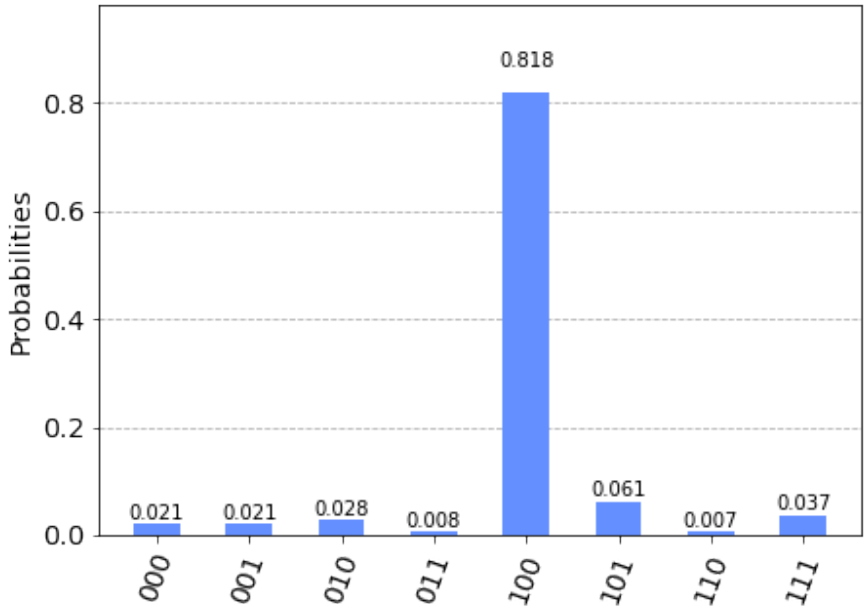


Figure 9: **Histogram:** 3 Qubit Circuit executed on an actual Quantum Computer

List of Figures

List of Tables