

## **MAZE NAVIGATION BY QUANTUM ROBOT: DEVELOPMENT OF QUANTUM MAZE SOLVING ALGORITHM**

YI CHEN LEE, NAI SHYAN LAI\*, ALEXANDER HON CHEONG CHEE

School of Engineering, Asia Pacific University of  
Technology and Innovation, Kuala Lumpur 57000, Malaysia

\*Corresponding Author: nai.shyan@apu.edu.my

### **Abstract**

Quantum Computing is a new and potential way of computing. It uses a different approach and mechanics compare to the current classical computing. The potential or highest boundaries of quantum computing is still unknown. Thus, the goal is to develop a quantum algorithm robot to solve the maze using tree diagram maze. A tree diagram maze has  $2^n$  possible paths, and the task is to find the shortest path among the paths. The quantum algorithm uses ratio-based as the approach in solving the maze. Qiskit from IBM has been used as the library where quantum algorithm is developed to solve the maze by executing the quantum circuit built from quantum gates. Two quantum algorithms have been developed and tested. The shortest path accuracy for the two quantum algorithms was averaging at 78% and 84%, respectively. Both quantum algorithms won over their corresponding classical opponents with the same approach.

Keywords: Maze-solving, Qiskit, Quantum algorithms, Quantum computing, Tree diagram maze.

## 1. Introduction

Quantum computing in robotic science and its applications is becoming more popular as years pass by. The demand for more powerful and intelligent robots via quantum computation is also rising. According to an article from [1], the quantum computing market is projected to reach \$64.98 billion by 2030 from just \$507.1 million in 2019. One of the applications of quantum computing in robotics is maze navigation. Quantum algorithms can be used to replace classical algorithms for maze solving and provide a quicker number of paths solving for better navigation of the robot.

Maze solving is a research area that can benefit people. By solving the number of paths in a maze, people can gather every possible path to the destination, which will benefit the most in the map navigation. Undoubtedly, a classical algorithm can perform the tasks quickly and simultaneously in finding possible paths between two cities. However, suppose the decision of the directions increases hugely such as finding the possible paths between two countries or even two continents. In that case, a massive increment in the decision will require a colossal increment of computation power and computation time. It might take a few hours to a few days for the decisions to be calculated. A quantum robot can be realised by using the open-source library through microprocessor. Here, a quantum algorithm will be developed using open-source library software to calculate all possible paths, and the result will then be used to navigate the robot in the maze. The formater will need to create these components, incorporating the applicable criteria that follow.

## 2. Literature Review

Table 1 reviewed journals in method to solve maze-related problems in a quantum way. The first method is Quantum Parallelism or Quantum Walk due to the existence of a superposition state for quantum particle. The second method is implementing Grover's Search Algorithm, which has been proven to take lesser iteration in finding the solution. The simplest way to solve the maze in a quantum way is by building a circuit of Quantum Parallelism from quantum logic gates. This method is proposed and testified in [2, 3]. It required lesser hardware resources to perform the operations as no actual quantum devices such as quantum annealer is needed.

Table 2 reviewed journals in comparison between the quantum software platforms. Some of the well-known platforms are the pyQuil, Qiskit, ProjectQ, Q# and Cirq. These platforms were mainly compared in term of features, supported libraries and documentation resources. As a result, Qiskit from IBM has the upper hand among these platforms. The reasons are it is beginner-friendly in term of enormous resources available. Furthermore, Qiskit is also user-friendly as it has a lower degree of complexity in implementation. Table 3 reviewed journals in showing or teaching of the quantum gates or quantum circuit. By researching on this area, the characteristic and function of each quantum gates have been learnt. Besides, the effect of different combination of quantum gates have also been observed. Lastly, the programming method to build the quantum circuit from the quantum gates have also been explained.

**Table 1. Summary of literature review in quantum algorithm research.**

No.	Author	Method to solve maze	Advantages	Limitations
1.	[2]	Quantum Parallelism using Quantum Gates	Simple	Not applicable for all maze
2.	[3]	Quantum Parallelism using Quantum Gates in simulation	Simple	Not entirely autonomous (require manual inputs)
3.	[4]	Quantum version of Dijkstra algorithm by using Quantum circuit design	Simple	Conversion is complicated and unclear
4.	[5]	Quantum Walk by generating quantum particle to 'walk' in the maze	Rapid and genuine	Unstable and require actual quantum hardware
5.	[6]	Quantum Walk of state transfer with Grover's Search	Speedup in required steps	Not applicable for all types of maze and tasks are complicated
6.	[7]	Grover's Search for maximum fitness value	Significant Speedup	Not fully suited for shortest path problem and Inconvenient due to large number of loops
7.	[8]	Simulate in Quantum Annealing	-	Only applicable in certain condition

**Table 2. Summary of literature review in quantum software platform.**

No.	Author	Aim	Outcome
1	[9]	To compare the open- source quantum software	pyQuil and Qiskit perform better
2	[10]	To compare the quantum software platform	Qiskit wins with its all-aroundness
3	[11]	To compare the quantum software platform	Selected Qiskit due to low degree of complexity

**Table 3. Summary of literature review in quantum gates and circuit.**

No.	Author	Outcome
1	[12]	Fundamental quantum gates' characteristic and function have been learnt
2	[13]	The simulation effect of combining the quantum gates have been learnt
3	[14]	The method for building the quantum circuit from quantum gates have been learnt

### 3. Concept Design and Research Methodology

#### 3.1. Selection of quantum software platform

Quantum Software Platform is the platform for the user to program the quantum code or simulate the quantum circuit. With the additional help from the literature review, three platforms created by the solid and well-known institution will be compared. The three platforms are the pyQuil from Google, Qiskit from IBM and QDK from Microsoft as shown in Table 4. In comparison, a programming language will determine the microprocessor or control board in building the robot. In this case, pyQuil and Qiskit are using Python while the QDK is using C#. On the other hand, the number of simulation qubits will determine the maximum limitation of the generated maze. As for a basic quantum algorithm, a qubit can perform one decision. Thus, a platform will 20 qubits can solve a maze with around 20 decisions. Therefore, the higher the available simulation qubits, the better the project can be.

Next, the feature of simulating in an actual backend quantum device is also essential. Simulating the results in a real quantum computer can provide more realistic results. Only pyQuil and Qiskit can perform the simulation in the backend quantum computer, and Qiskit can perform an actual simulation up to 20 qubits. Lastly, pyQuil and Qiskit have vast libraries of resources on the internet, including tutorials and projects. QDK has been eliminated in the first place as it does not provide simulation on a real quantum computer. Next, Qiskit has the upper hand in performing real quantum device simulation up to 20 qubits compared with eight qubits from pyQuil. In conclusion, Qiskit is selected as the quantum software platform.

**Table 4. Comparison table of quantum software platform.**

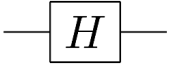

	pyQuil	Qiskit	QDK
<b>Institution</b>	Google	IBM	Microsoft
<b>Programming Language</b>	Python	Python	C#
<b>Quantum Language</b>	Quil	OpenQASM	-
<b>No. of simulation qubits (for free)</b>	Local: 20 API key: 25	Local: 25 Cloud: 30	Local: 30 Cloud: 40
<b>Actual Quantum Device</b>	8 qubits device	IBMQX2 (5 qubits), IBMQX4 (5 qubits), IBMQX5 (16 qubits), QS1 1 (20 qubits)	-
<b>Resources</b>	High	High	Low

#### 3.2. Concept design derived from fundamental engineering principles

In quantum mechanics, Bra-Ket notation is used. ‘Ket’ notation is the combination of the vertical bar “|” and the right-angle bracket “>” to represent

the state of the quantum system. Thus,  $|0\rangle$  is representing the ‘0’ and  $|1\rangle$  is representing the ‘1’. Additionally, the matrix form of  $|0\rangle$  is  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$  and  $|1\rangle$  is  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ . Hadamard Gate is a quantum logic gate that acts on a qubit. If a qubit moves through the Hadamard Gate, it will convert into a superposition state of having ‘0’ and ‘1’ at the same time. The Hadamard Gate is represented by the symbol ‘H’ as shown in Fig. 1. Table 5 shows the truth table of Hadamard Gate. In conclusion, when the Hadamard Gate is applied, the output will be in the superposition of having ‘0’ and ‘1’ at the same time. This output will help in the realisation of Quantum Parallelism.

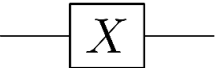
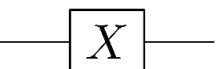
**Table 5. Truth table of Hadamard gate.**

Input	Logic Gate	Output
$ 0\rangle$		$\frac{ 0\rangle +  1\rangle}{\sqrt{2}}$
$ 1\rangle$		$\frac{ 0\rangle -  1\rangle}{\sqrt{2}}$

**Fig. 1. Hadamard gate.**

Pauli-X is a quantum gate that acts on a qubit. If a qubit moves through the Pauli-X Gate, it will convert into the opposite state. In this case, the Pauli-X Gate works exactly like the NOT Gate in a classical way. The Pauli-X Gate will be showed in the symbol ‘X’ as showed in Fig. 2. Table 6 shows the truth table of Pauli-X Gate. In summary, when  $|0\rangle$  moves into the Pauli-X Gate, the output will be  $|1\rangle$ . On the other side, when  $|1\rangle$  moves into the Pauli-X Gate, the output will be  $|0\rangle$ .

**Table 6. Truth table of Pauli-X gate.**

Input	Logic Gate	Output
$ 0\rangle$		$ 1\rangle$
$ 1\rangle$		$ 0\rangle$

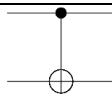
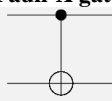
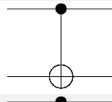

**Fig. 2. Pauli-X gate.**

Controlled Pauli-X Gate as shown in Fig. 3 is a quantum logic gate that requires two qubits to operate. One qubit will be the controlled qubit, while the other one will be the target qubit. The controlled qubit will determine when will the target qubit to undergo the NOT Gate effect. The controlled qubit is represented by a dot, while the NOT-type target qubit is represented by a ‘+’ inside a circle. Table 7 shows the truth table of Controlled Pauli-X Gate. In summary, when the controlled qubit is  $|0\rangle$ , the operated qubit will remain the same. However, when the controlled qubit is  $|1\rangle$ , the operated qubit will perform NOT gate effect.

Controlled Hadamard Gate as shown in Fig. 4 is a logic gate that requires one controlled qubit and one target qubit. The target qubit is controlled to perform

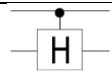
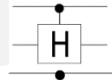
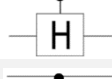
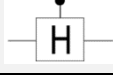
Hadamard Gate by the controlled qubit. Similarly, the controlled qubit is represented by a dot, while the Hadamard-type target qubit is represented by a block with symbol 'H'. Table 8 shows the truth table of Controlled Hadamard Gate. In summary, when the controlled qubit is  $|1\rangle$ , the target qubit will perform Hadamard Gate to produce the superposition state.

**Table 7. Truth table of controlled Pauli-X gate.**

Input		Logic Gate	Output	
Controlled	Target		Controlled	Target
$ 0\rangle$	$ 0\rangle$		$ 0\rangle$	$ 0\rangle$
$ 0\rangle$	$ 1\rangle$		$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$		$ 1\rangle$	$ 1\rangle$
$ 1\rangle$	$ 1\rangle$		$ 1\rangle$	$ 0\rangle$

**Fig. 3. Controlled Pauli-X gate.**

**Table 8. Truth table of controlled Hadamard gate.**

Input		Logic Gate	Output	
Controlled	Target		Controlled	Target
$ 0\rangle$	$ 0\rangle$		$ 0\rangle$	$ 0\rangle$
$ 0\rangle$	$ 1\rangle$		$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$		$ 1\rangle$	$\frac{ 0\rangle +  1\rangle}{\sqrt{2}}$
$ 1\rangle$	$ 1\rangle$		$ 1\rangle$	$\frac{ 0\rangle -  1\rangle}{\sqrt{2}}$

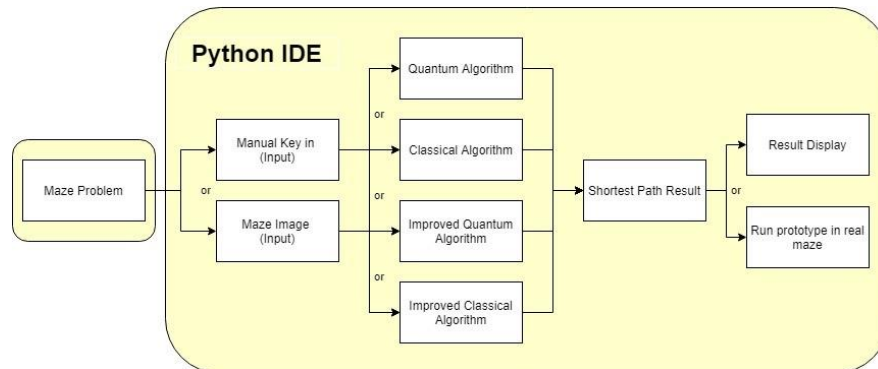
**Fig. 4. Controlled Hadamard gate.**

## 4. Final Design and Implementation

### 4.1. Overall block diagram

Figure 5 shows the overall block diagram of the project. In order to run the system, a maze problem must be identified. Next, the useful maze information must be able to extract from the maze problem. There are two ways of extracting the usable maze information to use in the system. The first way is by manually key in the maze information, such as the number of junctions and all path length values. On the other hand, the second way is by importing the image of the maze problem into the system. Then, the system will extract all useful information by itself through the

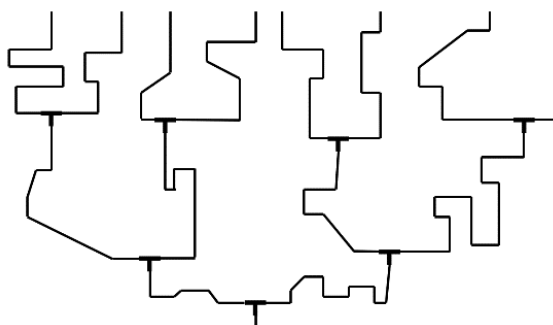
image processing method. After collecting all necessary information, the maze problem can now be solved using the maze solving algorithm. There are four types of mazes solving algorithms, including two quantum algorithms and two classical algorithms. As for clarification, the two quantum algorithms are the main ones in this project, while the two classical algorithms are only developed as a comparison target in the testing part. Then, the shortest path result will be obtained after solving the maze by one of these algorithms. Lastly, the shortest path result can have two outputs. The first output is by displaying the results on the python IDE as a simulation purpose of solving the maze. On the other hand, the second output is by decoding the shortest path result and run the result in a real-time prototype in the actual maze.



**Fig. 5. Overall block diagram of the project.**

#### 4.2. Construction details (maze)

The first constructional details will be the details of the maze. The type of maze is set to use the tree diagram maze as shown in Fig. 6. This type of maze will have a single starting point and multiple destination points. The target of solving this type of maze is to determine which combination of paths will provide the shortest route. Several simple rules have to follow to build this kind of maze. Firstly, the maze will be divided into several layers. For each layer, it must have  $2^{n-1}$  junctions. Then, for each junction, it must have two possible paths. Therefore, the total number of possible paths will depend fully on how many layers are in the maze with a formula of  $2^n$ . For instance, a ten layers maze will have  $2^{10} = 1024$  possible paths.



**Fig. 6. Example of tree diagram maze.**

### 4.3. Construction details (algorithm)

The second constructional section will be the details of the development of the algorithm. A new maze solving approach will be built. The same approach will be implemented into the quantum way and classical way for comparison in accuracy tests. The newly designed approach is a ratio-based method in solving the maze. The main logic of this approach is to help the robot determine the decision of going left or going right when encountering the junctions to move in the calculated shortest path. The determination of going left or going right will be based on the ratio of the left paths with the right paths. Each layer will have to calculate its own ratio. The basic formula of this approach will be listed in Eq. (1) below.

$$\text{Ratio} = \frac{\sum \text{All Left Paths}}{\sum \text{All Right Paths}} \quad (1)$$

Figure 7 shows an example of one layer maze. If the length of 'a' is 10 unitless while 'b' is 8 unitless, the ratio will be  $\frac{a}{b} = \frac{10}{8} = 1.25$ . If the ratio is larger than 1, it means that the left path is longer than the right path. Thus, by going right, it has a higher possibility of getting the shortest path.

Similarly for a two layers maze, a two layers maze will have one junction at Layer 1 and two junctions at Layer 2. Based on Fig. 8, the Layer 1 ratio will be  $\frac{a}{b}$  while the Layer 2 ratio will be  $\frac{c+e}{d+f}$ . Assume that both layer ratios are more than 1, which means that the robot is taught to turn right when encountering the first junction and the second junction. Therefore, the calculated shortest path result by the system will be START  $\rightarrow$  b  $\rightarrow$  f. Therefore, based on the two examples above, the system's output teaches the robot to decide whether to turn left or turn right when seeing the junctions. This maze solving approach that has been written in a quantum way will be known as the 'Quantum Algorithm' throughout the entire report. On the other hand, the same approach that has been written in a classical way will be assigned with the name of 'Classical Algorithm' throughout the whole report. Furthermore, a higher degree of consideration has been applied to the above approach to increase accuracy. Even though it is logical and should be working fine from the above approach, some scenarios still disobey the above approach.

Assume a maze with the path length listed in Table 9 above. The focus point in this example is that the average path length values in Layer 2 are significantly way higher than the average path length values in Layer 1. Thus, Layer 2 should be more focused or more dominant in calculating the shortest path results for logical thinking. The path length values in Layer 1 will only contribute a minor effect in the final path length values. The shortest path ranking can be stated in Table 10.

$$\text{Average Path Length (Layer 1)} = \frac{5+7}{2} = 6 \quad (2)$$

$$\text{Average Path Length (Layer 2)} = \frac{50+45+20+33}{4} = 37 \quad (3)$$

Based on the previous mentioned approach, the Layer 1 ratio will be  $\frac{5}{7} = 0.714$  which is lower than 1. Next, the Layer 2 ratio will be  $\frac{50+20}{45+33} = 0.897$  which is also lower than 1. Thus, the calculated shortest path result of using the above approach will be left at the first junction and left at the second junction with the code result of



'00'. This example shows that the above approach has weaknesses in several cases, such as huge value differences in average path length values between the layers.

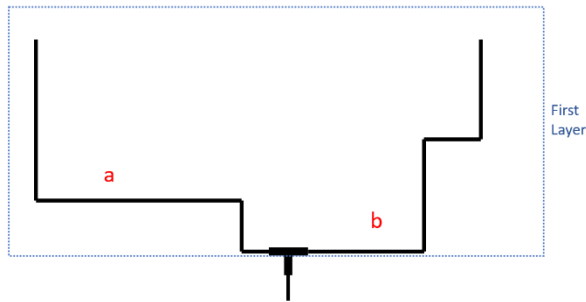


Fig. 7. Example of one layer maze.

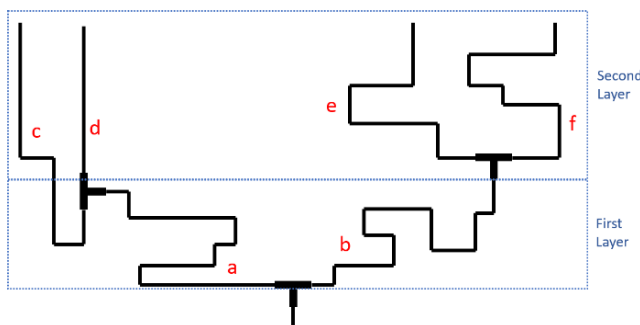


Fig. 8. Example of two layers maze.

Table 9. Example of two layers maze with path length.

	Path Length				
Layer 2	50	45	20	7	33
Layer 1	5		7		

Table 10. Shortest path ranking for the paths.

	Path			
	00	01	10	11
Total Path Length	55	50	27	40
Shortest Path Ranking	4 <sup>th</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	2 <sup>nd</sup>

Therefore, modifications have been added to the original approach to come out with an improved approach version. There is one sentence to conclude the whole modifications, which is the following layer's information must also be a part to affect the previous layer's ratio. The modifications are adding more linkages or relationships between the layer's ratios. In this improved approach, the first layer (Layer 1) ratio will also be affected by the following layer's information.

Assume that the scenario has calculated to turn left at the first junction; the robot or the human follows the instruction to turn left at the first junction. However, after turning left at the first junction, the following two paths are extremely long compared to the other two paths on the right side. In this case, turning left at the first junction might be a wrong decision in the first place. Therefore, this scenario has proved that the following layers must also be considered when calculating the decision in the first layer (Layer 1).

This improved approach has also been written in both quantum and classical ways for comparison. The quantum way with the improved approach will be called the 'Improved Quantum Algorithm', while the classical way with the improved approach will be called the 'Improved Classical Algorithm'.

To summarise, four algorithms will be tested: the 'Quantum Algorithm', 'Improved Quantum Algorithm', 'Classical Algorithm', and 'Improved Classical Algorithm'.

#### 4.4. Construction details (quantum circuit)

The third construction detail will be the details for building the quantum circuit. After getting all the layer's ratios, the information will be run into the quantum circuit for the 'Quantum Algorithm' and 'Improved Quantum Algorithm'. The quantum circuit will start the calculation and output all the path's possibilities. The path with the highest possibility value will be the shortest path. On the other hand, the path with the lowest possibility value will be the longest path. However, the entire project will mainly focus on the shortest path only.

Build a quantum circuit requires quantum gates. For the project, three types of quantum gates have been used: the Hadamard Gate, Controlled Hadamard Gate, and Pauli-X Gate. By combining different gates, it will represent different scenarios. For example, applying a single Hadamard Gate to a qubit will output a result of 50% '0' and 50% '1'. This output can represent the scenario when the left and right paths have similar lengths of equal lengths. Therefore, the scenario should be 50% going left and 50% going right, which is exactly similar to the output from the quantum circuit. Figure 9 shows the quantum circuit of one Hadamard Gate and its simulation result in histogram for the scenario when layer ratio is 1.

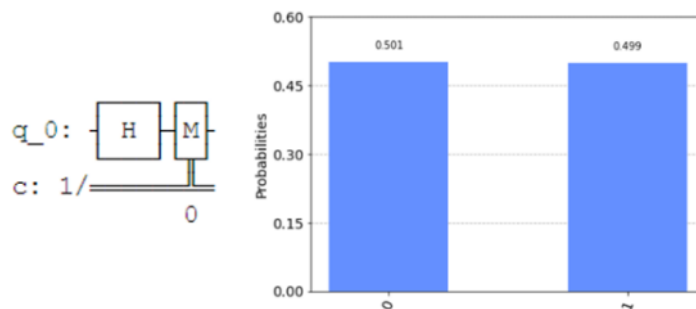


Fig. 9. Quantum circuit and simulation result for ratio = 1 scenario.

If the layer ratio is 0.5, the left path is two times shorter than the right path. Thus, the possibility of going left should be 66.67%, and going right should be 33.33%. The scenario can also be represented by combining one Hadamard Gate,

one Pauli-X Gate and two Controlled Hadamard Gates. Figure 10 shows the quantum circuit and simulation results in histogram for scenario when ratio is 2.

For the opposite of the previous scenario, if the layer ratio is 2, the right path is now two times shorter than the left path. Thus, the possibility of going left and going right will be 66.67% and 33.33%. In this case, the gates' combination will be the same as the previous example but with an extra Pauli-X gate added at the most end of the circuit. Figure 11 shows the quantum circuit and simulation result in histogram for scenario when layer ratio is 2.

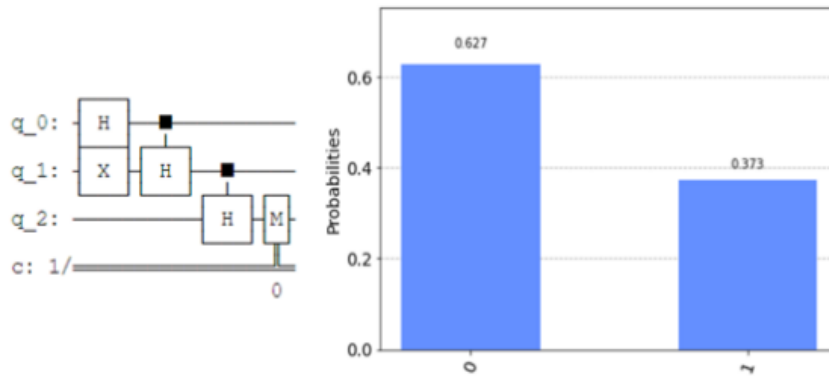


Fig. 10. Quantum circuit and simulation result for ratio = 0.5 scenario.

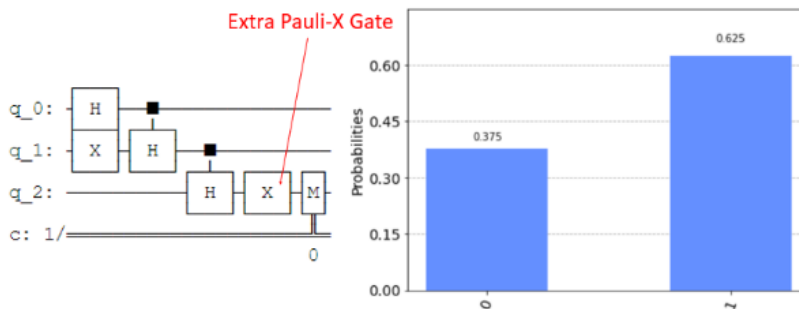


Fig. 11. Quantum circuit and simulation result for ratio = 2 scenario.

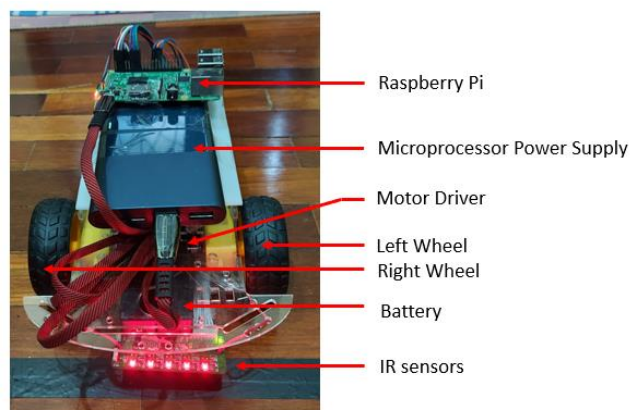
Thus, ratio =  $x$  and ratio =  $\frac{1}{x}$  such as ratio = 0.5 and ratio = 2 are using almost the same combination of quantum gates. The only difference will be adding an extra Pauli-X Gate at the end of the circuit for ratio > 1 part. In this case, the ratio = 2 case will be the one that needs an extra Pauli-X Gate. The combination of quantum gates in building the quantum circuit to represent ten different ratio scenarios has been listed in Table 11. It has listed out the scenarios for ratio =  $x$  when  $x < 1$ . On the other hand, add an extra Pauli-X Gate at the end of the combination for ratio =  $\frac{1}{x}$  to obtain the combinations for representing the other ten different ratio scenarios. By having these 10+10 combinations of gates, it can represent 20 different scenarios of the layers in the maze.

**Table 11. Combination of quantum gates circuit for different ratio scenario.**

Ratio	Quantum Circuit	% getting '0'	% getting '1'
$Ratio < 0.03$		0.97	0.03
$0.03 \leq Ratio < 0.11$		0.95	0.05
$0.11 \leq Ratio < 0.2$		0.875	0.125
$0.2 \leq Ratio < 0.3$		0.8	0.2
$0.3 \leq Ratio < 0.4$		0.75	0.25
$0.4 \leq Ratio < 0.47$		0.7	0.3
$0.47 \leq Ratio < 0.65$		0.667	0.333
$0.65 \leq Ratio < 0.9$		0.55	0.45
$0.9 \leq Ratio < 1$		0.5	0.5

#### 4.5. Construction details (prototype)

The fourth construction detail is the details of the prototype building. It shows the construction requirements and steps in building the prototype robot, as shown in Fig. 12. The wiring connections of the components' wires to the Raspberry Pi pin are shown in Table 12. Since the maze solving algorithm result only outputs the decisions of turning left or turning right at the junction, the path detecting approach must identify the junctions correctly so that the decisions can be applied as shown in Table 13. Therefore, making the prototype capable of running in real-time mazes requires a path detecting approach to identify and differentiate the maze's roads/paths. The general logic behind constructing the path detecting approach is simple. If the robot detects the junction, the robot will use the algorithm's calculation results, such as turning left at the first junction and turning right at the second junction. On the other hand, if the junctions have not been detected, the robot will move in the maze based on a line follower alike logic until it encounters a junction. Next, to move in the real-time maze successfully, it must fully utilise the motion of two wheels to perform several movements such as turn right, turn left, backward or forward movement. The motion of two wheels is controlled by the motor driver with the method of Pulse Width Modulation (PWM) as shown in Table 14. The PWM method relies on the voltage supply to the motor driver. Thus, the details of the voltage supply will be listed below. For the prototype, the voltage is supplied from two 3.7V Li-Ion Battery in series, contributing to a total of 7.4V voltage supply.



**Fig. 12. Prototype of the project.**

**Table 12. Wiring table of the project.**

No.	Component's Wire	GPIO
1	Left Wheel Enable	1
2	Left Wheel Input 1	8
3	Left Wheel Input 2	7
4	Right Wheel Enable	14
5	Right Wheel Input 1	18
6	Right Wheel Input 2	15
7	IR Sensor 1	22
8	IR Sensor 2	27
9	IR Sensor 3	17
10	IR Sensor 4	3
11	IR Sensor 5	2

**Table 13. Combination of IR sensors values to corresponding motion.**

IR Sensor					Motion
IR1	IR2	IR3	IR4	IR5	
0	0	1	0	0	Forward
0	1	1	1	0	Forward
0	0	1	1	0	Slightly Turn Right
0	1	1	0	0	Slightly Turn Left
0	X	1	X	1	Turn Right
1	X	1	X	0	Turn Left
1	X	1	X	1	JUNCTION
0	0	0	0	1	Slightly Turn Right
1	0	0	0	0	Slightly Turn Left
1	0	0	0	1	Backward
0	0	0	0	0	Stop
X	0	0	1	X	Slightly Turn Right
X	1	0	0	X	Slightly Turn Left
ELSE					Backward

**Table 14. PWM (%) of the motors in performing the motions.**

Motion	PWM (%)	
	Left Wheel Motor	Right Wheel Motor
Forward	40	40
Backward	40	40
Stop	0	0
Slightly Turn Left	0	40
Slightly Turn Right	40	0
Turn Left	0	50
Turn Right	50	0

## 4.6. Discussion—project finding and testing

### 4.6.1. Introduction

The two quantum algorithms are the primary testing target for the tests. Six tests are carried out on the shortest path accuracy test for different path layers maze, the shortest path accuracy test for random maze images, the shortest path accuracy test for random path length difference, the shortest path accuracy test for different quantum shot values, the computation time test, and the compatibility test for the prototype.

### 4.6.2. Shortest path accuracy test for different path-layers maze

The first test will be the shortest path accuracy test for different path layers maze. Due to system limitations, the test can only be done using one layer maze up to seven layers maze. The test was set to discover the ability of quantum algorithms in solving different layers of mazes. It is also a test for the quantum algorithms to compare and compete with their same approach classical version algorithms. The two quantum algorithms and two classical algorithms had been applied in this shortest path accuracy test. The maze information, such as the path length for the test were randomly generated by the system using the random key in mode. In the end, 1000 sets of maze information were generated for each number of layers maze,

which will contribute to a total of 7000 sets of different mazes for testing purposes. The term ‘accuracy’ has been defined as the ability or correctness of the algorithm in finding the shortest path. It uses similar logic as calculating the marks for any exam. For example, assume that a person answered 90 questions correctly out of 100 questions; that person should be marked as 90% for the exam. Similarly, if there are 100 paths in the mazes, the system that output the second shortest path among all paths should be granted 98% ~ 99% for the correctness or accuracy of the system.

When the number of layers in a maze increases, the number of paths will increase exponentially, leading to a massive number of possible paths. In this case, although the system may not be able to output the shortest path, the system that can output the second shortest paths out from few thousand or a few hundred thousand paths should also be considered as high accuracy or high correctness. Throughout the testing, there are several values to be calculated and recorded for the algorithms. Firstly, the shortest path accuracy percentage of each algorithm will be calculated. From the formula, the ranking index is the output path result ranking of the algorithm. If the algorithm outputs the second shortest path result, the ranking index will be ‘2’, which indicates the second shortest path. On the other hand, the number of paths in the formula is the total number of possible paths in the maze. For example, a three layers maze will have a total of eight possible paths.

$$\text{Shortest Path Accuracy (\%)} = 100 - \left( \frac{\text{Ranking Index} - 1}{\text{Number of Paths} - 1} \times 100\% \right) \quad (4)$$

Secondly, the average shortest path accuracy percentage will also be calculated and displayed. For the accuracy test, each layer maze has been generated 100 sets of maze data per time and continuing generated ten times. The ten sets of 100 mazes data will be recorded in tabular form. The average percentage is the average value for the ten sets of 100 mazes data which can also be known as the average shortest path accuracy percentage for the 1000 mazes data.

$$\text{Average Shortest Path Accuracy (\%)} = \frac{\sum \text{Shortest Path Accuracy (\%)}}{10} \quad (5)$$

Figure 13 shows the average shortest path accuracy for the four algorithms including two quantum algorithms and two classical algorithms. The analysis of the testing results will be classified into simple analysis and advanced analysis. As a simple analysis, the quantum algorithms and classical algorithms can be compared by comparing the average shortest path accuracy for all layers as shown in Table 15.

From the above Fig. 14 and Table 15, there are two conclusions. Firstly, the two quantum algorithms have a better accuracy compared to their classical algorithms’ opponents. Secondly, it is undeniable that the accuracy can be increased by a deeper consideration of the algorithm approach. The analysis has shown that the algorithm is capable and can improve in the future by modifying and improving the current logic approach. Microsoft Excel can produce the trendline equation for the plotted graph. Since the testing was done only for up to seven layers due to system limitations, it will be great to have a trendline equation to forecast the future number of layers’ conditions. Thus, the trendline equation will be produced to further extend the accuracy percentage graph for each algorithm. The trendline equation for both algorithms is set to be in the logarithmic equation as shown in Table 16.

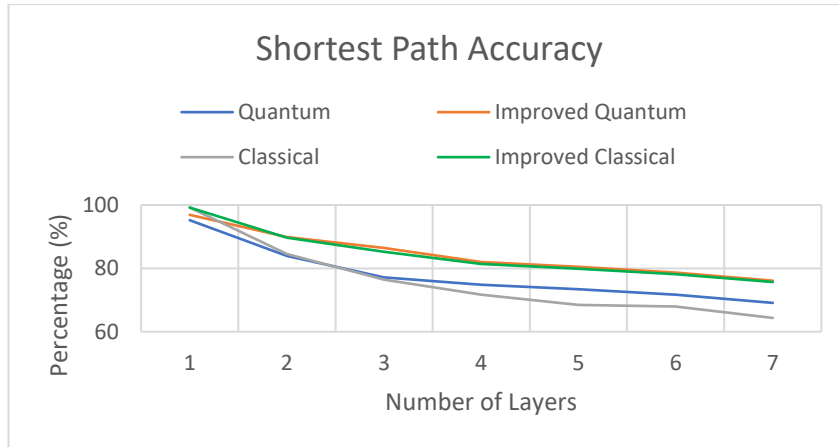


Fig. 13. Average shortest path accuracy for algorithms.

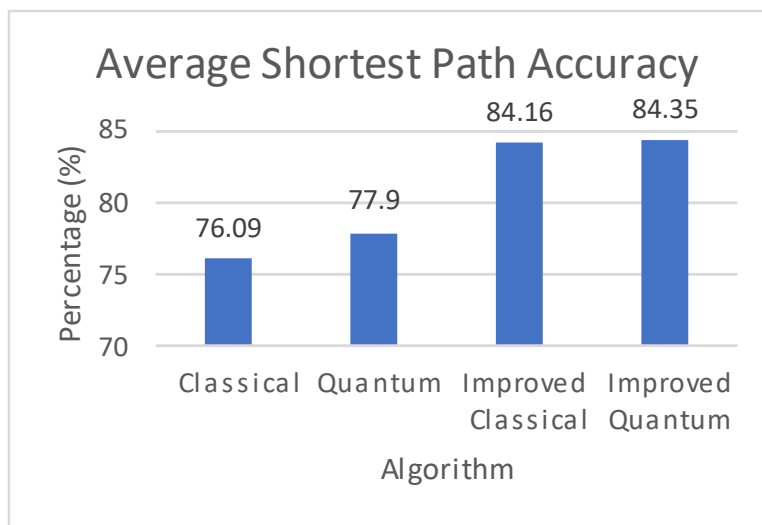


Fig. 14. Average shortest path accuracy histogram.

Table 15. Average all layer's shortest path accuracy.

Number of Layers	Algorithm			
	Classical	Quantum	Improved Classical	Improved Quantum
1-L	99.2	95.2	99.2	96.9
2-L	84.568	83.967	89.635	89.867
3-L	76.428	77.115	85.188	86.456
4-L	71.646	74.78	81.36	81.979
5-L	68.425	73.436	79.848	80.507
6-L	67.996	71.705	78.174	78.655
<b>Average</b>	<b>76.09</b>	<b>77.9</b>	<b>84.16</b>	<b>84.35</b>



**Table 16. Trendline equation for the algorithms.**

Algorithm	Trendline Equation
Classical	$y = -17.62\ln(x) + 97.545$
Quantum	$y = -12.93\ln(x) + 93.653$
Improved Classical	$y = -11.75\ln(x) + 98.471$
Improved Quantum	$y = -10.52\ln(x) + 97.165$

For the advanced analysis, the algorithms can be further analysed in terms of their workable range. The workable range is the current capable range for the algorithm to solve the maze. As the number of layers increases, it might be more challenging for the system to maintain the correctness or output a good accuracy since there are more possibilities and more complex. At that time, the system might output an accuracy of around 0%, which is unusable as it can no longer calculate the shortest path. However, the system might be useless earlier way before reaching 0%. Thus, 50% accuracy was set as the minimum usable range for the system. Using the accuracy trendline equation of the algorithms can calculate the workable range of the algorithms in solving the mazes. The variable 'y' in the equation is the accuracy percentage, while the 'x' in the equation is the number of layers. Table 17 can conclude that the quantum algorithms have a more extensive workable range and usable range than the classical algorithms. It has another inner meaning: the accuracy of classical algorithms drops faster or more significantly than quantum algorithms. Thus, the quantum algorithms are better than the classical algorithms in terms of accuracy.

**Table 17. Workable range for the algorithms.**

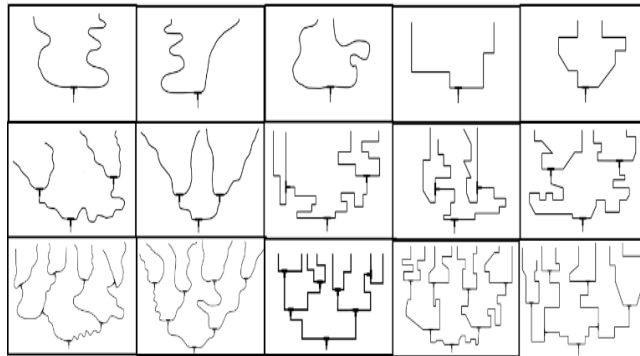
Algorithm	Number of Layers	
	50% Accuracy	0% Accuracy
Classical	14 layers	253 layers
Quantum	29 layers	1398 layers
Improved Classical	61 layers	4361 layers
Improved Quantum	88 layers	10262 layers

#### 4.6.3. Shortest path accuracy test for random maze image

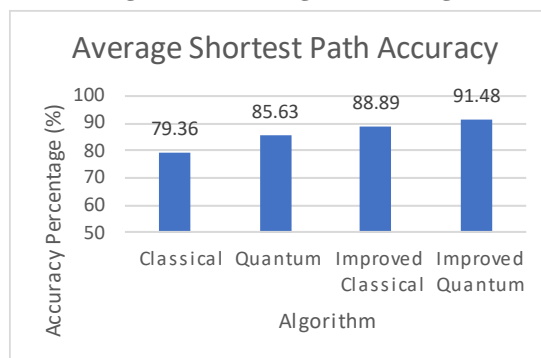
The second test is the shortest path accuracy test for maze images. The system consists of a mode to solve the maze by using maze images as the maze information input. The test setup is by preparing the maze images and inserting the maze images into the system for the algorithms to solve the maze. For the testing, 15 different maze images have been prepared, as shown in Fig. 15. The 15 maze images consist of layer maze up to three layers maze. The system will test each maze image for 1000 times.

The main objective of having this test is to discover the performance of the quantum algorithms in solving the maze by inputting maze images. At the same time, the quantum algorithms will be compared and competed with their classical version of the same approach to show the potential of the quantum algorithms. From Fig. 16 and Table 18, it can be observed that the improved quantum algorithm has the highest accuracy of 91.48%. The improved quantum algorithm is 3% higher accuracy than the improved classical algorithm. On the other side, the quantum

algorithm is 6% higher than the classical algorithm. Thus, it can be concluded that both quantum algorithms have better accuracy than their classical opponents with the same approach.



**Fig. 15. Maze images for testing.**



**Fig. 16. Average shortest path accuracy for the algorithms.**

**Table 18. Testing results for 15 maze images for the algorithms.**

Maze	Average Shortest Path Accuracy Percentage (%)			
	Classical	Quantum	Improved Classical	Improved Quantum
Maze 1	100	100	100	100
Maze 2	100	100	100	100
Maze 3	100	100	100	100
Maze 4	100	100	100	100
Maze 5	100	100	100	100
Maze 6	85.71	92.74	85.71	93.1
Maze 7	33.33	51.23	33.33	50.77
Maze 8	0	48.11	100	100
Maze 9	100	100	100	100
Maze 10	100	100	100	100
Maze 11	85.71	78.6	85.71	85.71
Maze 12	71.43	71.43	85.71	85.71
Maze 13	85.71	85.71	85.71	92.86
Maze 14	85.71	85.71	85.71	85.71
Maze 15	42.86	70.86	71.43	78.39
<b>Average</b>	<b>79.3639</b>	<b>85.6259</b>	<b>88.88762</b>	<b>91.48362</b>

#### 4.6.4. Shortest path accuracy test for random maze image

The third test will be the shortest path accuracy test for random path length distance. This test will investigate the effect of the path length values difference between the layer to the shortest path accuracy for the quantum algorithms. The problem statement is “Will the accuracy varies when the difference in the path length values in the layer changes?”. Figure 17 visualises the path length differences. The formula and an example of calculating the path length difference are listed in Eq. (6) below.

$$\text{Path Length Difference in Each Layer} = \frac{\sum_0^N \text{Num}_{\max} - \text{Num}_N}{N-1} \times 100\% \quad (6)$$

Based on Fig. 18, it can conclude that the path length difference is directly proportional to the shortest path accuracy percentage. When the differences in path length values inside each layer increase, the shortest path accuracy percentage increases. This statement can be proved by checking with the gradient of the linear trendline equations. For both Quantum Algorithm and Improved Quantum Algorithm, the gradient of the lines is positive. A positive gradient can be used to prove that the shortest path accuracy percentage is increasing. After knowing that a higher path length difference can produce a higher accuracy, the next step is to determine what random range will output the highest path length value difference in the layer. Theoretically, if the path length's random number range is larger, the path length difference will be larger. The following steps will be identifying whether the theoretical statement is correct or not. Five cases of random path length range as shown in Table 19 had been applied for the testing. For example, Case 1 has a path length range of 1 to 10. Thus, when the random path length generates and assigned into the layers, the path length value inside that layer will only be in the range of 1 to 10, such as [1, 6, 8, 3].

To test for the effect of path length difference, 35000 sets of data have been randomly generated using the random key in mode of number generator function in the system. The system has calculated the 35000 sets of data by the Quantum Algorithm and Improved Quantum Algorithm. The shortest path accuracy percentage of each data by both quantum algorithms has been recorded and plotted in the graph. Then, two linear trendlines of the two quantum algorithms as shown in Table 20 were calculated and displayed to summarise the 35000 sets of plotted data.

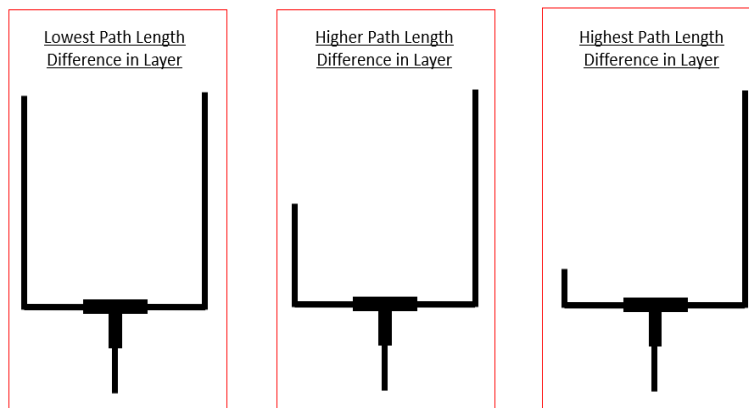
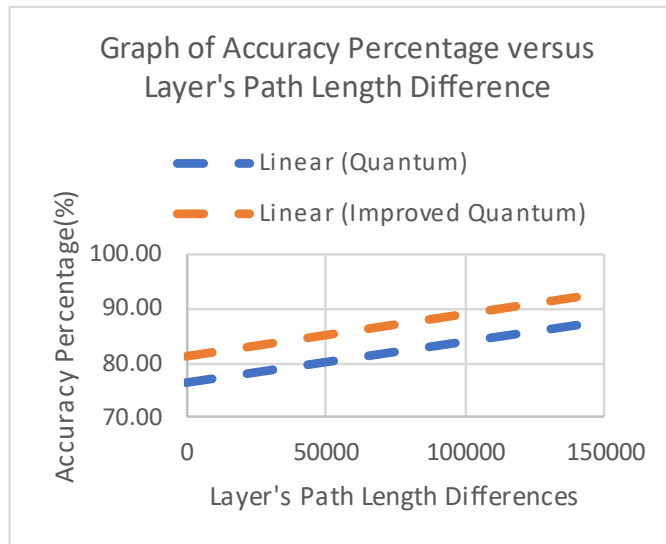


Fig. 17. Example of path length differences.



**Fig. 18. Graph of accuracy percentage versus layer’s path length differences.**

**Table 19. Random path length range for each case.**

Random Path Length Range (Normalised Unit)	
Case 1	1 to 10
Case 2	1 to 100
Case 3	1 to 1000
Case 4	1 to 10000
Case 5	1 to 100000

**Table 20. Linear trendline equation for shortest path accuracy.**

Algorithm	Linear Trendline Equation	Gradient
Quantum	$y = 8 \times 10^{-5} x + 76.294$	+
Improved Quantum	$y = 8 \times 10^{-5} x + 81.187$	+

For the testing, 1000 sets of data have been generated for each case in each number of layers. So, the next step of the testing is to get the average path length difference values inside the number of layers maze. A three layers maze will have three layers inside leads to having three path length difference values. In this case, the average path length difference will be the average from the three path length difference values. The formula for calculating the average path length difference values is listed in Eq. (7).

$$\text{Average path length difference values} = \frac{\sum \text{Path length difference values}}{\text{Number of Layers}} \quad (7)$$

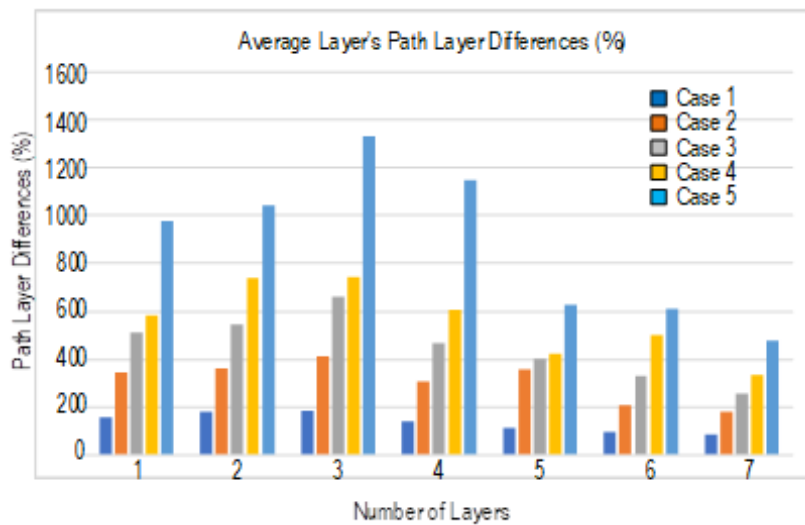
Table 21 shows the average path layer difference in each layer produced by the five different cases. Then, the values for each number of layer maze are plotted into histogram as shown in Fig. 19.

From Fig. 20, the average path layer difference increases from Case 1 to Case 5 for all the multiple layers maze. Therefore, it can be concluded that when the path layer difference in each layer increases, the shortest path accuracy percentage increases. Similarly, if the path length values in the same layer are close, the shortest path accuracy percentage for the quantum algorithms will drop. To prove the statement above, linear trendlines of the lines have been recorded as shown in Table 22.

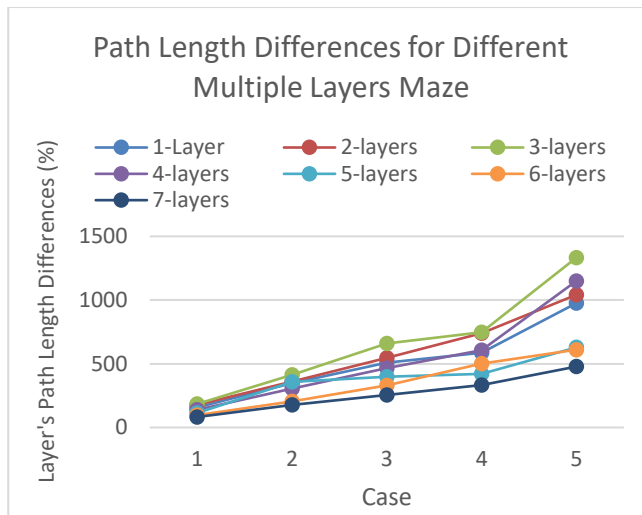
From Fig. 21, all linear trendlines are observed to be increasing when the random range increases. At the same time, the gradient is positive for all linear trendlines. It means that the average path length differences are increasing. In conclusion, when the path length difference in each layer is large for both quantum algorithms, the algorithms have a higher chance of outputting a higher shortest path accuracy. Most importantly, the characteristic of the quantum algorithms has been learnt in this test.

**Table 21. Average path layer difference produced by different cases.**

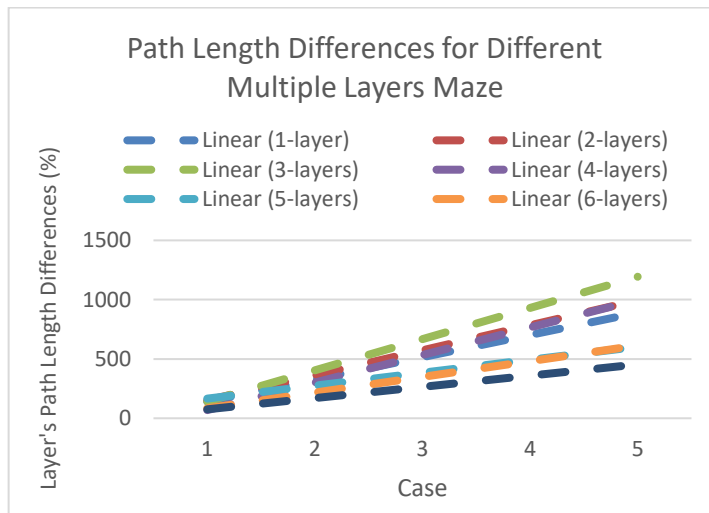
Number of Layers	Average Layer's Path Layer Difference (%)				
	Case 1 (1-10)	Case 2 (1-100)	Case 3 (1-1000)	Case 4 (1-10000)	Case 5 (1-100000)
1	158.166	347.794	508.992	585.72	976.133
2	177.63	360.886	545.838	740.348	1041.784
3	184.77	413.472	658.488	746.714	1333.449
4	139.052	305.6	466.518	605.904	1150.473
5	113.931	358.586	399.273	420.634	628.711
6	96.886	204.657	331.238	501.545	610.545
7	82.57	176.809	254.648	332.282	478.147



**Fig. 19. Average path length differences for different multiple layers maze.**



**Fig. 20. Graph of path length differences for different multiple layers maze.**



**Fig. 21. Trendline graph of path length differences for different multiple layers maze.**

**Table 22. Linear trendline equation for different multiple layers maze.**

Number of Layers	Linear Trendline Equation	Gradient
1	$y = 187.39 x - 46.797$	+
2	$y = 210.78 x - 59.034$	+
3	$y = 263.06 x - 121.8$	+
4	$y = 232.31 x - 163.43$	+
5	$y = 109.16 x + 56.744$	+
6	$y = 132.42 x - 48.287$	+
7	$y = 94.663 x - 19.097$	+

#### 4.6.5. Testing of accuracy effect from different quantum shots values

The fourth test is the shortest path accuracy test from different quantum shot values. The term "quantum shot" is used for the execution of the quantum circuit. The two quantum algorithms have been tested for shortest path accuracy by varying the quantum shot values. It can be imagined as the number of times running the quantum circuit built from the quantum gates. There were 13 different test numbers of quantum shots for the fourth test, which started from 10 to 10000. The setup of the testing is by generating the maze information using the Random Key in Mode. Then, the system will solve the maze by using the two quantum algorithms. Each quantum algorithm will run 13 times by varying the quantum shots value for each time. Thus, there will be a total of 26 times algorithm running for one set of maze information. The generated maze information will start from one layer maze to seven layers maze. Each layer maze will be generated 100 sets of maze data 10 times. Thus, it will contribute to 1000 sets of maze data per multiple layers maze. In the end, it will contribute to a total of 7000 sets of maze data for seven types of multiple layers maze, and the analysis will be based on the 7000 sets of data. Tables 23 and 24 list out the testing results.

Figure 22 was plotted by using the average shortest path accuracy testing results. From Fig. 22, it can be observed that the accuracy percentage is increasing. However, the increment is getting smaller and eventually stabilise around a certain percentage. For example, the Quantum Algorithm stabilised at around 77%, while the Improved Quantum Algorithm stabilised at around 82%. Fidelity (%) is defined as the closeness of the value to the highest value. In this case, the fidelity percentage has been calculated to determine which quantum shot value will reach the saturation point. The saturation point will be the point when the increment is less or insignificant and the point that starts to close to the highest accuracy value. The difference in fidelity value has been calculated to assist in finding the saturation point. If the difference in fidelity is less than 0.5%, the state will be concluded as the saturation state.

$$\text{Fidelity (\%)} = \frac{\text{Highest Accuracy} - \text{Self Accuracy}}{\text{Highest Accuracy}} \times 100\% \quad (8)$$

$$\text{Fidelity Difference (\%)} = 100 - \text{Fidelity (\%)} \times 100\% \quad (9)$$

**Table 23. Results of 13 quantum shot values for quantum algorithm.**

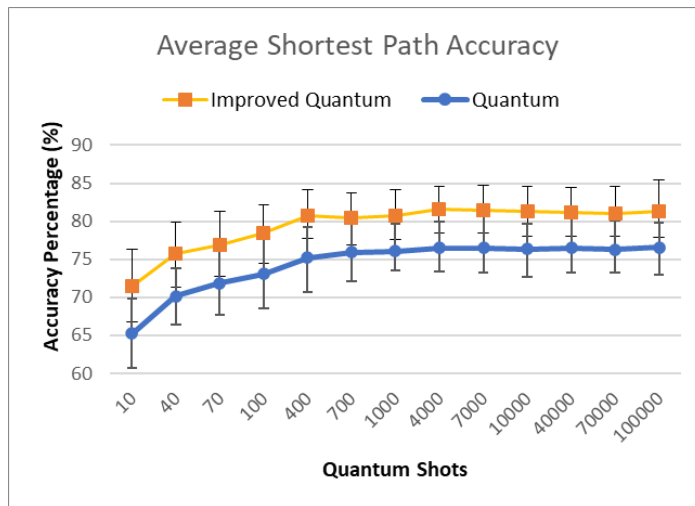
Random Shots	Quantum							Average
	1-L	2-L	3-L	4-L	5-L	6-L	7-L	
10	80.2	72.77	66.34	61.33	60.76	58.16	56.84	<b>65.2</b>
40	87.5	77.17	71.21	67.46	65.46	62.07	60.04	<b>70.13</b>
70	90.6	78.17	73.37	69.45	65.98	64.82	60.58	<b>71.85</b>
100	92.4	79.37	73.79	70.39	68.49	65.51	61.71	<b>73.09</b>
400	95	80.07	75.83	72.15	70.85	67.97	64.51	<b>75.2</b>
700	95.6	80.73	76.57	71.51	71.35	68.45	67.1	<b>75.9</b>
1000	95.5	80.47	77.14	71.86	71.76	68.97	66.63	<b>76.05</b>
4000	95	80.63	77.17	72.31	73	69.86	67.39	<b>76.48</b>
7000	94.5	81.67	76.16	73.01	72.03	69.56	68.33	<b>76.47</b>
10000	94.9	81.07	76.77	72.78	71.94	69.9	66.92	<b>76.33</b>
40000	95.3	81	77.13	72.76	71.32	70.43	67.27	<b>76.46</b>
70000	94.4	80.47	76.46	72.51	72.65	70.03	67.53	<b>76.29</b>
100000	95.6	80.87	77.11	73.41	71.77	69.59	67.75	<b>76.59</b>

**Table 24. Results of 13 quantum shot values for improved quantum.**

Random Shots	Quantum							Average
	1-L	2-L	3-L	4-L	5-L	6-L	7-L	
10	83.4	75.77	72.47	69.47	67.37	64.44	60.42	<b>70.48</b>
40	88.7	81.7	78.3	73.59	70.47	69.17	67.6	<b>75.65</b>
70	89.2	81.77	78.57	75.99	72.83	70.58	67.79	<b>76.68</b>
100	93.9	83.47	80.06	76.29	73.99	68.83	69.65	<b>78.03</b>
400	95.3	84.8	81.26	79.57	75.68	74.5	72.61	<b>80.53</b>
700	94.4	84.87	81.51	78.89	76.51	73.75	74.02	<b>80.56</b>
1000	95.2	84.73	81.89	78.49	76.15	74.45	73.3	<b>80.6</b>
4000	95.7	84.43	82.44	79.92	77.51	75.47	74.61	<b>81.44</b>
7000	95.4	84.9	81.76	79.56	77.65	75.57	75.15	<b>81.43</b>
10000	95.4	84.5	81.46	79.25	77.7	76.09	75.21	<b>81.37</b>
40000	94	84.13	81.5	80.03	77.26	74.46	74.27	<b>80.81</b>
70000	94.7	84.3	81.57	79.64	76.85	75.78	74.84	<b>81.1</b>
100000	95.9	84.47	82.3	79.49	77.64	75.25	74.45	<b>81.36</b>

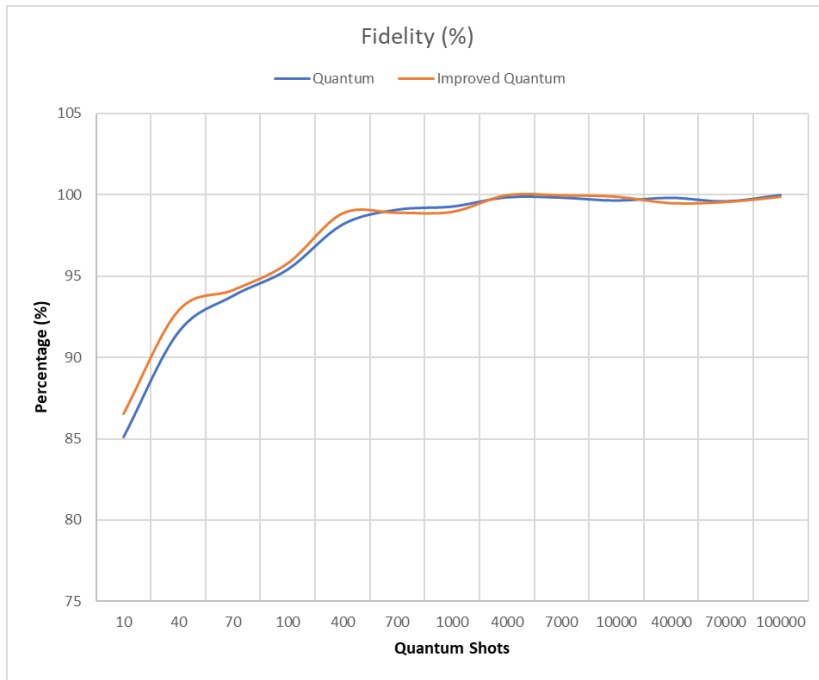
From Figs. 23 and 24, the fidelity difference for both algorithms is less than 0.5% starts from the quantum shot value of 4000. The closeness of the value from the quantum shots value of 4000 onwards is more than 99.5%, as shown in Tables 25 and 26. Therefore, it can be concluded that the accuracy will be saturated after 4000 quantum shots.

By knowing the saturation point, it will benefit in saving time. For example, 4000 quantum shots and 100000 quantum shots can output a similar accuracy by requiring less execution time.

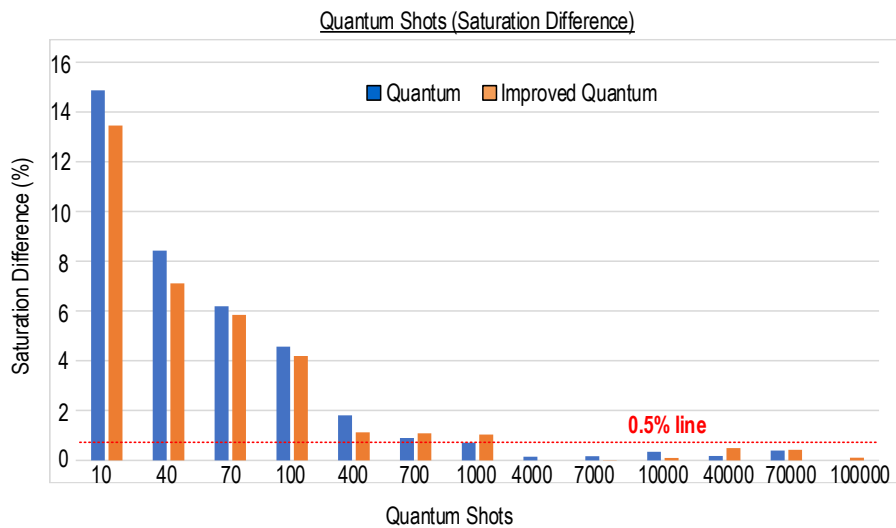


**Fig. 22. Average shortest path accuracy from different quantum shot values.**





**Fig. 23. Fidelity graph of shortest path accuracy.**



**Fig. 24. Fidelity differences of shortest path accuracy.**

As a result of Table 27, by using the quantum shots value of 4000 instead of quantum shots value of 100000, the system can maintain a similar accuracy of less than 0.5% in difference, while the time taken can be significantly saved by around 59%. Based on the fourth test, another characteristic of quantum algorithms has been learnt.

**Table 25. Fidelity of shortest path accuracy.**

Quantum Shots	Fidelity (%)	
	Quantum	Improved Quantum
10	85.13	86.54
40	91.57	92.89
70	93.81	94.16
100	95.43	95.81
400	98.19	98.88
700	99.1	98.92
1000	99.29	98.97
4000	99.86	100
7000	99.84	99.99
10000	99.66	99.91
40000	99.83	99.51
70000	99.61	99.58
100000	100	86.54

**Table 26. Fidelity difference of shortest path accuracy.**

Quantum Shots	Fidelity Difference (%)		
	Quantum	Improved Quantum	Below 0.5%?
10	14.87	13.46	NO
40	8.43	7.11	NO
70	6.19	5.84	NO
100	4.57	4.19	NO
400	1.81	1.12	NO
700	0.9	1.08	NO
1000	0.71	1.03	NO
4000	0.14	0	YES
7000	0.16	0.01	YES
10000	0.34	0.09	YES
40000	0.17	0.49	YES
70000	0.39	0.42	YES
100000	0	0.1	YES

**Table 27. Quantum shots time comparison of 4000 and 100000.**

Quantum Shots	Average Time	Time saving
4000	0.388	$\frac{0.945 - 0.388}{0.945} \times 100\% \approx 59\%$
100000	0.945	

#### 4.6.6. Testing of computation time

The fifth test is the testing of the computation time of the two quantum algorithms. In this test, the quantum algorithms will be compared with the traditional basic method of finding the shortest path. The traditional basic uses the method of adding the path values in each layer simultaneously. The testing setup is by recording the computation time for multiple layers maze up to seven layers maze. The maze information will be the same when recording the computation time for the algorithms. For the testing, 1000 sets of maze samples have been generated for each layer maze. Therefore, a total of 7000 sets of maze samples have been used to investigate the computation time.

Based on Tables 28 and 29 above, it can be observed that the traditional basic method has a lesser computation time compared to the quantum algorithms. However, the computation time for the basic method increases more exponentially after adding more layers. On the other hand, the increment of the quantum algorithms

is comparatively lesser. Eventually, the traditional basic method will catch up with the quantum algorithms method and requires comparable more computation time after that. Assume that the computation time growth is exponentially for all the methods; the exponential equation for both methods can be obtained using the graph trendline function from Microsoft Excel. The exponential equations are listed in Table 30 and plotted in Fig. 25.

The trendline equations as recorded in Table 30 can be used to calculate the intersection point between the equations. The intersection point will be the point that the methods have the same computation time. The calculation will be done in two sections: 'Basic vs Quantum' and 'Basic vs Improved Quantum'.

Table 31 shows the number of layers for the basic classical method to catch up with the quantum algorithms. Therefore, when the number of layers maze reaches 23 layers, the computation time for the basic method will catch up with the two quantum algorithms. For the layers maze after 23 layers, the computation time for the basic method will require more than the quantum algorithms. Thus, it can also be concluded that the quantum algorithms can be faster and win over the current traditional basic method after reaching certain layers maze.

**Table 28. Computation time for different algorithms.**

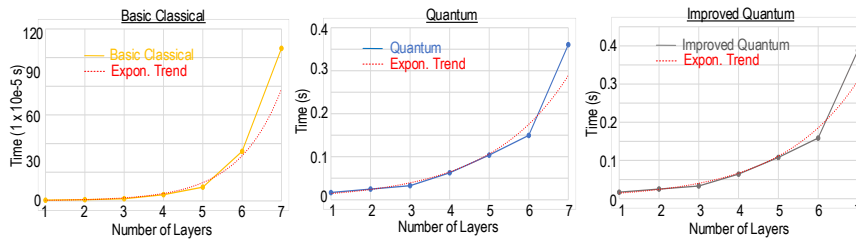
Number of Layers	Computation Time (s)		
	Quantum	Improved Quantum	Traditional Basic
1-L	0.0168	0.0169	$5.0 \times 10^{-6}$
2-L	0.0248	0.0251	$8.506 \times 10^{-6}$
3-L	0.0327	0.0332	$1.503 \times 10^{-5}$
4-L	0.0627	0.0643	$4.318 \times 10^{-5}$
5-L	0.1039	0.1079	$9.593 \times 10^{-5}$
6-L	0.1497	0.1586	$3.456 \times 10^{-4}$
7-L	0.3607	0.3883	$1.065 \times 10^{-3}$

**Table 29. Increment of computation time for different algorithms.**

Number of Layers	Computation Time (s)		
	Quantum	Improved Quantum	Traditional Basic
1-L	-	-	-
2-L	147	149	170
3-L	132	132	177
4-L	192	194	287
5-L	166	168	222
6-L	144	147	361
7-L	241	245	308
<b>Average</b>	<b>170.33</b>	<b>172.5</b>	<b>254.17</b>

**Table 30. Trendline equation for different algorithms.**

Algorithm	Trendline Equation
Basic	$y = 1 \times 10^{-6} e^{0.9053x}$
Quantum	$y = 0.0088 e^{0.4983x}$
Improved Quantum	$y = 0.0087 e^{0.5097x}$



**Fig. 25. Computation time for different algorithms with exponential trendline equation.**

**Table 31. Intersection points result from calculations.**

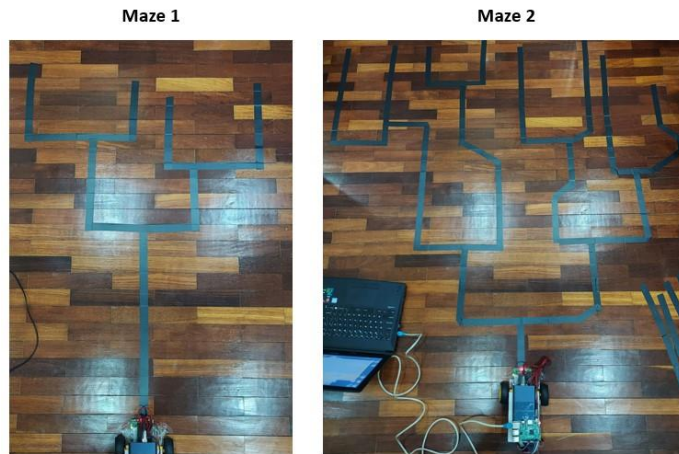
Algorithm	Number of Layers to Overcome
Basic & Quantum	23
Basic & Improved Quantum	23

**4.6.7. Testing of compatibility with prototype for quantum algorithms**

The sixth test is the compatibility test with the prototype. The microprocessor of the prototype will run the two quantum algorithms. This test is set to check whether the quantum algorithms can control the prototype to move in the calculated shortest path route.

For the testing, two actual mazes, as shown in Fig. 26, have been prepared. Each maze will be run 50 times by the ‘Quantum Algorithm’ prototype and 50 times by the ‘Improved Quantum Algorithm’ prototype. Therefore, it will contribute to 100 testing samples for both quantum algorithms. The main task is to compare the calculated path by the algorithm to the final path by the prototype. If the calculated path and the final path are the same, the algorithms are proven to be compatible and usable for the prototype. For Maze 1, 50 times of ‘Quantum Algorithm’ calculated path are ‘10’. Fortunately, the prototype was also moved to the ‘10’ path 50 times. The same result has occurred for the ‘Improved Quantum Algorithm’.

The ‘Improved Quantum Algorithm’ has calculated ‘10’ for 50 times, and the prototype has also moved to the ‘10’ path for the 50 times. For Maze 2, 50 times of ‘Quantum Algorithm’ calculated path are varied from ‘001’, ‘011’, ‘101’ and ‘111’. Fortunately, the final path from the prototype was also moved to the same path 50 times. For example, when the calculated path was ‘001’, the prototype was also moved to the ‘001’. On the other hand, the ‘Improved Quantum Algorithm’ has calculated a constant of ‘111’ for 50 times, and the prototype has also moved to the ‘111’ path for the 50 times. Therefore, 100 samples for ‘Quantum Algorithm’ and 100 samples for ‘Improved Quantum Algorithm’ has proven that the quantum algorithms can perfectly control the prototype. The prototype can understand the instruction from the quantum algorithms and move to the desired path in the actual maze. It has provided proof to show that quantum algorithms are capable of controlling the robots, similar to what the classical algorithms can do.



**Fig. 26. Actual maze of maze 1 and maze 2 for testing.**

#### **4.6.8. Summary**

In summary, six tests have been applied to quantum algorithms. The first outcome from the test is that the quantum algorithms have a higher shortest path accuracy than their classical version of the same approach for all kinds of inputs. Compared between the two quantum algorithms, the improved version wins over its original version at around 10%. The second outcome from the test is that the path length differences in the layer will affect the shortest path accuracy. If the path length differences are higher, the quantum algorithms can produce a higher shortest path accuracy. The third outcome from the test is that the quantum shot value for the quantum algorithms can be set as 4000. The Quantum shot value of 4000 is tested to be the starting point of the saturation point of the highest accuracy while requiring a lesser computation time comparing to the other values in saturation state. The fourth outcome from the test is that the quantum algorithms can win over the traditional basic classical way of solving the maze after 23 layers. The fifth outcome from the test is that the quantum algorithm results are compatible with the robot. Besides the testing, other information such as time management and costing has also been discussed.

#### **5. Conclusion and Recommendation**

In conclusion, two quantum algorithms have been successfully developed in solving the shortest path of the maze. The algorithms were designed using the ratio-based approach. Both algorithms can solve the maze and output the shortest path result. A system was additionally made to execute the algorithms smoothly. The robot was built by using the Raspberry Pi as the microprocessor. The quantum robot has been constructed by implementing or programming the quantum algorithms into the robot. The quantum algorithms were compatible with the robot as the robot can use the output result from the algorithms to move and solve the maze. Lastly, six tests had been performed on the system, including the navigation accuracy test and computation time test. The testing results were evaluated and analysed well. The navigation accuracy for the quantum algorithms was proven to outperform their classical version opponent. The navigation accuracy range for both quantum algorithms is falling on around 78% to 84%. On the other hand, the computation

time has been recorded. Both algorithms will outperform the traditional basic maze solving method after reaching a certain number of layers.

### Acknowledgment

The authors thank technical assistants at School of Engineering, APU for their technical support and acknowledge the infrastructure support provided by IBM.

### References

1. Robison, K. (2021). Here's how quantum computing could transform the future. Retrieved December 2, 2022, from <https://www.businessinsider.com/quantum-computing-investing-computers-enterprise-2021-3>.
2. Bashuk, M.A. (2003). Solving a maze with a quantum computer. Retrieved December 2, 2022, from <https://arxiv.org/abs/quant-ph/0304146>.
3. Pronin, C.; Ostroukh, A.; Pronin, B.; Vasiliev, Y.; and Kotliarskiy, E. (2019). Development of a quantum algorithm based on quantum parallelism for finding the shortest path in a graph. *APRN Journal of Engineering and Applied Sciences*, 14(4), 848-851.
4. Ray, P. (2014). Quantum simulation of Dijkstra's algorithm. *International Journal of Advance Research in Computer Science and Management Studies*.
5. Caruso, F.; Crespi, A.; Ciriolo, A.G.; Sciarrino, F.; and Osellame, R. (2016). Fast escape of a quantum walker from an integrated photonic maze. *Nature Communications*, 7.
6. Reitzner, D.; Hillery, M.; and Koch, D. (2017). Finding paths with quantum walks or quantum walking through a maze. *Physical Review A*, 96.
7. Kumar, N.; and Goswami, D. (2013). Quantum algorithm to solve a maze: converting a maze problem into search problem. Retrieved December 2, 2022, from <https://arxiv.org/abs/1312.4116>.
8. Krauss, T.; and Mccollum, J. (2020). Solving the network shortest path problem on a quantum annealer. *IEEE Transactions on Quantum Engineering*, 1, 1-12.
9. Janlahmann. (2021). RasQberry. Retrieved December 2, 2022, from <https://github.com/JanLahmann/RasQberry>.
10. LaRose, R. (2019). Overview and comparison of gate level quantum software platforms. *Quantum*, 3.
11. Carrascal, G.; Del Barrio, A.A.; and Botella, G. (2020). First experiences of teaching quantum computing. *The Journal of Supercomputing*, 77, 2770-2799.
12. Roy, P.K. (2020). Quantum logic gates. Retrieved December 2, 2022, from [https://www.researchgate.net/publication/343833536\\_Quantum\\_Logic\\_Gates](https://www.researchgate.net/publication/343833536_Quantum_Logic_Gates).
13. Shaik, E.; and Rangaswamy, N. (2020). Implementation of quantum gates based logic circuits using IBM Qiskit. *Proceedings of the 5th International Conference on Computing, Communication and Security (ICCCS)*. Patna, India, 1-6.
14. Singh, P.N.; and Aarthi, S. (2021). Quantum circuits—an application in Qiskit-python. *Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. Tirunelveli, India, 661-667.