

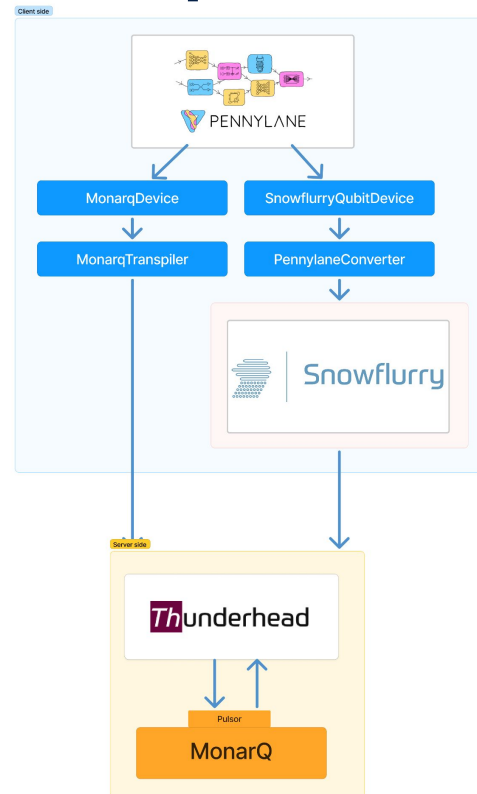


# Chapitre 3 : Comment programmer un ordinateur quantique?

# Kits de développement logiciel quantique

- **PennyLane**
  - Développé en Python par Xanadu
  - Interface directement avec MonarQ
- **Snowflurry**
  - Développé en Julia par Anyon Systems
  - Interface directement avec MonarQ
- **Qiskit**
  - Développé en Python par IBM
  - Plugin Qiskit-calculquebec en développement

Écrire des circuits et beaucoup plus !

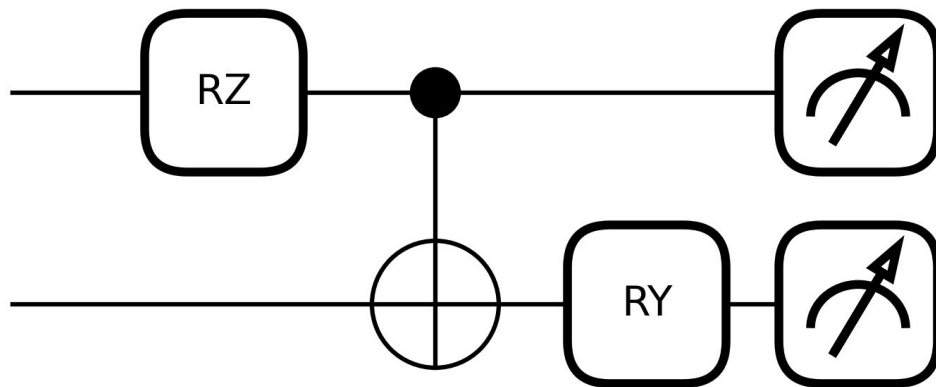


# Circuits quantiques

```
dev = qml.device('default.qubit', wires = 2)
@qml.qnode(dev) # qnode decorator
def quantum_function(x, y):
    qml.RZ(x, wires=0)
    qml.CNOT(wires=[0,1])
    qml.RY(y, wires=1)
    return qml.state()
```



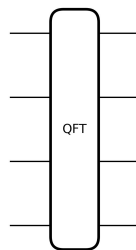
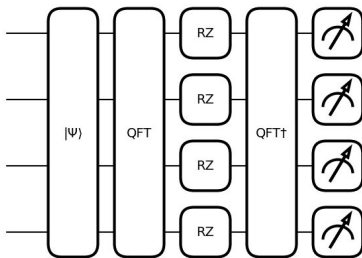
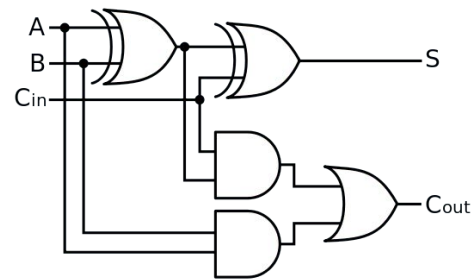
PENNYLANE



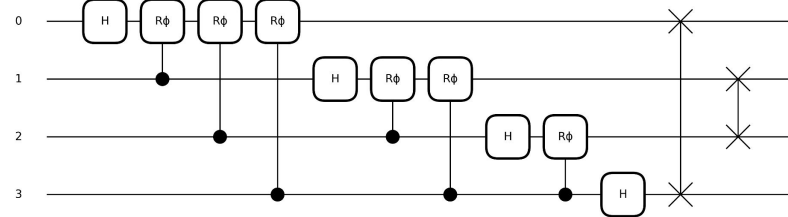
# Circuits quantiques

Les ordinateurs classiques utilisent également des circuits, mais nous n'y pensons généralement pas de manière explicite.

- Circuit booléen **classique** pour l'addition
- Circuit **quantique** pour l'addition

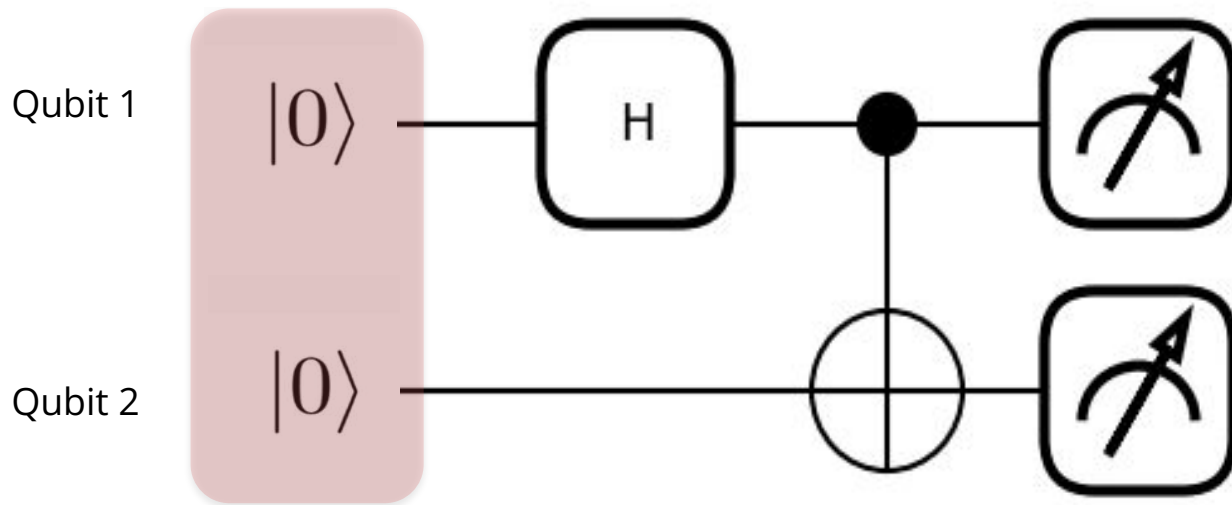


=

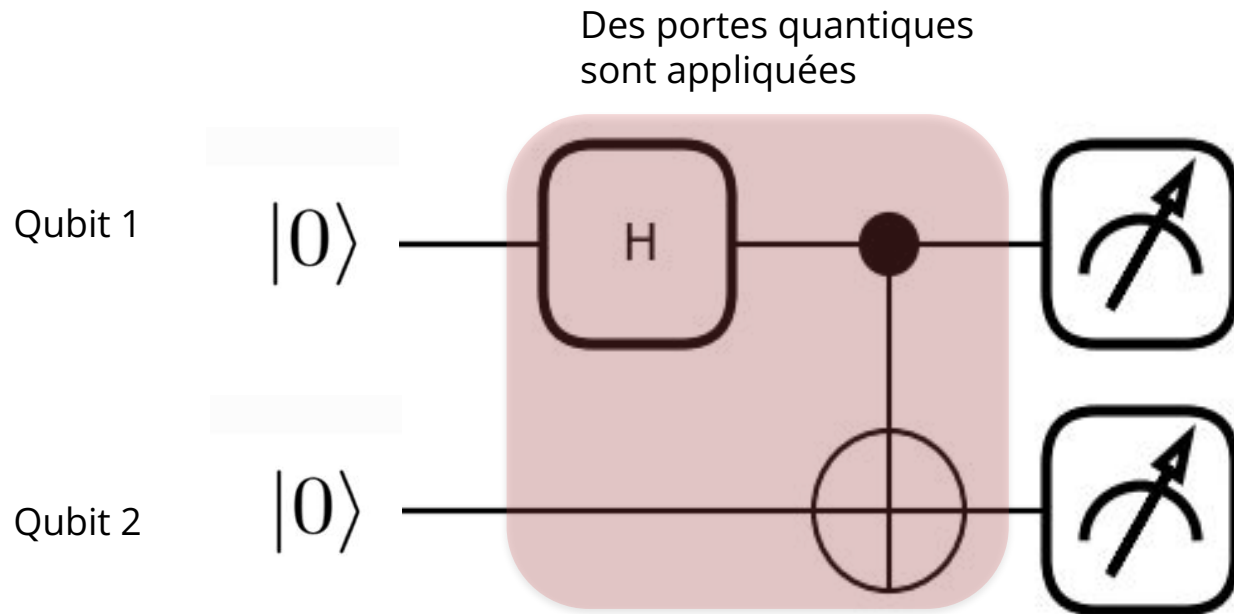


# Éléments clés d'un circuit quantique

Chaque qubit est initialisé à l'état 0 par défaut

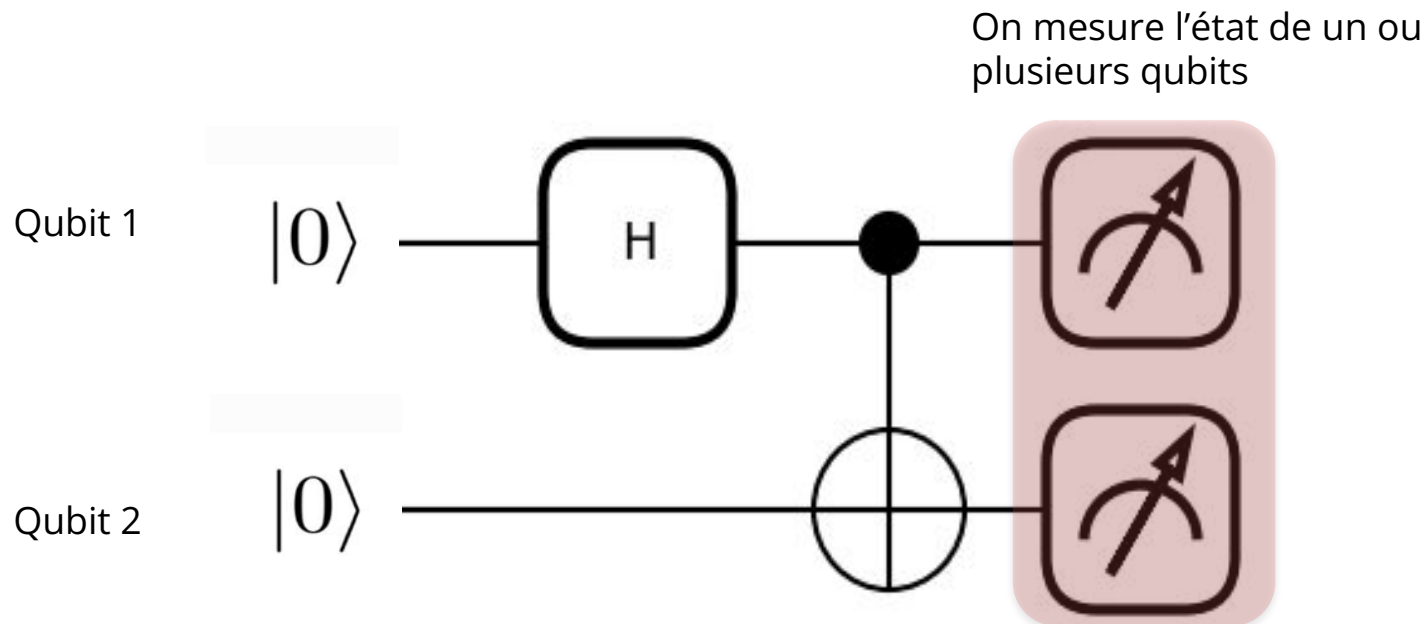


# Éléments clés d'un circuit quantique





# Éléments clés d'un circuit quantique



**Pause  
programmation**



**Notebook 2 :  
Les bases de PennyLane**








# Notebook 2 : Aide-mémoire



PENNYLANE 

## 1. Un circuit quantique est composé de portes

Gate	Circuit Element	Matrix Representation	Action on Basis States
Hadamard Gate $H$		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$H 0\rangle = \frac{1}{\sqrt{2}}( 0\rangle +  1\rangle)$ $H 1\rangle = \frac{1}{\sqrt{2}}( 0\rangle -  1\rangle)$
Pauli-X Gate $X$		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$X 0\rangle =  1\rangle$ $X 1\rangle =  0\rangle$
Pauli-Y Gate $Y$		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$Y 0\rangle = i 1\rangle$ $Y 1\rangle = -i 0\rangle$
Pauli-Z Gate $Z$		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$Z 0\rangle =  0\rangle$ $Z 1\rangle = - 1\rangle$
CNOT Gate		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$CNOT 00\rangle =  00\rangle$ $CNOT 01\rangle =  01\rangle$ $CNOT 10\rangle =  11\rangle$ $CNOT 11\rangle =  10\rangle$

## 2. Un circuit quantique doit retourner une mesure (`qml.state`, `qml.expval`, `qml.probs`, `qml.counts`)

## 3. Circuit et 'device' sont liés avec un 'Qnode'

**Qnode**

**Quantum function**

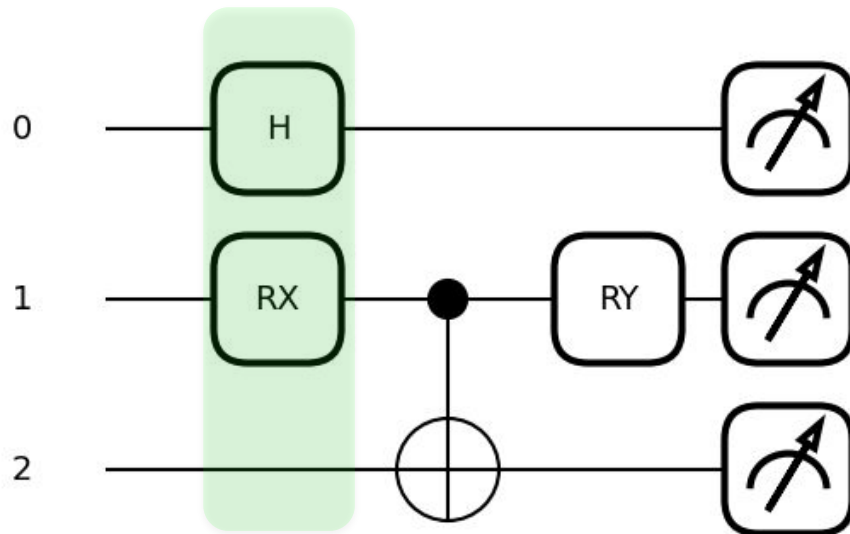
```
def circuit():
    ...
    return qml.counts(0)
```

**Device**

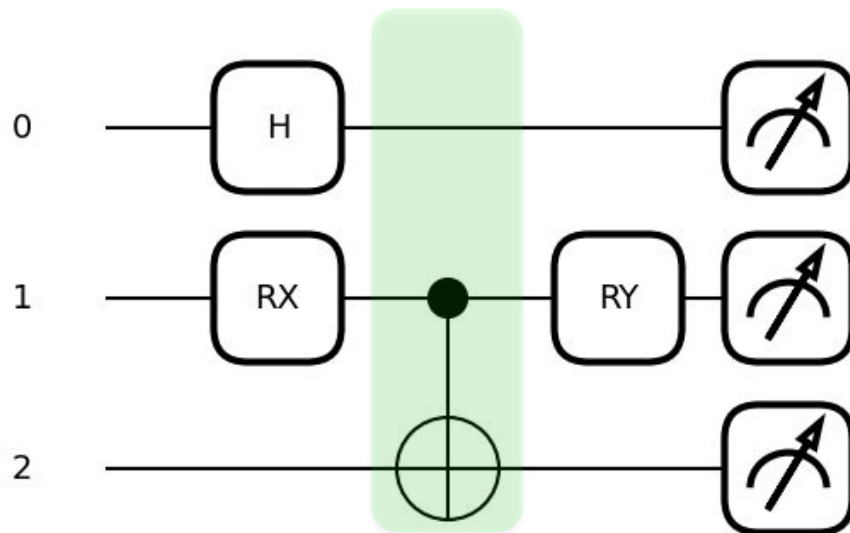
`qml.device(...)`



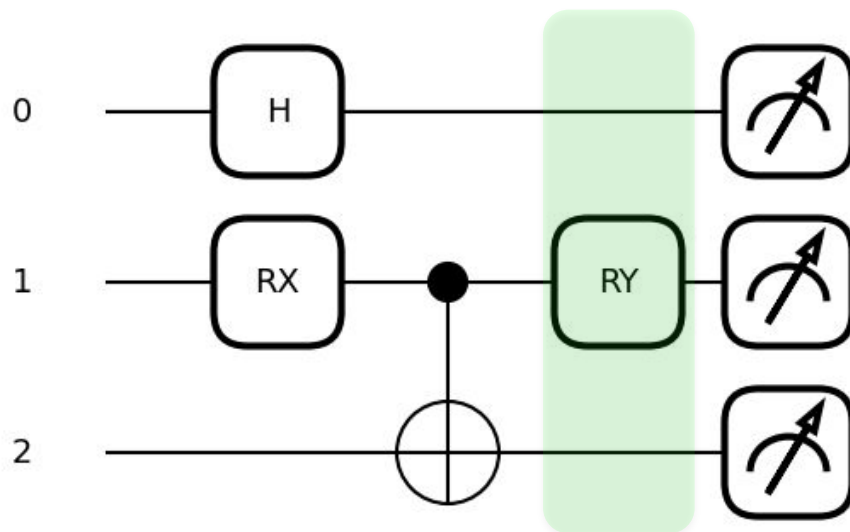
# Profondeur de circuit



# Profondeur de circuit

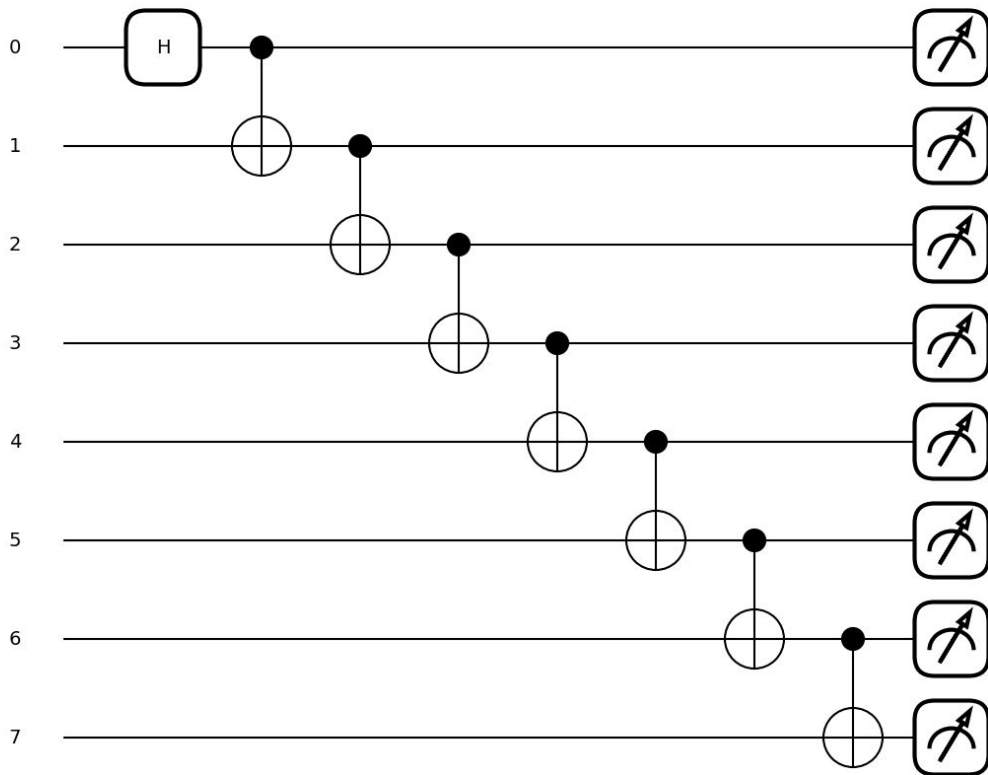


# Profondeur de circuit



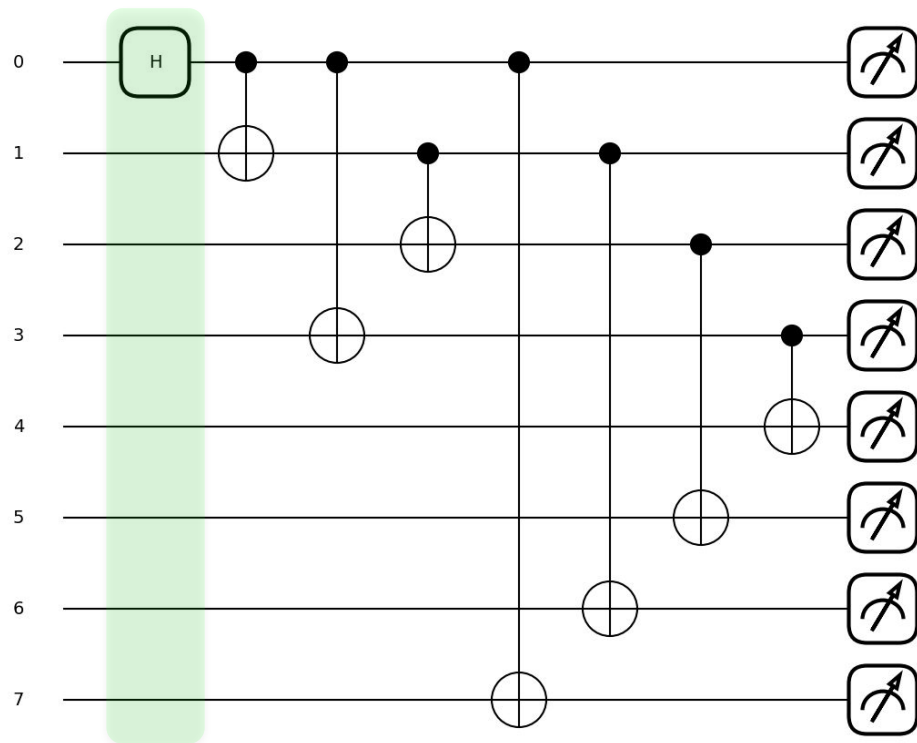
La profondeur du circuit est 3.

# Profondeur de circuit

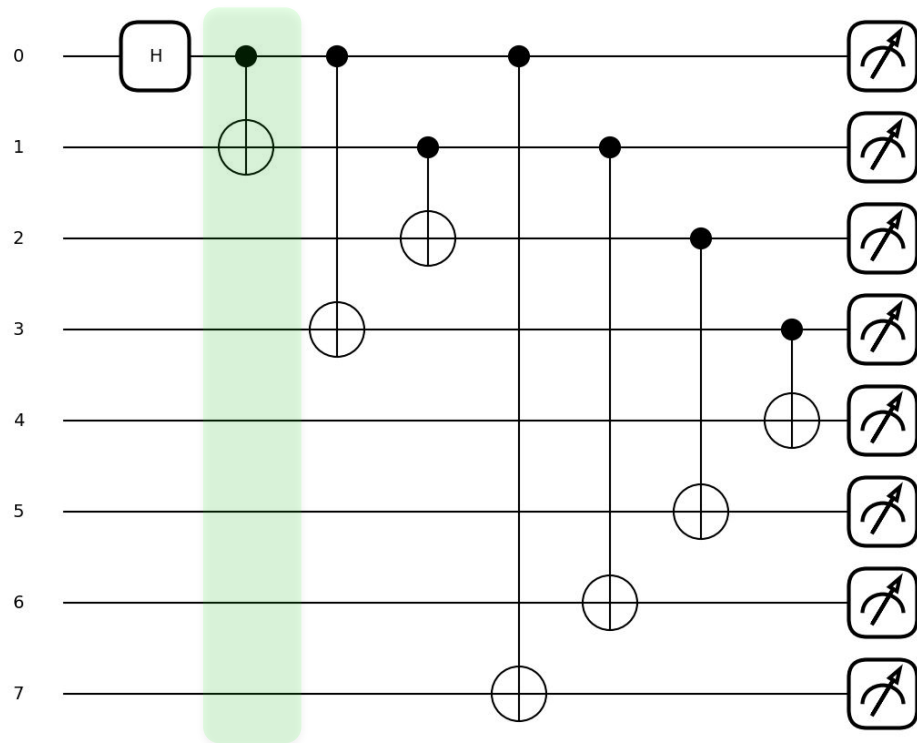


Quelle est la profondeur de ce circuit ?

# Profondeur de circuit

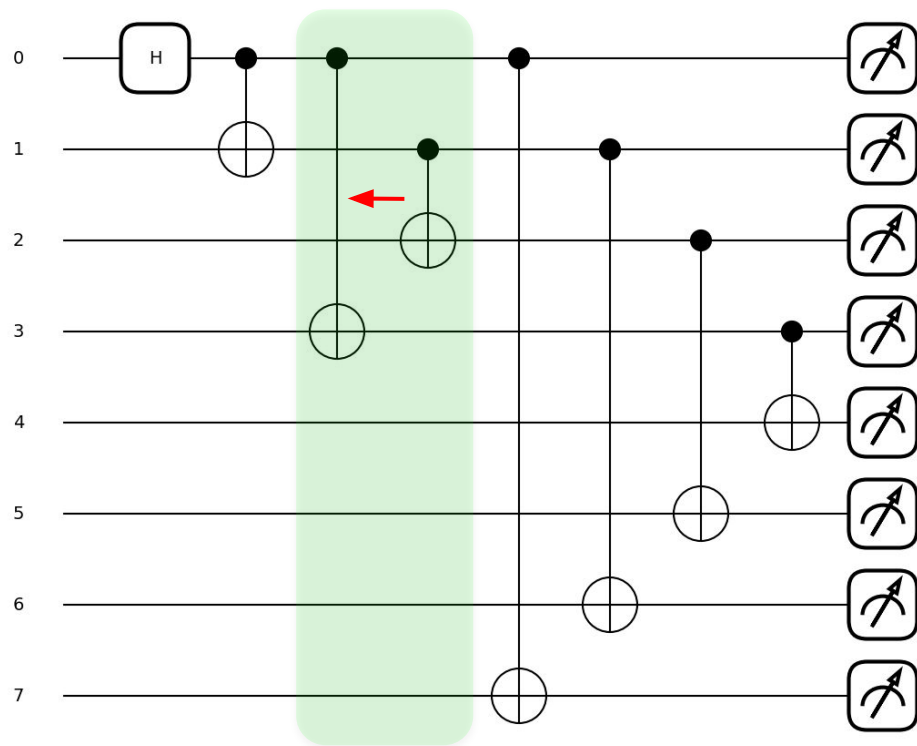


# Profondeur de circuit

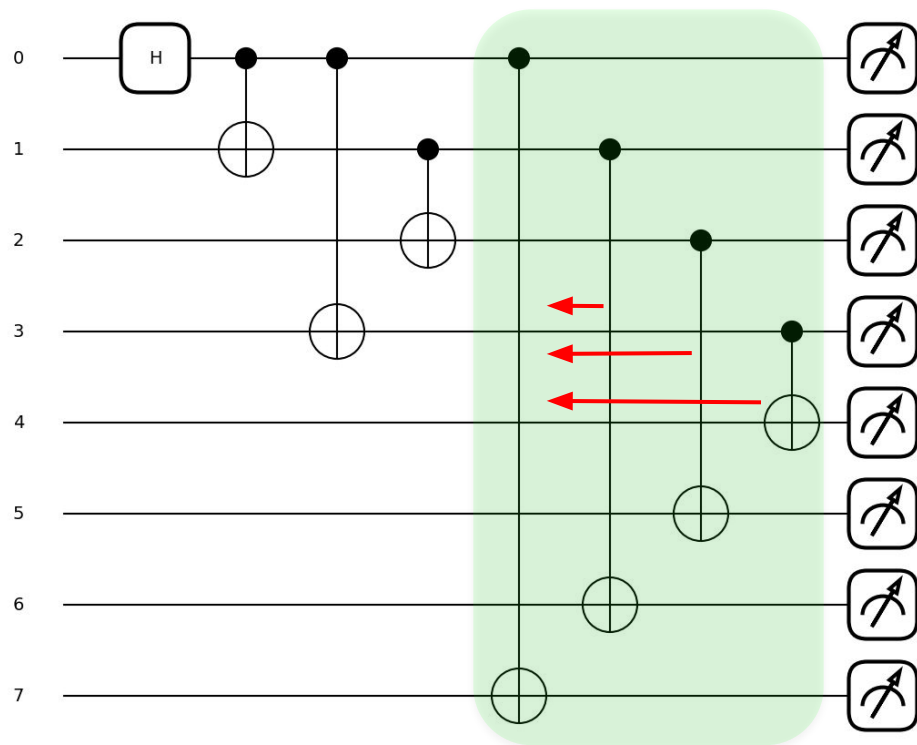




# Profondeur de circuit



# Profondeur de circuit



La profondeur du circuit est 4.

## Plan pour l'après-midi

- Exécuter des circuits sur MonarQ
  - MonarQ, transpilation et mitigation d'erreurs
  - Notebook 3 : État GHZ sur MonarQ
- Pause
- Calcul hybride
  - Circuits variationnels et apprentissage machine quantique
  - Notebook 4 : Circuits variationnels
- Questions