Test Examen IOT

Exercicie 1 :

1) La vision de Mark Weiser sur l'informatique ubiquitaire se concrétise grâce à l'Internet des objets (IoT). Les principales raisons sont les avancées technologiques qui ont miniaturisé les capteurs, rendant les composants moins chers et plus puissants, et une connectivité omniprésente (Wi-Fi, 5G). De plus, les progrès en intelligence artificielle permettent de traiter les données en temps réel, tandis que la consommation énergétique réduite des appareils facilite leur déploiement. L'interopérabilité entre plateformes IoT et la demande croissante pour des solutions connectées accélèrent également cette adoption, intégrant la technologie de façon invisible dans la vie quotidienne.

2) Le réseau wif pour LAN et LoRaWAN pour LPWAN

3) a,c

Exercice 2 :

1.

a) Exactly Once (QoS 2)

- Signification : Ce niveau garantit que chaque message est reçu une seule fois, ni plus, ni moins(pas des doublons de message)
- Niveau de qualité : C'est le niveau le plus élevé de QoS. Il est plus sûr et fiable, mais également plus lent, car il nécessite plusieurs échanges de confirmation entre l'émetteur et le récepteur pour assurer que le message n'est livré qu'une fois.

b) At Least Once (QoS 1)

- Signification : Avec ce niveau, le message est assuré d'être reçu au moins une fois par le récepteur. Cependant, il est possible que le message soit reçu plus d'une fois si le récepteur ne parvient pas à envoyer l'accusé de réception dans un délai donné.
- Niveau de qualité : Ce niveau offre un bon compromis entre fiabilité et efficacité, car il assure la livraison, mais peut entraîner des doublons dans certains cas.

c) At Most Once (QoS 0)

- Signification : Ce niveau n'assure pas la livraison du message. Le message est envoyé au plus une fois, ce qui signifie qu'il peut ne pas être reçu du tout en cas de perte de paquet.

- Niveau de qualité : C'est le niveau le plus bas de QoS, adapté aux cas où la rapidité est prioritaire et où la perte d'un message n’a pas de conséquences critiques (comme des données de capteurs transmises fréquemment).

2.

	MQTT	CoAP	Web Socket
Type de protocole	Messagerie	Transfert reseau	Transfert web
Architecture	Publication / Abonnements	Client/Serveur	Client/serveur
Synchronisme	Asynchrone	Asynchrone	Synchrone

Etudes de cas :

Question 1 :

	Porté	debit	Consommation d’energie
Bluetooth	Courte (10 à 100 mètres)	Modéré (jusqu'à 3 Mbps)	Faible à modérée
4G	Longue (plusieurs kilomètres)	Élevée	Élevée
LoRaWAN	Très longue (jusqu'à 15-20 km en zones rurales)	Très faible	Très faible

Question 2 :

Oui, un smartphone peut être considéré comme une passerelle (gateway) car Les smartphones disposent de multiples interfaces de communication, comme Wi-Fi, Bluetooth, 4G/5G, et parfois NFC. Ils peuvent donc facilement se connecter à des capteurs ou des dispositifs IoT qui utilisent des protocoles variés, tels que Bluetooth ou Zigbee, et retransmettre ces données via Internet.

Question 3 :

Le **publisher** peut supprimer un message localement une fois qu'il a reçu l'accusé de réception approprié du broker, pour le niveau **QoS 1 (At Least Once)** : Le message est conservé par le publisher jusqu’à ce qu’un accusé de

réception PUBACK soit reçu du broker. Après cela, le message peut être supprimé localement car il a été reçu au moins une fois par le broker.

Question 4 :

```
public void Choc_Alert() {
    if (is_cancelled()) {
        return;
    }

    CountDownTimer countDownTimer = new CountDownTimer(30000, 1000) {
        @Override
        public void onFinish() {
            if (!is_cancelled()) {
                publishCrashAlert();
            }
        }
    }.start();
}

private void publishCrashAlert() {
    try {

        MqttMessage crashMqttMessage = new MqttMessage();
        crashMqttMessage.setPayload( get_ID().getBytes());
        crashMqttMessage.setQos(2);
        mqttAndroidClient.publish("crash/id_cyc", crashMqttMessage);
        addToHistory("Crash alert message published on crash/id_cyc: " + get_ID());


        MqttMessage locationMqttMessage = new MqttMessage();
        locationMqttMessage.setPayload(get_location().getBytes());
        locationMqttMessage.setQos(2);
        mqttAndroidClient.publish("crash/location_cyc", locationMqttMessage);
        addToHistory("Location alert message published: " + get_location());

    } catch (MqttException e) {
        System.err.println("Error Publishing: " + e.getMessage());
        e.printStackTrace();
    }
}

```