Détection de Chocs & Surveillance Environnementale

Introduction :

Dans le domaine biomédical et industriel, la surveillance des conditions environnementales et la détection des chocs jouent un rôle crucial pour garantir la sécurité et l'intégrité des produits sensibles, tels que les médicaments, les vaccins et les équipements médicaux. Des chocs ou des variations environnementales non contrôlés peuvent compromettre la qualité et l'efficacité de ces produits, entraînant des pertes économiques et des risques pour les patients.

La technologie IoT (Internet of Things) offre des solutions innovantes pour répondre à ces enjeux grâce à des capteurs intelligents comme l'accéléromètre MPU6050.

Dans cette présentation, nous explorerons une application IoT dédiée à la **détection de chocs** et à la **surveillance environnementale** dans le contexte du transport pharmaceutique et de l'industrie biomédicale. Nous verrons l'IoT, permet de suivre en temps réel les paramètres critiques pour assurer la sécurité des produits et des patients ainsi optimiser la logistique dans le domaine biomédicale.

SLIDE 4.

Publisher : Représenté par le réfrigérateur sur le schéma. Le rôle du publisher est d'envoyer des messages vers un topic spécifique.

Exemple : Un capteur dans le réfrigérateur pourrait publier la température sur le topic home/kitchen/fridge/temperature.

MQTT Broker : Son rôle est de recevoir les messages publiés par les publishers et de les distribuer aux subscribers inscrits au même topic.

Subscribers : Représentés ici par des appareils tels que des téléphones, des ordinateurs, ou des serveurs. Les subscribers s'abonnent à des topics pour recevoir les messages qui les intéressent.

SLIDE 5.

HTTP (HYPERTEXT TRANSFER PROTOCOL)

Étape 1 : L'utilisateur saisit une URL dans le navigateur.

Étape 2 : Traduction en requête HTTP Le navigateur transforme l'URL en une requête HTTP et l'envoie au serveur web.Type de requête : Par exemple, une requête de type GET est utilisée pour demander la ressource index.html.

Étape 3 : Le serveur web répond (HTTP Response) Le serveur reçoit et interprète la requête, récupère la ressource demandée depuis sa base de données ou son système defichiers.

Étape 4 : Affichage de la ressource Le navigateur reçoit la réponse HTTP et traite les données pour les afficher à l'utilisateur sous forme d'une page web.

Rôle : Traduire les données reçues (HTML, CSS, JavaScript) en une interface visuelle que l'utilisateur peut consulter.

Partie 1 : Détection de choc à l’aide d’un accéléromètre

SLIDE 7.

Objectifs principaux du projet :

Détection de chocs : Utiliser un accéléromètre pour mesurer les variations brusques d'accélération et identifier un choc ou un impact.

Signalisation visuelle : Allumer une LED lorsqu'un choc est détecté pour fournir un retour visuel immédiat.

Communication des données : Exploiter la connectivité Wi-Fi de l'ESP32 pour transmettre les données de choc vers un smartphone, une application ou un serveur distant.

Fiabilité et précision : Assurer une détection fiable des chocs tout en minimisant les fausses alertes dues à des vibrations ou mouvements normaux.

SLIDE 8.

Hardware

Accéléromètre MPU6050 : Le MPU6050 est un capteur avancé qui combine un accéléromètre 3 axes et un gyroscope 3 axes dans un seul module. Conçu par InvenSense, ce capteur est largement utilisé dans des applications nécessitant une mesure précise des mouvements et de l'orientation.

Caractéristiques principales :

1. Accéléromètre 3 axes :
   - Mesure l'accélération linéaire sur les axes X, Y et Z.
   - Permet de détecter les chocs, les vibrations et l'inclinaison.
   - Plage de mesure configurable : ±2g, ±4g, ±8g, ±16g.

2. Gyroscope 3 axes :
   - Mesure la vitesse angulaire (rotation) autour des axes X, Y et Z.
   - Plage de mesure configurable : ±250, ±500, ±1000, ±2000 degrés par seconde.

3. Fonctionnalités supplémentaires :
   - Intègre un capteur de température pour surveiller les variations thermiques.
   - Inclut un processeur de mouvement numérique (DMP) capable de fusionner les données du gyroscope et de l'accéléromètre pour fournir des mesures précises sans bruit.

Applications dans le contexte biomédical :

- **Transport pharmaceutique :** Détection des chocs pouvant endommager les produits sensibles.

- **Équipements médicaux :** Surveillance des vibrations pour éviter des dysfonctionnements.
- **Prothèses intelligentes :** Mesure des mouvements pour adapter les réglages dynamiquement.

!! Grâce à sa précision et sa flexibilité, le MPU6050 est un choix populaire pour développer des systèmes IoT fiables, capables de surveiller les conditions environnementales et mécaniques en temps réel.

SLIDE 9.

WOKWI

Nous avons utilisé Wokwi, une plateforme en ligne de simulation de circuits électroniques, pour concevoir, simuler et tester notre projet IoT sans matériel physique. Compatible avec l'ESP32, elle offre une interface intuitive pour programmer en C/C++ ou MicroPython.

SLIDE 10.

Arduino Cloud

Arduino IoT Cloud est une plateforme PaaS qui permet de connecter facilement des dispositifs Arduino à Internet pour la gestion et le contrôle des projets IoT. Elle offre des outils pour créer des dashboards interactifs et le contrôle des appareils à distance. Avec des services intégrés tels que le stockage de données, la gestion des appareils et la communication via MQTT, **Arduino IoT Cloud** facilite l'interaction avec Wokwi pour récupérer les données en temps réel.

SLIDE 11.

Communication Arduino Cloud & Wokwi **(VIDEO)**

Étape N°0 : Création d'un compte sur Arduino IoT Cloud.

Après avoir créé votre compte sur Arduino IoT Cloud, accédez à l'interface de développement en cliquant sur "Get Started".

Étape N°1 : Accéder à l'interface principale.

Une fois sur la plateforme, accédez à l'interface principale en cliquant sur "Home".

Cliquez sur "Create New Thing" pour créer un nouveau projet IoT (appelé "Thing").

Donnez un nom à votre projet et définissez le type de carte que vous utiliserez dans l'onglet "Associated Device".

Vous obtiendrez alors un **Device ID** et une **Secret Key**, qui seront utilisés dans votre code pour assurer la communication entre l'ESP32 et le cloud. Ces informations doivent être conservées en toute sécurité.

Arduino IoT Cloud génère automatiquement un identifiant MQTT.

Étape N°2 : Définir les variables dans Arduino IoT Cloud.

Ajoutez les variables nécessaires à votre projet. Ces variables détermineront les paramètres essentiels de votre projet.

Pour chaque variable, vous devez définir :

Nom : Le nom de la variable.

Type : Le type de donnée (par exemple, int, float, bool, etc.).

Permissions : Définissez si la variable est en lecture, en écriture ou les deux (lecture/écriture).

Déclaration : Cela sera automatiquement généré dans un fichier appelé thingProperties.h.

Dans notre projet nous aaurons besoins de Quatres variables

1. **Magnitude :** pour la mesure de la vitesse dans les trois axes de l'accéléromètre et la comparer a un seuil afin de déterminer le choc qui sert à mesurer la force ou la vitesse combinée sur les trois axes.

2. **LED (Variable booléenne) :** Qui permet de **visualiser physiquement** qu'un choc a été détecté.

3. **ShockDetected (Variable booléenne) :** Permet d'envoyer des alertes ou de synchroniser les données avec le cloud lorsque qu'un choc est détecté. Utile pour les systèmes de monitoring distants ou la création de journaux d'événements.

4. **Temperature (Variable float) :** Complémentaire à la détection de choc, elle permet d'évaluer si le choc est lié à des conditions extrêmes (comme une surchauffe).Fournit des informations supplémentaires pour des applications comme le transport pharmaceutique.

Dans la même interface, configurez les paramètres réseau pour votre appareil.

Étape N°4 : Générer le code et obtenir les informations nécessaires.

Un code généré contient automatiquement les fichiers nécessaires pour établir la connexion entre votre projet IoT et le cloud. Parmi ces fichiers, on trouve :

Le code généré inclut trois fichiers essentiels :

- **thingProperties.h** : Définit les propriétés (variables) du projet IoT.
- **arduino_secrets.h** : Contient les informations de sécurité et de configuration réseau (Device ID et Secret Key).

- **Fichier .ino** : Gère la logique principale, la communication Wi-Fi, la collecte des données des capteurs, leur publication sur le cloud, et les réponses aux événements comme les alertes de choc.

Étape N°5 : Importer les fichiers dans WOKWI.

Une fois que vous avez généré le code, vous pouvez importer les fichiers dans **Wokwi** pour simuler votre projet et tester la communication entre l'ESP32 et Arduino IoT Cloud.

SLIDE 12.

Dashboard Arduino Cloud. (Video)

Le dashboard sous Arduino Cloud permet de créer une interface dynamique et interactive pour surveiller et contrôler votre système IoT. Vous pouvez y ajouter des widgets en fonction des besoins de votre projet, chacun étant associé à une variable définie dans le code.

Dans notre cas :

Un widget "Value" : Pour afficher la valeur en temps réel de la magnitude calculée par l'accéléromètre.

Un widget "LED" : Pour simuler visuellement un choc détecté en s'allumant lorsqu'un seuil prédéfini est dépassé.

Une alerte : ainsi une courbe pour le suivie en temps réelle et avoir du DATA.

SLIDE 13/14/15.

Partie Code

#define SHOCK_THRESHOLD 2.0 : Définit la magnitude minimale (en g) qu'une accélération doit atteindre pour être considérée comme un choc. Elle correspond à la norme vectorielle des accélérations sur les axes X, Y et Z.

#define SHOCK_DURATION 500 : Spécifie un intervalle minimal de 500 ms entre deux détections de chocs pour éviter les détections répétées dues à un même événement ou à des vibrations continues.

#define MOVING_AVG_SIZE 8 : Définit la taille de la fenêtre de moyenne mobile utilisée pour lisser les données d'accélération, réduisant les variations rapides et le bruit pour des mesures plus stables.

Lecture des données du capteur : Les données brutes du MPU6050 sont converties en "g" (accélération gravitationnelle) en divisant par 16384.0, donnant des valeurs entre -2.0 et +2.0 g pour chaque axe X, Y, et Z.

SLIDE 16.

Simulation 1.

L'objectif de cette simulation est de valider la capacité du système IoT basé sur le MPU6050 à détecter les chocs au-delà d'un seuil critique et à déclencher une alerte en temps réel. Cette fonctionnalité est essentielle pour garantir la sécurité des produits sensibles dans les environnements biomédicaux, notamment pendant le transport pharmaceutique.

Déroulement de la simulation :

1. Configuration initiale :

- Le seuil de détection de choc est défini à **2g** sur l'accéléromètre MPU6050.
- Le capteur est intégré à une plateforme IoT (par exemple, un ESP32 connecté à une interface utilisateur comme Blynk ou un tableau de bord personnalisé).

2. Événement simulé :

- Une variation brusque de l'accélération, simulant un choc ou une vibration importante, est générée.
- Cette variation dépasse le seuil fixé de 2g.

3. Réaction du système :

- Lorsque l'accélération dépasse 2g sur l'un des axes (X, Y ou Z), le système interprète cela comme un choc significatif.
- Une alerte est immédiatement générée via la plateforme IoT. Cette alerte peut inclure :
    - Une notification en temps réel sur une application mobile.
    - L'enregistrement des données sur un serveur pour analyse ultérieure.
    - Un indicateur visuel (LED) ou sonore (buzzer) pour une réponse locale.

Résultat et analyse :

1. Observation : Après le dépassement du seuil de 2g, une alerte a été générée avec succès.

Le système a montré une fiabilité dans la détection des chocs et une réactivité dans la transmission de l'alerte.

L'objectif principal, qui était de détecter et signaler les chocs critiques, a été atteint, validant l'efficacité du système pour des applications biomédicales sensibles.

2. Représentation graphique :

- Inclure un graphique des données d'accélération (en g) sur les axes X, Y, et Z au cours du temps. Mettre en évidence :
    - La valeur de seuil (2g).

- Le moment où l'alerte a été déclenchée (avec un marqueur ou une flèche).

- Un aperçu de l'alerte (exemple : capture d'écran de l'application montrant la notification).

3. **Domaine d'application :**

Ce test démontre l'utilité du système dans des scénarios tels que :

- **Transport de vaccins et médicaments sensibles,** où les chocs doivent être minimisés pour préserver l'efficacité des produits.
- **Surveillance des équipements biomédicaux en transit,** pour détecter tout événement pouvant affecter leur fonctionnement.

Partie 2 : Serveur Web basé sur ESP32 Surveillance Environnementale

SLIDE 18.

Objectif.

Développer un système IoT intégrant le contrôle d'une LED ainsi que l'acquisition des données de température et d'humidité, accessibles via une interface web.

Utiliser un ESP32 pour exécuter un web serveur accessible à partir d'un réseau Wi-Fi local.

SLIDE 19.

L'ESP32 agit comme un serveur web. Il est connecté au PC via un port USB, ce qui permet deux choses : d'abord de l'alimenter en énergie et ensuite de le programmer ou de récupérer des données.

Esp32 se connecte à un réseau Wi-Fi, via un routeur. Grâce à cette connexion, il devient accessible à d'autres appareils sur le même réseau local

Le PC agit comme un client HTTP en utilisant un navigateur pour accéder à l'ESP via son adresse IP. Le serveur web sur l'ESP reçoit les requêtes, comme allumer ou éteindre une LED, et les exécute en conséquence.

Conclusion :

En conclusion, ce projet a montré l'intégration de technologies pour la détection de chocs et la surveillance environnementale.

La comparaison des protocoles a révélé que HTTP, simple mais gourmand, est moins adapté aux besoins IoT en temps réel, tandis que MQTT, plus léger, convient mieux aux communications rapides et fiables. Le choix du protocole dépend donc des besoins spécifiques de l'application.

Dans ce projet, nous avons démontré la mise en œuvre d'un système IoT fiable pour la détection de chocs et la surveillance environnementale dans le domaine biomédical. En intégrant l'accéléromètre MPU6050 à une plateforme IoT basée sur Arduino IoT Cloud et simulée avec Wokwi, nous avons exploité deux protocoles de communication :

1. Protocole MQTT pour la détection de chocs, permettant une transmission efficace des alertes en temps réel.
2. Protocole HTTP pour la surveillance des conditions environnementales, offrant une gestion fiable des données via une interface conviviale.

Résultats et Comparaison :

- Le protocole MQTT a prouvé son efficacité dans les applications nécessitant des notifications rapides et des échanges légers, comme la gestion des alertes critiques en cas de dépassement du seuil de 2g.
- En parallèle, le protocole HTTP s'est avéré adapté pour des tâches moins sensibles en termes de latence, telles que la surveillance continue des conditions environnementales.

Objectif atteint :

L'objectif de comparer ces deux approches a été pleinement satisfait. Nous avons pu mettre en évidence les avantages spécifiques de chaque protocole dans leur domaine d'application respectif, offrant ainsi une solution optimisée et polyvalente pour les besoins du transport et de la surveillance biomédicale.

ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

Présenté par :

Himedi Makrame

Encadré par :

Pr. ZAZ GHITA

Année Universitaire : 2024 – 2025