1.1 Analyse et architecture du système

1. Décrire avec vos mots à quoi sert le mot-clé volatile et quand l'utiliser en programmation embarquée ?

2. Quel outil pouvez-vous utiliser pour protéger une variable partagée entre deux

threads ?

mutex (`k_mutex`) pour l'exclusion mutuelle, et sémaphore (`k_sem`) pour la synchronisation/accès compté.

3. Lister les différentes tâches du système

- ➔ **Tâche acquisition PPG** (lecture MAX30102 à 100 Hz via interruption/FIFO).
- ➔ **Tâche traitement HR/SpO_2**
- ➔ **Tâche communication Wi-Fi** (envoi périodique toutes les 10 s + alerte).
- ➔ **Tâche contrôle/commande** (START/STOP, réglage LED).

4. Décrivez brièvement le fonctionnement du système avec un diagramme haut niveau des différentes tâches.

Capteur MAX30102 -> Interruption DATA_READY -> trigger_handler
-> Lecture IR/RED -> processing_push_sample
-> Buffer PPG SEGMENT_SAMPLES
-> Sémaphore ppg_seg_ready
-> processing_task -> FFT_processing
-> f0 -> HR = 60 * f0
-> Affichage sur console

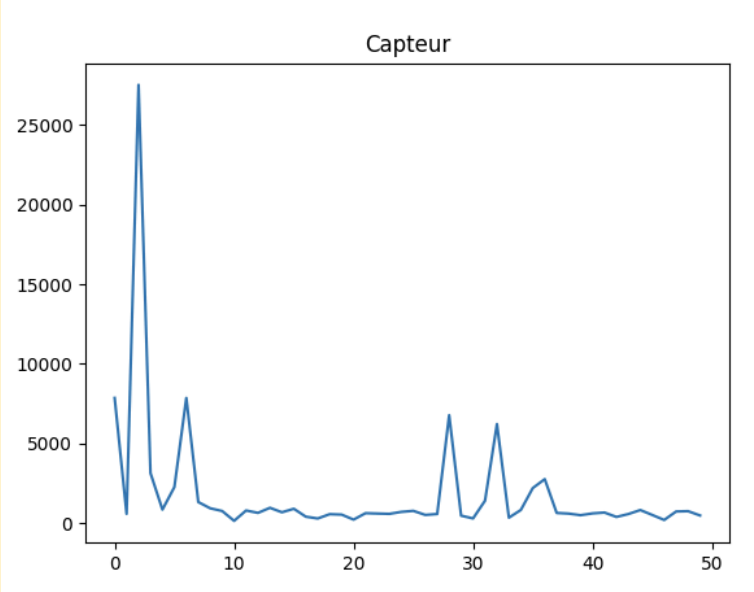
1.2 Capteur MAX30102

5. Lister les pins du MAX30102 ainsi que leur fonction à partir de la datasheet. Lister ensuite les pins sur la carte capteur à votre disposition. Quelle est la tension d'alimentation du capteur ? Que faut-il prévoir pour le pin INT ?

- ➔ VDD : alimentation logique 1.7–2.0 V.
- ➔ VLED+ : alimentation LED 3.1–5.0 V (driver LED).
- ➔ GND, PGND : masses (logique / puissance LED).
- ➔ SCL, SDA : bus I^2C (jusqu'à 400 kHz).
- ➔ INT : sortie open-drain active bas, nécessite pull-up (→ typiquement 4.7 kΩ).

Le FIFO interne stocke 32 échantillons, sur la carte capteur INT sur IO9 de l'ESP32. Ne pas alimenter sur 5 V côté VIN rester en 3.3 V.

10. En utilisant la fonction printk affichez un signal PPG rouge et infrarouge à partir de la console. Réalisez un script python permettant de visualiser le signal (insérer le graphe dans le rapport). Vérifier que cela corresponde bien à un signal PPG. Déterminez manuellement la fréquence du signal.



11/12. Implémentez votre stratégie et vérifiez son bon fonctionnement.

L'acquisition est réalisée dans la routine d'interruption du pilote capteur via trigger.

- ➔ **Acquisition**
  - ◆ trigger_handler() est appelé à chaque DATA_READY capteur.
  - ◆ On lit IR/RED et on pousse dans le buffer processing_push_sample().
- ➔ **Synchronisation**
  - ◆ Quand 1024 points sont acquis, processing_push_sample() fait k_sem_give(&ppg_seg_ready).
- ➔ **Thread**
  - ◆ processing_task() fait k_sem_take(K_FOREVER), lance fft_processing(seg), récupère f0, calcule HR = 60*f0 et l'affiche.

```
Hello World
HR = 52.7 bpm / f0 = 0.88 Hz
```