Documentation EmetteurAQOR

Présentation du projet

EmetteurAQOR est un système IoT de surveillance de la qualité de l'air et de l'environnement qui :

- Mesure les particules fines (PM2.5 et PM10) avec un capteur SDS011
- Capture les données environnementales (température, humidité, pression, COV) avec un capteur BME680
- Enregistre les niveaux sonores avec un microphone INMP441
- Surveille le niveau de batterie
- Transmet toutes ces données via LoRa
- Optimise la consommation d'énergie avec des cycles de veille profonde

Structure du code

Le projet est organisé en plusieurs fichiers pour améliorer la lisibilité et la maintenance :

```
EmetteurAQOR/
├── EmetteurAQOR.ino     # Programme principal
├── config.h             # Configuration globale et constantes
├── lora_manager.h       # Gestion des communications LoRa
├── power_manager.h      # Gestion de l'alimentation et du sommeil
├── SDS011Sensor.h       # Gestionnaire du capteur de particules fines
├── BME680Manager.h      # Gestionnaire du capteur environnemental
├── INMP441Manager.h     # Gestionnaire du microphone
└── BatteryMonitor.h     # Gestionnaire de batterie
```

Fonctionnement général

Le système fonctionne en deux phases distinctes :

Phase 1 : Mesure du niveau sonore (20 premiers cycles)

- Le système se réveille
- Mesure le niveau sonore
- Calcule et stocke les moyennes et maximums
- Retourne en sommeil profond pendant `TIME_TO_SLEEP` secondes
- Répète ce cycle 20 fois

Phase 2 : Collecte et transmission complète des données

- Mesure les particules fines (PM2.5 et PM10) avec le SDS011
- Collecte les données environnementales avec le BME680 (température, humidité, pression, COV)
- Utilise les données sonores accumulées lors de la phase 1
- Vérifie le niveau de batterie
- Transmet toutes les données via LoRa
- Réinitialise le compteur de cycles
- Retourne en sommeil profond

Format des données transmises

Les données sont transmises dans un format CSV avec les valeurs suivantes :

```
PM2.5, PM10, Température, Humidité, Pression, Gas Resistance, IAQ, IAQ Accuracy, dB Moyen, dB Max, Compteur dB>60, % Batterie, Tension Batterie
```

Exemple : `12.5,25.3,22.1,45.5,1013.2,12000.0,45.0,3.0,42.5,68.3,5,85,3.78`

Configuration

Paramètres principaux (config.h)

- `TIME_TO_SLEEP` : Durée du sommeil entre deux mesures (secondes)
- `TIME_TO_MEASURE_PM` : Durée entre deux mesures de particules (secondes)
- `SOUND_MEASUREMENT_CYCLES` : Nombre de cycles pour mesurer le son (défaut : 20)

Configuration LoRa (config.h)

- `RF_FREQUENCY` : Fréquence radio (défaut : 868 MHz)
- `TX_OUTPUT_POWER` : Puissance de transmission (défaut : 18 dBm)
- `LORA_BANDWIDTH` : Bande passante (défaut : 125 kHz)
- `LORA_SPREADING_FACTOR` : Facteur d'étalement (défaut : SF7)

Configuration des broches (config.h)

- `RX_PIN` et `TX_PIN` : Connexion série pour le SDS011 (défaut : 5 et 4)
- `POWER_PIN` : Broche pour alimenter le capteur SDS011 (défaut : 21)
- `Vext` : Broche pour l'alimentation externe (défaut : 43)

Personnalisation

Modifier la durée du cycle de veille

Pour changer l'intervalle entre les mesures, modifiez `TIME_TO_SLEEP` dans `config.h`.

Modifier le nombre de cycles de mesure sonore

Pour ajuster combien de cycles sont consacrés aux mesures sonores, modifiez `SOUND_MEASUREMENT_CYCLES` dans `config.h`.

Ajuster les paramètres LoRa

Si vous rencontrez des problèmes de transmission ou si vous souhaitez optimiser la portée/consommation d'énergie, modifiez les paramètres LoRa dans `config.h` :

- Augmentez `LORA_SPREADING_FACTOR` pour une meilleure portée (au détriment du débit)
- Modifiez `TX_OUTPUT_POWER` pour ajuster la puissance d'émission

Calibration de la batterie

Vous pouvez calibrer la mesure de batterie dans la fonction `setup()` du fichier principal :

```cpp
battery.setMaxVoltage(3.9);      // Tension maximale
battery.setMinVoltage(3.3);      // Tension minimale
battery.setDividerRatio(2.745);  // Ajustement du diviseur de tension
```

Dépannage

Le système ne se réveille pas correctement

- Vérifiez que `gpio_deep_sleep_hold_dis()` est correctement appelé après le réveil
- Assurez-vous que les broches sont correctement configurées pour le sommeil profond

Les données LoRa ne sont pas transmises

- Vérifiez que l'initialisation du module LoRa est correcte
- Confirmez que les paramètres LoRa correspondent à votre récepteur
- Assurez-vous que la fonction `Radio.IrqProcess()` est appelée

Erreurs de lecture des capteurs

- Vérifiez les connexions physiques
- Assurez-vous que l'alimentation des capteurs est suffisante
- Consultez les messages d'erreur série pendant le débogage

Extension du projet

Voici quelques idées pour étendre ce projet :

1. **Ajout de nouveaux capteurs** :
    - Capteur de CO2
    - Capteur de lumière
    - Détecteur de mouvement PIR
2. **Améliorations du logiciel** :
    - Mise en mémoire flash des données en cas d'échec de transmission
    - Mise à jour OTA (Over-The-Air)
    - Interface de configuration WiFi
3. **Optimisation énergétique** :
    - Alimentation solaire avec gestion de charge
    - Adaptation dynamique des cycles de mesure selon le niveau de batterie