I2HM - DOCUMENTATION TECHNIQUE

Utilisation de la devboard RAK3272S

Auteur
Dorian Poissonnet - dorian.poissonnet@gmail.com

Table des matières

1 Introduction **3**

2 Utilisation de la devboard RAK3272S **3**
2.1 Prise en main . . . 3
2.2 Mise à jour du firmware . . . 3
2.3 Configuration de la carte . . . 4
2.4 Connexion des capteurs . . . 4

3 Compression des données **5**
3.1 Compression des données du BME680 . . . 6
3.1.1 Température . . . 6
3.1.2 Humidité . . . 6
3.1.3 Pression . . . 6
3.2 Compression des données de l'accéléromètre . . . 6
3.2.1 Accélération . . . 7

4 Transmission des données **7**
4.1 Code de détection . . . 7
4.2 Trame de données . . . 8

5 Création d'un décodeur de données **8**

6 Conclusion **10**

Table des figures

1 RAK3272S Breakout Board Pinout . . . 3
2 Vue globale du système . . . 5
3 Données décodées depuis ChirpStack . . . 10
4 Données décodées depuis Grafana . . . 10

1 Introduction

Ce rapport expose les résultats obtenus après huit semaines de travail dans le cadre du projet de fin d'année sur le projet I2HM. Il détaille le progrès réalisé et les résultats obtenus en ce qui concerne l'utilisation de la devboard RAK3272S, équipée de la carte RAK3172L. La carte opérant à une fréquence de 433 MHz, une passerelle 433 MHz a été employée pour assurer la liaison entre le réseau 433 MHz et le réseau LoRa 868 MHz.

2 Utilisation de la devboard RAK3272S

2.1 Prise en main

La devboard RAK3272S [1] est extrêmement simplifiée et n'implémente pas l'ensemble des fonctionnalités disponibles de la carte RAK3172. Sa schématisation est disponible sur la figure 1.

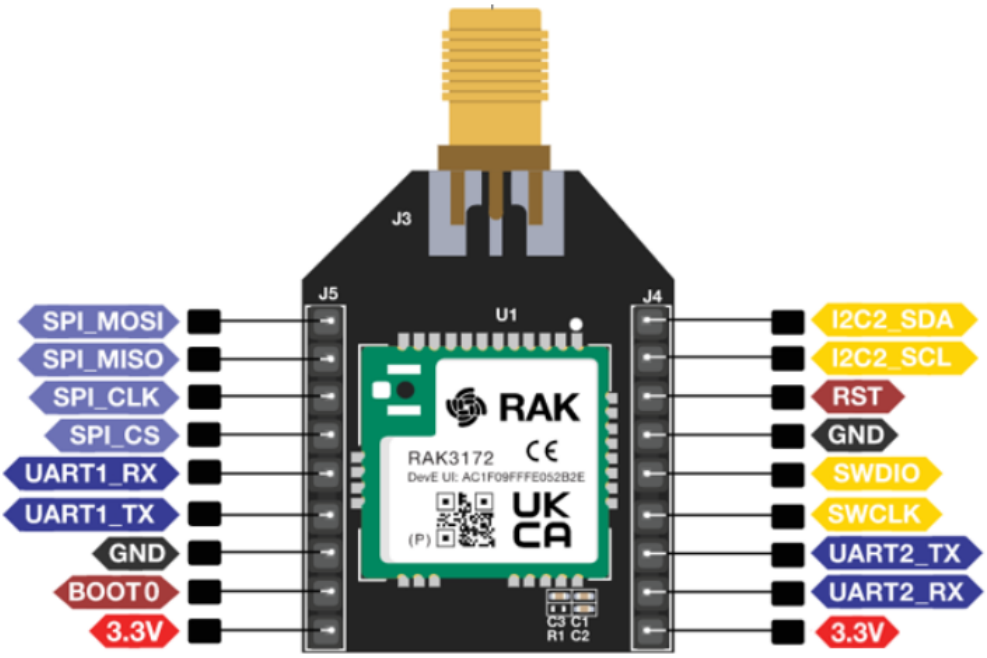

FIGURE 1 – RAK3272S Breakout Board Pinout

Les fonctionnalités disponibles comprennent le SPI, l'I2C, un UART supplémentaire, ainsi que le BOOT et le SWD. Aucune broche analogique n'est disponible sur cette devboard, ce qui a considérablement ralenti l'implémentation des capteurs analogiques. Pour cette première version, seuls les capteurs I2C ont été intégrés : le BME680 [2] et le MMA8451 [3].

2.2 Mise à jour du firmware

Avant de pouvoir utiliser pleinement la devboard, la première étape consiste à mettre à jour le firmware si celui-ci n'a jamais été utilisé. Voici les étapes à suivre :

— **Installer le logiciel STM32CubeProgrammer** [4] : ce logiciel vous permettra d'accéder à la mémoire flash de votre carte RAK3172.
— **Télécharger la dernière version du firmware au format .bin** : la dernière version est accessible depuis la datasheet [5].
— **Connecter le pin BOOT0 à 3.3V** : cette étape est indispensable lors du processus de flashage de votre carte RAK.
— **Utiliser le SWD de la carte avec un adaptateur FTDI (ou une carte STM)** : flasher votre carte RAK.
— **Mettre à jour le firmawre** : Accéder à la mémoire flash en utlisant le SWD et flasher le firmware .bin sur la carte.

Cette série d'étapes permettra de mettre à jour le firmware de votre devboard RAK3272S et de vous assurer qu'elle fonctionne correctement. Vous pouvez désormais reconnecter BOOT0 à la masse afin d'accéder au port série et utiliser les commandes AT.

Si l'opération s'est bien déroulé, vous devriez pouvoir utiliser la commande AT+VER=? et observer la mise à jour de votre firmware : AT+VER=RUI_4.1.0_RAK3172-E OK.

2.3 Configuration de la carte

Avant d'utiliser la carte, plusieurs paramètres doivent être configurés en amont. Parmi ces derniers, nous pouvons citer les éléments suivants : APPKEY, APPEUI, DEVEUI, DR, BAND. La liste complète des commandes AT est disponible sur le site officiel de RAKwireless [6]. Une commande est disponible pour la configuration de chaque paramètre. Voici la liste des configurations à faire.

— **AT+NWM = 1** - Utilisation du mode LoRaWAN.
— **AT+NJM = 0 ou 1** - Utilisation du mode réseau APB : 0 ou OTAA : 1.
— **AT+APPKEY = <your_appkey>** - Disponible sur le serveur hébergeant votre application (ChirpStack).
— **AT+APPEUI = <your_appeui>** - Obsolète sur ChirpStack (utiliser DEVEUI).
— **AT+DEVEUI = <your_deveui>** - Disponible sur la carte RAK3172.
— **AT+DR = <your_dr>** - Utiliser la puissance désirée.
— **AT+BAND = <your_band>** - RAK3172L : 0 (EU433) ou RAK3172H : 4 (EU868).

Votre carte est désormais correctement configurée et prête à être utilisée pour l'émission de données.

2.4 Connexion des capteurs

Avant le début du projet, un plan global avait été élaboré pour la connexion des différents capteurs ainsi que pour le fonctionnement de l'environnement. Vous pouvez consulter ce plan sur la figure 2.

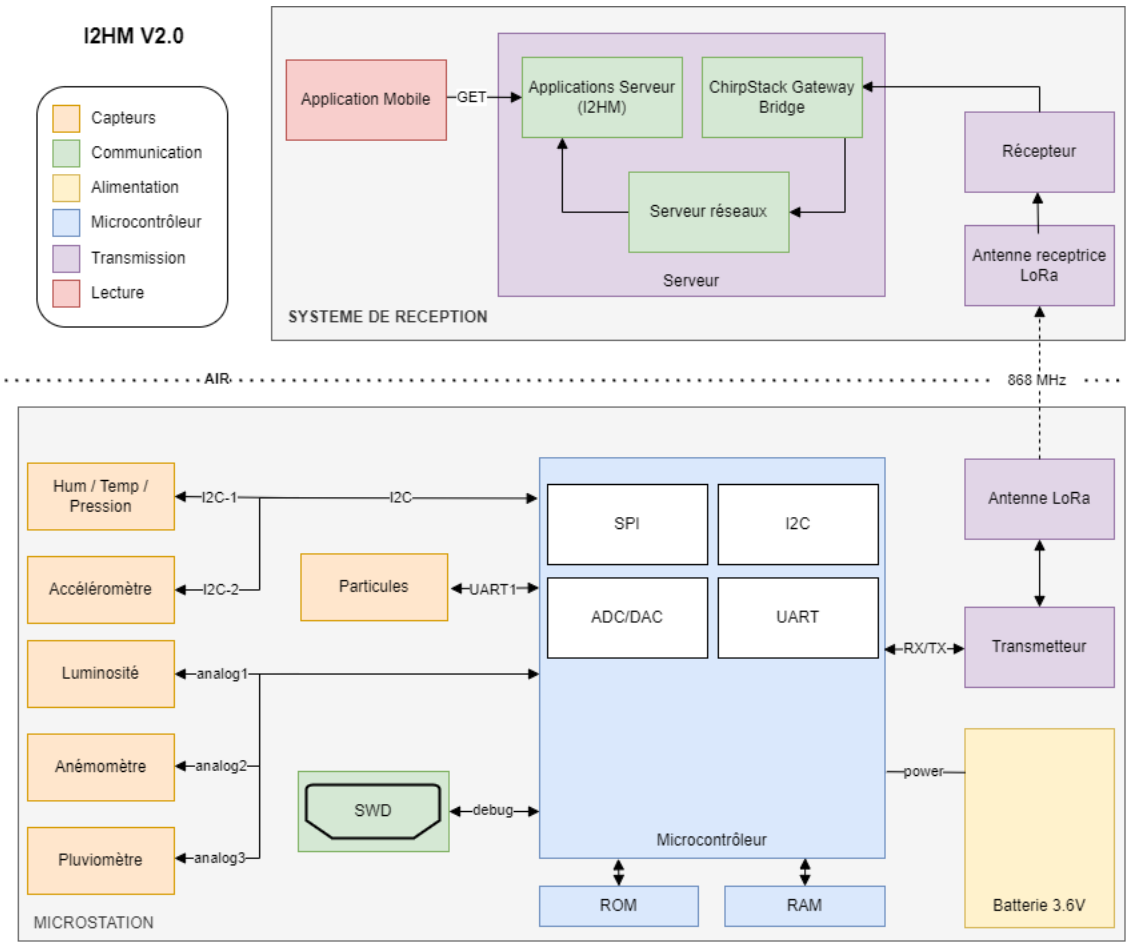

FIGURE 2 – Vue globale du système

À la fin de ce projet, nous avons réussi à connecter uniquement des capteurs utilisant une communication I2C : le BME680 et le MMA8451. Bien que d'autres capteurs aient été envisagés, nous avons rencontré des problèmes. Le PMS6003, conçu pour fonctionner avec une liaison UART, n'a pas pu être intégré en raison du manque d'un câble spécifique, qui n'était pas disponible au moment du projet. De plus, l'implémentation de capteurs analogiques était prévue, mais aucune broche analogique n'était disponible sur la devboard.

Cependant, la stratégie adoptée n'a pas d'incidence sur la poursuite du projet. Cette décision sera expliquée en détail dans la section sur la Compression des données.

3 Compression des données

Dans le but de réduire la consommation d'énergie lors de la transmission des données, nous avons travaillé sur la compression des données des capteurs avant leur envoi. Notre approche consiste à optimiser l'utilisation des données en se concentrant sur le cadre d'utilisation de l'environnement, tout en négligeant les valeurs moins pertinentes, afin d'assurer une précision maximale tout en minimisant la taille des données.

L'idée est donc d'envoyer une plage associée à une température plutôt que d'envoyer directement la température finale. Cette plage varie de 0 à 255 afin que les données soient contenues dans un seul octet.

Cette stratégie permet, d'une part, de réduire le coût énergétique lors de l'émission des données, d'autre part, de répondre aux exigences spécifiées par les étudiants en TIS concernant la précision minimale requise, et enfin, d'obtenir un modèle entièrement modulable.

3.1 Compression des données du BME680

Le capteur BME680 [2] est facile à prendre en main notamment grâce aux librairies disponibles. En revanche, les données résultantes sont de types float soit de quatre octets.

3.1.1 Température

Le BME680 est capable de mesurer une plage de température allant de -40 à 85°C. En se limitant à la plage de 0 à 50°C, qui couvre la totalité des cas d'utilisation, il est possible d'obtenir une précision de $50/256 \approx 0.2$°C par plage sur un octet.

Ainsi, la formule résultante est la suivante :

$$\text{Température} = \text{plage\_temp} \times 0.2 \tag{1}$$

3.1.2 Humidité

Le capteur est également capable de mesurer l'humidité en pourcentage. Il n'est pas possible de restreindre l'intervalle d'utilisation, mais celui-ci est déjà très limité : de 0 à 100 %. Il est possible d'obtenir une précision de $100/256 \approx 0.4$°% par plage sur un octet.

Ainsi, la formule résultante est la suivante :

$$\text{Humidité} = \text{plage\_humi} \times 0.4 \tag{2}$$

3.1.3 Pression

Ce capteur est aussi capable de mesurer la pression atmosphérique (hPa). En se limitant à la plage de 990 à 1020 hPa, qui couvre la totalité des cas d'utilisation, il est possible d'obtenir une précision de $((1020 - 990)/256) \approx 0.15$ hPa par plage sur un octet.

Ainsi, la formule résultante est la suivante :

$$\text{Pression} = (\text{plage\_pression} \times 0.15) + 990 \tag{3}$$

3.2 Compression des données de l'accéléromètre

Le capteur MMA8451 [3], tout comme le BME680, est facile à prendre en main grâce aux bibliothèques disponibles pour son implémentation. Il retourne une accélération (m/s^{-2}) sur chaque axe sous forme de valeur flottante définie sur quatre octets.

3.2.1 Accélération

Le MMA8451 renvoie les valeurs d'accélération (en $\mathrm{m/s^{-2}}$) sur les axes x, y et z. Ce capteur peut être configuré jusqu'à $8g$, mais dans le cadre de notre utilisation, nous nous contenterons d'une configuration jusqu'à $2g$, ce qui est amplement suffisant dans le contexte du projet. Par conséquent, nous utiliserons l'intervalle -20 à 20 $\mathrm{m/s^{-2}}$ (rappel : $1g = 9{,}81$ $\mathrm{m/s^{-2}}$). Il est donc possible d'obtenir une précision de $\frac{20-(-20)}{256} \approx 0{,}16$ $\mathrm{m/s^{-2}}$ par plage sur un octet.

Ainsi, la formule résultante est la suivante :

$$\text{Accélération} = \text{plage\_accel} \times 0.16 \tag{4}$$

4 Transmission des données

L'étape suivante consiste à envoyer les données des capteurs vers l'application distante, dans notre cas ChirpStack. Pour ce faire, une première opération est nécessaire pour selectionner le mode d'activation :

— **AT+NJM = <your_njm>** - Mode d'activation selon votre utilisation (0 : APB et 1 : OTAA)

Il est possible d'effectuer une tentative de connexion pour d'assurer de la communication entre l'émetteur et le recepteur :

— **AT+JOIN = 1 :0 :8 :8** - Tentatives de connexion vers l'application distante.

Si la connexion est établie, vous recevrez le message suivant depuis le port série :

— RAKwireless LoRaWan OTAA - I2HM

— ————————————————

— Wait for LoRaWAN join...+EVT :JOINED

4.1 Code de détection

Dans un souci d'efficacité énergétique, nous avons instauré un protocole pour évaluer la fiabilité des capteurs. Ce processus implique l'envoi d'un code de détection, encapsulé dans un octet et transmis au début de la trame. Étant donné qu'un octet comprend 8 bits, il devient ainsi envisageable d'indiquer l'état de fonctionnement de jusqu'à 8 capteurs au moyen d'une séquence binaire composée de 1 et de 0. Par exemple la séquence 0b10000001 indique que le capteur 1 et le capteur 8 sont en état de fonctionner.

Le code de détection (DC) permet, en l'état, de donner l'information de viabilité des capteurs BME680 et MMA8451. Il est facilement possible d'implémenter la même fonctionnalité pour les capteurs manquants.

4.2 Trame de données

En fonction du centre de données (DC), la trame s'adapte pour recenser les données des capteurs opérationnels :

DC	Temp	Humi	Pression	Ax	Ay	Az
0x3	0xA	0xB	0xC	0xD	0xE	0xF

TABLE 1 – Trame de données - Tous les capteurs sont opérationnels

Dans le cas suivant, le code de détection vaut 0x3 soit 0b11. Cela indique que les deux capteurs sont bien en état de fonctionnement et que les données issues des capteurs peuvent être émises.

DC	Temp	Humi	Pression
0x2	0xA	0xB	0xC

TABLE 2 – Trame de données - Seul le capteur BME680 est opérationnel

Dans le cas suivant, le code de détection vaut 0x2 soit 0b10. Cela indique que seul le capteur BME680 est en état de fonctionnement et que les données issues de ce capteur peuvent être émises.

DC	Ax	Ay	Az
0x1	0xA	0xB	0xC

TABLE 3 – Trame de données - Seul le capteur MMA8451 est opérationnel

Dans le cas suivant, le code de détection vaut 0x1 soit 0b01. Cela indique que seul le capteur MMA8451 est en état de fonctionnement et que les données issues de ce capteur peuvent être émises.

5 Création d'un décodeur de données

Après la mise en place de notre plan d'action, la création du décodeur est relativement simple est dépend uniquement de la valeur associée au code de détection et de la formule permettant de convertir une plage de mesure en une valeur.

```
function Decode(fPort, bytes, variables) {

  // Information sur la liste des capteurs opérationnels
  var detected_code = bytes[0];
  var state_bme680 = (detected_code >> 1) & 1;
  var state_accel = detected_code & 1;

  // Pointeur vers l'index de la trame courante (ici 1 car 0 étant le detected_code)
  var indexPacket = 1;

  // Stockage des données potentielles
```

```
  var json_result = {};

  if(state_bme680)
  {
    var indexTemp = bytes[indexPacket++];
    var indexHumidity = bytes[indexPacket++];
    var indexPressure = bytes[indexPacket++];
    var Temp = indexTemp * 0.2;
    var Humidity = indexHumidity * 0.4;
    var Pressure = (indexPressure * 0.15) + 990;
    json_result.temperature = Temp;
    json_result.humidity = Humidity;
    json_result.pressure = Pressure;
  }

  if(state_accel)
  {
    var indexAx = bytes[indexPacket++];
    var indexAy = bytes[indexPacket++];
    var indexAz = bytes[indexPacket++];
    var ax = (indexAx * 0.16) - 20;
    var ay = (indexAy * 0.16) - 20;
    var az = (indexAz * 0.16) - 20;
    json_result.ax = ax;
    json_result.ay = ay;
    json_result.az = az;
  }

  return json_result;
}
```

Vous devriez alors être en mesure de recevoir les données décodées (voir la figure 3) depuis votre dashboard ChirpStack [7].

FIGURE 3 – Données décodées depuis ChirpStack

Il est possible d'obtenir un dashboard avec un aspect encore plus graphique notamment par le biais de Grafana (voir la figure 4). Un tutoriel est disponible depuis GitHub [8].

FIGURE 4 – Données décodées depuis Grafana

6 Conclusion

Pour conclure, nous constatons que le projet progresse de manière satisfaisante. La dev-board présente quelques limitations et que la réception du PCB ayant pris du temps, nous n'avons pas pu avancer dans l'implémentation des capteurs analogiques et UART. Cependant, la stratégie que nous avons adoptée permettra une intégration rapide et efficace de ces composants ultérieurement.

À ce jour, aucun test n'a été effectué concernant la consommation pratique, en dehors des stratégies visant à prolonger la durée de vie de la batterie. Il serait intéressant de fournir des données chiffrées à l'aide d'un PPK2 [9], notamment pour évaluer la durée de vie prévisible de la batterie en l'absence d'un système de recharge interne.

À la fin du projet, une nouvelle carte de développement, la RAK3272S-SiP [10], a été reçue, offrant des fonctionnalités plus avancées que son prédécesseur. Cette carte fonctionne directement à la fréquence 868MHz, éliminant ainsi le besoin d'une passerelle (gateway) lors

de vos tests. De plus, l'intégration des capteurs manquants pendant les phases d'essai est possible grâce aux broches supplémentaires. La disposition des broches reste identique à celle de la RAK3272S, facilitant ainsi le processus de plug and play. Une application a été créée depuis ChirpStack afin de procéder à quelques tests qui s'avèrent concluants.

Références

[1] RAKwireless, “Rak3272s breakout board datasheet,” 2022. [Online]. Available : https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3272S-Breakout-Board/Datasheet/

[2] Adafruit, “Datasheet bme680,” 2017. [Online]. Available : https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf

[3] NXP, “Datasheet mma8451,” 2017. [Online]. Available : https://www.nxp.com/docs/en/data-sheet/MMA8451Q.pdf

[4] ST, “Download stm32cubeprogrammer,” 2020. [Online]. Available : https://www.st.com/en/development-tools/stm32cubeprog.html

[5] RAKwireless, “Firmeware rak3272s,” 2017. [Online]. Available : https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3272S-Breakout-Board/Datasheet/#specifications

[6] ——, “At commands list,” 2024. [Online]. Available : https://docs.rakwireless.com/RUI3/Serial-Operating-Modes/AT-Command-Manual/#overview

[7] D. Poissonnet, “Dashboard i2hm chirpstack,” 2024. [Online]. Available : https://lns.campusiot.imag.fr/#/organizations/57/applications/494

[8] D. Donsez, “Tutoriel grafana,” 2022. [Online]. Available : https://github.com/CampusIoT/RIOT-wyres/blob/main/tutoriel/08a.md

[9] Nordicsemi, “Power profiler kit ii,” 2022. [Online]. Available : https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2

[10] RAKWireless, “Power profiler kit ii,” 2022. [Online]. Available : https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3272-SiP-Breakout-Board/Quickstart/