Louis Allain

1. Passerelle entre les capteurs et le broker MQTT

1.1 Gestion de la connexion Wifi

Un premier moyen pour que la passerelle puisse se connecter à un réseau Wifi est de donner dans le code notamment le SSID et le mot de passe du point d'accès auquel devra se connecter la passerelle.

Pour plus de flexibilité, on lui permet d'être capable de supporter deux modes.

Un mode dans lequel la passerelle agit en tant que point d'accès Wifi et de serveur web. Ce serveur web permet à un client de configurer notamment le SSID et le mot de passe du réseau Wifi auquel la passerelle pourra ensuite se connecter. Cette configuration est sauvegardée dans la mémoire EEPROM de la carte afin que ces informations ne se perdent pas d'un démarrage à l'autre.

L'autre mode de la passerelle a pour but la transmission des données issues des capteurs. Pour cela, la passerelle agit en tant que station Wifi et se connecte au réseau Wifi défini depuis la configuration enregistrée dans la mémoire EEPROM de la carte. Dans le cas où le point d'accès Wifi auquel la passerelle doit se connecter tombe en panne, la passerelle tente de s'y connecter toutes les 10 secondes afin d'éviter trop de tentatives de connexion à la suite. Toujours dans le même scénario, la passerelle n'essaie plus de recevoir les données des capteurs et donc n'essaie pas d'envoyer les données issues des capteurs au broker MQTT puisqu'elle n'est plus connecté au point d'accès Wifi.

Afin de basculer d'un mode à l'autre, la passerelle est munie d'un bouton. Dans la boucle du programme, l'état du bouton est consulté afin de passer en mode configuration ou en mode envoie de données.

1.2 Réception des données depuis les capteurs

La réception des données peut se faire grâce à la librairie **RadioHead** et notamment grâce à la classe **RHMesh** de cette librairie. Le détail de l'utilisation de cette librairie est dans la partie suivante concernant les boîtiers capteurs.

Pour ce qui est de la passerelle, elle est chargée uniquement de recevoir des messages, pour cela on lui assigne une adresse (un entier sur 8 bits selon la classe RHMesh). Et c'est à cette adresse que les boîtiers capteurs enverront leurs messages.

Les données des capteurs se transmettent dans une structure C qui peut être par exemple :

```
struct payload_t {
  uint8_t id_capteur[8]; // id de l'émetteur, DS18B20 id = 64 bits
  uint8_t id_passerelle[8]; // id de la passerelle, DS18B20 id = 64 bits
  float temperature;
  float humidite;
}
```

Le premier élément de cette structure est l'identité du capteur. Celui-ci correspond à l'identifiant unique du capteur DS18B20 présent sur chaque capteur (chaque puce 1-Wire possède un identifiant unique).
Le reste de la structure comporte la température et l'humidité.

En utilisant la classe RHMesh, on évite de lire du bruit qui serait présent sur la bande de fréquence utilisée mais cela n'empêche pas que d'autres appareils puissent utiliser la même libraire voire la même classe et la même adresse de destination que la passerelle.
Pour cela, il est possible d'ajouter à la passerelle une puce 1-Wire qui pourrait servir d'identifiant unique, ainsi les boîtiers capteurs ajouteraient dans la charge utile du message l'identifiant unique de la passerelle (voir la structure C précédente) et celle-ci pourrait le vérifier lorsqu'elle reçoit un message.
Ou bien, sans utiliser une puce 1-Wire, il faudrait générer un identifiant unique par programmation.

1.3 Transmission des données au broker MQTT

Les données reçues des capteurs sont transmises à un broker MQTT si la passerelle est bel et bien connectée à un réseau Wifi et si les données reçues sont bien destinées à la passerelle.

Si toutes ces conditions sont validées, la passerelle peut tenter de se connecter au broker MQTT, si cette connexion est valide, la passerelle peut enfin transmettre les données reçues.

L'adresse et le port du broker MQTT peuvent être donnés dans le code directement ou bien configurés selon le même mécanisme de configuration décrit dans la partie précédente "1.1 Gestion de la connexion Wifi".

Il est possible de publier ces informations de plusieurs manières.
Une première façon serait de publier l'identifiant du capteur, la température et l'humidité mesurées par celui-ci dans un sujet MQTT (par exemple : /mesures). Ainsi, pour exploiter les données, un client qui s'abonne à ce sujet MQTT doit être capable de faire le lien entre l'identifiant du capteur et la pièce où il se trouve.

Une autre manière de faire implique que la passerelle soit en mesure elle-même de faire le lien entre les identifiants des capteurs et les pièces où ils se trouvent. Ainsi, plutôt que de publier

l'identifiant du capteur sur un sujet MQTT, la passerelle pourrait publier soit directement dans un sujet MQTT les valeurs (par exemple : /mesures/cuisine) ou bien publier le nom de la pièce en plus des valeurs dans un sujet comme "/mesures". Pour ce faire, on pourrait ajouter au mode configuration de la passerelle une fonctionnalité permettant d'ajouter un capteur. En ajoutant par exemple un champ permettant de préciser le nom de la pièce dans laquelle se trouve le capteur et son identifiant. Ces informations seront de la même manière ajoutées à la mémoire EEPROM. C'est une solution plus confortable pour exploiter les données mais le nombre de capteurs pouvant être ajoutés serait limité à cause de la taille de la mémoire EEPROM de la carte (par exemple 1 Ko pour une carte Arduino Nano).

1.4 Bibliothèques utilisées

Voici la liste des bibliothèques susceptibles d'être utilisées pour réaliser la passerelle :

- Pour la connexion Wifi : la libraire **Wifi** intégrée au cadre de développement Arduino (https://www.arduino.cc/en/Reference/WiFi).
- Éventuellement pour le serveur web en mode configuration : la librairie **WifiClient** intégrée au cadre de développement Arduino (https://www.arduino.cc/en/Reference/WiFiClient).
- Pour la gestion des communications OOK : la librairie **RadioHead** de Mike McCauley (https://www.airspayce.com/mikem/arduino/RadioHead/).
- Pour la connexion au broker MQTT et la publication des données sur un sujet : la librairie **MQTTClient** de Andreas Motzek (https://bitbucket.org/amotzek/mqtt-client/src/master/).

2. Boîtier capteur

2.1 Démarrage

Hormis l'initialisation des composants du boîtier capteur, au démarrage un boîtier capteur attend un nombre aléatoire de ms (<10 000 ms par exemple) afin d'éviter que tous les capteurs envoient des messages en même temps.

2.2 Mesures et transmission des données

Toutes les 5 minutes, la carte récupère les mesures depuis la sonde de température et la sonde d'hygrométrie.

Les capteurs peuvent ne pas être tous à portée radio de la passerelle, mais on suppose qu'à tout moment (sauf si un ou plusieurs capteurs tombent en panne) il existe une route permettant à un capteur via d'autres d'atteindre la passerelle.

En supposant cela, il est possible d'utiliser la classe RHMesh (en plus du driver approprié) proposée par la librairie RadioHead. Cette classe fournit un moyen d'envoyer des messages adressés, potentiellement acquittés à travers un réseau avec de la découverte automatique de route. Dans notre cas, c'est intéressant puisque nous ne connaissons pas à l'avance le graphe du réseau que forment les boîtiers capteurs. Pour ce faire, chaque boîtier capteur dans le réseau doit posséder une adresse unique (un entier sur 8 bits) afin de construire une table de routage.
Pour générer une adresse unique, il est possible d'utiliser un générateur de nombre entier aléatoire entre 0 et 254 (255 est réservé pour l'adresse de diffusion dans la classe RHMesh). Mais intuitivement, vu l'intervalle dans lequel seront tirées aléatoirement les adresses il est possible que des adresses soient dupliquées surtout s'il y a un nombre relativement important de boîtiers capteurs.
Il pourrait être aussi possible d'utiliser une partie de l'identifiant unique de la puce 1-Wire du boîtier capteur pour l'adresse.

Une fois le problème des adresses résolu, le boîtier capteur envoie les données de mesures à l'adresse de la passerelle (connue de tous les boîtiers capteurs) et la librairie se charge de résoudre la route vers celle-ci. Cette procédure est détaillée dans la documentation de la librairie[1].

Un problème inhérent à l'utilisation de cette bibliothèque est qu'elle permet à un nœud de traiter uniquement un message à la fois, c'est pour cela qu'il est important de déphaser l'envoi des messages entre tous les boîtiers capteurs afin qu'ils soient capables de relayer les messages vers la passerelle. De même pour cette dernière, il faut qu'elle soit capable de traiter tous les messages provenant des boîtiers capteurs.

Evidemment, le fait pour chaque nœud de maintenir un système de découverte de route (table de routage, etc.) consomment un montant de SRAM significatif (environ 2 Ko). Il est donc important de choisir pour les boîtiers capteurs une carte Arduino disposant de suffisamment de mémoire vive. Typiquement la carte Arduino Nano 3.x basée sur le microcontrôleur ATmega328 ne suffirait pas puisque la SRAM de celui-ci est de 2 Ko.

Il faut aussi prendre en compte le fait que certains boîtiers capteurs peuvent tomber en panne temporairement, ce qui implique que certaines données de mesures ne peuvent plus êtres envoyées à la passerelle. Il est peut être utile de repérer les pannes.
Pour cela, la passerelle peut maintenir un nombre de cycle maximum avant le repérage d'une panne pour chaque boîtier capteur. Par exemple, si au bout de deux cycles (soit 2 fois 5 minutes), la passerelle n'a rien reçu pour tel capteur, elle peut indiquer une panne en envoyant un message spécifique au broker MQTT. Cela ne peut se faire que si la passerelle a connaissance au moins du nombre total de boîtiers capteurs (voire de leurs identifiants pour un repérage d'erreurs plus précis).
Cette détection d'erreurs peut se faire également du côté du client utilisant les données provenant du broker MQTT selon le même principe.

[1] https://www.airspayce.com/mikem/arduino/RadioHead/classRHMesh.html

2.3 Bibliothèques utilisées

Voici la liste des bibliothèques susceptibles d'être utilisées pour réaliser les boîtiers capteurs :

- Pour la gestion de la sonde d'hygrométrie DHT11 : la librairie DHT d'AdaFruit décrite sur le site d'Arduino (https://www.arduino.cc/reference/en/libraries/dht-sensor-library/).
- Pour la gestion de la sonde de température : la librairie DS18B20 de Mathias Munk Hansen décrite sur le site d'Arduino (https://www.arduino.cc/reference/en/libraries/ds18b20/).
- Pour la gestion des communications OOK : la librairie **RadioHead** de Mike McCauley (https://www.airspayce.com/mikem/arduino/RadioHead/).