Modem LoRa Postman

Version 1.0.4

Strataggem

nov. 08, 2017

Table des matières

CHAPITRE 1

Fonctionnement général

POSTMAN est un modem radio LoRa qui est contrôlé via un protocole série.

POSTMAN fonctionne comme un bureau de poste : l'utilisateur dispose de 2 boîtes aux lettres qui lui permettent d'envoyer et de recevoir des messages via le protocole LoRa.

La boîte aux lettres d'entrée est appellée INBOX.

La boîte aux lettres de sortie est appellée OUTBOX.

CHAPITRE 2

LED RVB

La LED RVB permet de donner le statut du modem.

Au démarrage elle donne en plus la version de code chargée.

2.1 Séquence de démarrage

- ROUGE : 200ms
- PAUSE
- BLANC
- SEQUENCE_VERSION
- BLANC
- PAUSE

Version	Couleur 1	Couleur 2
1.0.0	ROUGE	VERT
1.0.3	BLEU	ROUGE
1.0.4	VERT	BLEU

2.2 Fonctionnement normal

Clignotement

Couleur Clignotement	Mode de fonctionnement
VERT	Fonctionnement standard
ROUGE	Envoi LoRa

CHAPITRE 3

Broches d'interruptions

3.1 EMPTY_OUT

Lors d'un envoi radio (dépilage de la file de messages OUTBOX), la broche EMPTY_OUT passe à l'état HAUT pendant 1-2ms.

CHAPITRE 4

Protocole

4.1 Généralités

Structure d'une commande vers le modem :

start_marker (0x55)	command_type	command_payload_size	payload	crc16
uint8	uint8	uint16	bytes	uint16

Structure d'une réponse du modem :

start_marker (0x55)	command_type	request_status	command_payload_size	payload	crc16
uint8	uint8	uint8	uint16	bytes	uint16

À faire

Détailler requête et réponse

À faire

Le command_type d'une réponse est celui de la requête

À faire

Les requêtes envoyées vers le modem sont traitées et vérifiées champ par champ. Ainsi, si un paramètre de la requête est incorrecte, une réponse contenant l'erreur correspondante dans l'octet command_status sera émise immédiatement (donc sans attendre la fin de la réception de la requête). Une réponse émise pendant une réception d'une requête contiendra forcément un message d'erreur et pourra donc être traîté par interruption afin de couper l'émission de la requête.

4.1.1 Status de réponse (request_status)

Le code de status envoyé dans la réponse suite à une requête vers le modem peut prendre les valeurs suivantes :

request_status : uint8_t

→ **0x00 / OK** → payload de réponse à la commande

→ **0x01 / Commande inconnue** → payload vide

→ **0x02 / Délais d'attente dépassé**

→ **payload donnant la cause du timeout** → 0x01 / Attente du type de commande

→ 0x02 / Attente de la taille du payload

→ 0x03 / Attente du payload

→ 0x04 / Attente du CRC

→ **0x03 / Valeur de taille de payload trop grande** → payload vide

→ **0x04 / Valeur de payload inattendue** → payload vide

→ **0x05 / CRC invalide** → payload vide

→ **0x10 / Erreur d'execution de la commande** → payload de réponse d'erreur à la commande (voir commande)

4.1.2 Numérotation des messages des inbox/outbox

La numérotation commence à 0 et s'incrémente de 1 à chaque message. Au bout de 255 messages elle boucle.

4.1.3 Calcul du CRC

Tous les messages échangés avec le modem sont validés par un CRC (CRC16-CCITT) dont le polynôme est 0x1021 et la valeur d'initialisation 0xFFFF

```
uint16_t crc16(uint8_t* data_p, uint32_t length){
    uint8_t x;
    uint16_t crc = 0xFFFF;

    while (length--){
        x = crc >> 8 ^ *data_p++;
        x ^= x>>4;
        crc = (crc << 8) ^ ((uint16_t)(x << 12)) ^ ((uint16_t)(x <<5)) ^ ((uint16_
↪t)x);
    }
    return crc;
}
```

4.1.4 Messages

4.2 sys get version (0x01)

Récupère la version du microcode

4.2.1 Requête

Payload vide : [55 01 00 00 7A 8E]

4.2.2 Réponse

Chaine de caractères terminée par '\0' (0x00). La taille maximale de la chaine retournée est de 16 octets.

4.2.3 Exemple

field_name	binary	decoded
version	31 2e 30 2e 33 00	1.0.3

4.3 sys get uuid (0x02)

Récupère l'identifiant unique du modem

4.3.1 Requête

Payload vide

4.3.2 Réponse

uuid : uint8_t[16]

4.4 sys get addr (0x03)

Récupère l'identifiant réseau du modem

4.4.1 Requête

Payload vide

4.4.2 Réponse

address : uint8_t[4]

4.5 sys get timeout (0x04)

Récupère le timeout actuel en millisecondes pour la réception du payload pour la commande MB_OUT_PUSH

4.5.1 Requête

Payload vide

4.5.2 Réponse

timeout (in ms) : uint32_t

4.6 sys set timeout (0x05)

Fixe le timeout en millisecondes pour la réception du payload pour le prochain appel à MB_OUT_PUSH. Le timeout est remis à la valeur par défaut après l'appel à MB_OUT_PUSH (1sec).

Note : Plage de valeurs autorisées : 1000 à 60000ms.

4.6.1 Requête

timeout (in ms) : uint32_t

4.6.2 Réponse

Succès : payload vide

Echec : cause : uint8_t

→ 0x01 / OutOfRange (valeur en dehors de la plage autorisée)

4.7 net is_online (0x11) [WIP]

Avertissement : Work in progress ! This command might not be usable at the moment.

Donne le statut de connexion du modem

4.7.1 Requête

Payload vide

4.7.2 Réponse

connection_status : uint8_t → 0x00 / DISCONNECTED → 0x01 / CONNECTED

4.8 net set encryption_key (0x12)

Définit la clé de cryptage des données

4.8.1 Requête

encryption_key : uint8_t[16]

4.8.2 Réponse

Payload vide

4.9 net set_addr (0x13)

Définit l'adresse réseau

4.9.1 Requête

— address_field_1 : uint8
— address_field_2 : uint8
— address_field_3 : uint8
— address_field_4 : uint8

4.9.2 Réponse

— Pas de données supplémentaires à part le statut de réponse OK

4.10 mailbox out push (0x20)

Place en message dans l'outbox

4.10.1 Requête

msg_data : uint8_t [total_msg_size - (header_size + 1)]

4.10.2 Réponse

Succès : uid : uint8_t

Echec : cause : uint8_t
→ 0x01 / BufferFull (Mémoire insuffisante)
→ 0x02 / SlotsFull (Nombre maximal de message atteint)
→ 0x11 / UnknownError (erreur inconnue)

4.11 mailbox out count (0x21)

Donne le compte de messages dans l'outbox

4.11.1 Requête

Payload vide

4.11.2 Réponse

count : uint32_t

4.12 mailbox in get (0x22)

Récupère un message dans l'inbox à l'index donné

4.12.1 Requête

index : uint32_t

4.12.2 Réponse

Succès : msg_data : uint8_t [total_msg_size - (header_size + 1)]
Echec : cause : uint8_t
→ 0x05 / SlotOverflow (index supérieur au nombres max de messages)
→ 0x06 / MessageCorrupt (le message est corrompu)
→ 0x07 / EmptySlot (pas de message à cet index)
→ 0x08 / SizeLimit (taille du message supérieure au maximum autorisé)

4.13 mailbox in count (0x23)

Donne le compte de messages dans l'inbox

4.13.1 Requête

Payload vide

4.13.2 Réponse

count : uint32_t

4.14 mailbox in info (0x24) [WIP]

Avertissement : Work in progress ! This command might not be usable at the moment.

Récupère des informations sur un message dans l'inbox à l'index donné

4.14.1 Requête

index : uint32_t

4.14.2 Réponse

msg_type : uint8_t
msg_size : uint16_t

4.15 mailbox in pop (0x25)

Supprime un message de l'inbox à l'index donné

4.15.1 Requête

index : uint32_t

4.15.2 Réponse

Succès : payload vide

Echec : cause : uint8_t

→ 0x09 / MessageNotFound (Pas de message à cet index)

→ 0x0A / SlotCurrentlyReceiving (message en cour de reception à cet index)

→ 0x11 / UnkownError (erreur inconnue)

CHAPITRE 5

FAQ

1. Différence entre l'identifiant unique du modem (0x02) et l'identifiant réseau du modem (0x03) ?

L'identifiant réseau est variable

L'identifiant modem est unique et lié au matériel

2. Faut-il renvoyer ces informations à l'application Web ?

Uniquement pour paramétrer dans l'interface web, une fois

3. Quelle initialisation de la clef de cryptage (0x12) ?

Elle est choisie par vous ou bien générée par Wavebricks

4. Que veut dire net_is_online (0x11) : existence d'une communication au niveau du réseau ? ou communication possible ?

Permet de savoir si le modem se considère comme à portée d'une passerelle. C'est une simple information, tu n'as pas à la gérer

5. La valeur du compteur des messages est-elle initialisée à zéro à chaque mise sous tension ?

Elle est incrémentée, mais peut reboucler. Normalement tu ne dois pas faire une requête avec un numéro de message si tu ne l'as pas récupéré précédemment

7. Si cette valeur n'est pas réinitialisée, cela peut entraîner un problème de synchro avec le logiciel embarqué. Il faudrait alors une requête supplémentaire donnant le numéro du premier message dans la file d'attente.

Effectivement