Mode d'emploi LoRa

Un guide non exaustif de l'installation
compliquée d'une Gateway LoRa

Xavier Hueber
Noé Lindelaub
HE-ARC

27 avril 2023

Table des matières

1 Introduction

Ce guide créé dans le cadre du cours de Systèmes Communicants 2©vous expliquera comment installer les logiciels nécessaires à la création d'une Gateway LoRa à partir d'un HAT Raspberry Pi Seeed disposant d'une puce RHF0M301.
Nous utiliserons des nodes possédant un microprocesseur ESP32-S3 ainsi qu'un module série LoRa-E5-HF pour transmettre des messages par LoRaWAN®.

1.1 Matériel utilisé

Dans le cadre de ce projet, nous avons utilisé:

- Un kit LoRa/LoRaWAN®Gateway - 868MHz de Seeed.
- Un Raspberry Pi 3 modèle B.
- Deux ESP32-S3-DevKitC-1 v1.0 de Espressif.
- Deux modules LoRa-E5-HF de Seeed.
- Des antennes 868MHz SMA.
- Câble d'adaptation U-FL vers SMA.
- Une Gateway TheThingNetwork Kickstarter
- Une Gateway TheThingNetwork Indoor

1.2 Logiciels utilisés

Dans le cadre de ce projet, nous avons installé:

- Raspberry Pi OS 64-bit datant du 21 février 2023.
- Raspberry Pi Imager v1.7.4
- ChirpStack v3.
- RHF0M301-ChirpStack.
- Un éditeur de code moderne (ex: Sublime Text, VSCode, ...)
- Le compilateur IDF de Espressif

1.3 Glossaire

Terme	Acronyme
TheThingNetwork	TTN
Carte d'extension du Raspberry Pi	HAT

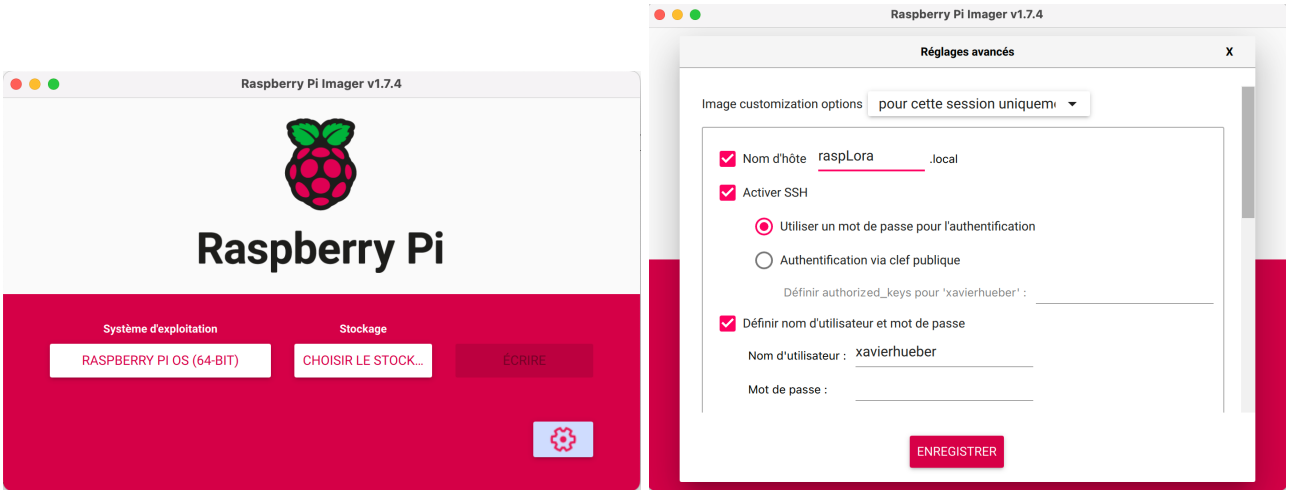
2 Installation

2.1 Installation du Raspberry Pi

2.1.1 Raspberry Pi OS

Le Raspberry Pi nous servira à recevoir, gérer et retransmettre les paquets reçus par LoRaWAN®.

Nous avons commencé par installer Rapsberry Pi OS 64 bit sur une carte sd grâce au logiciel Raspberry Pi Imager (voir figure 1a).
Dans celui-ci, nous pouvons activer la connexion SSH et configurer les logins de l'utilisateur, ainsi nous n'avons pas besoin d'utiliser un écran pour configurer le Raspberry Pi.



(a) Menu (b) Configuration du SSH et de l'utilisateur

Figure 1: Raspberry Pi Imager

2.1.2 Logiciels tiers

Maintenant que nous avons installé un système d'exploitation sur le Raspberry Pi, nous pouvons installer une application pour récupérer et gérer les paquets LoRaWAN®.

Nous voulions pour cela installer l'application proposé par Seeed pour l'utilisation de leur HAT, cependant nous n'avons pas trouvé de liens de téléchargement ni de guide d'installation disponible sur leur site web. Nous avons alors commencé par chercher des applications tierces compatibles avec le module RHF0M301. Une solution simple était d'installer basicstation, un logiciel développé par TTN qui transfère tous les paquets vers leurs serveurs.

Cependant, nous ne voulions pas pour commencer des tests, être dépendant de TTN et de leurs services (ceux-ci étant fortement limités, à 30 secondes d'émission par jour). Nous avons alors décidé de chercher une autre solution et sommes tombés sur ChirpStack[1]. ChirpStack est un ensemble d'applications permettant la réception, l'envoi et la gestion de paquets LoRaWAN®comme basicstation, mais elles permettent de gagner en flexibilité car chaque étape de traitement des paquets sont fait par une autre application de façon totalement indépendante.

ChirpStack nous donne ainsi accès au système que vous pouvez voir dans la figure 2:

1. L'application Gateway viens communiquer avec le HAT du Raspberry Pi 3 et publie les informations reçues par MQTT.
2. Le Network Server récupère alors les informations et les traite pour qu'elles soient utilisable par la User Application et publie sur MQTT.
3. La ou les User(s) Application(s) souscrivent au Network Server et reçoivent alors les informations traités.

[1] `https://www.chirpstack.io/docs/chirpstack-gateway-bridge/install/raspberry-pi.html`

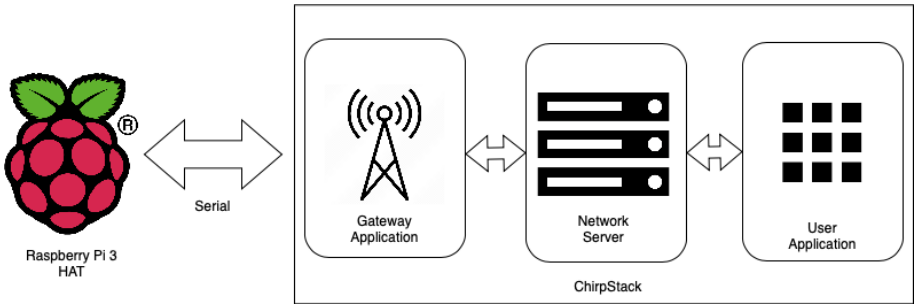


Figure 2: Système applicatif ChirpStack

2.2 Installation de l'environnement de développement Espressif

Maintenant que nous avons installé les tools requis pour faire fonctionner le Raspberry Pi, nous pouvons passer à l'installation de la toolchain ESP afin de pouvoir programmer nos deux Devkits.

Pour ce faire, nous nous rendons sur le site d'espressif et suivons le guide d'installation[2].
Nous avons ensuite compilé et flashé le projet Hello-World[3] pour tester le fonctionnement de la carte. C'est avec succès que celle-ci s'est mise en route et nous a donné ses premiers mots.

3 Programmation du Devkit ESP

Nous avons commencé par étudier la documentation fournie par Seeed pour son module LoRa-E5-HF. Sachant qu'il se pilote par série, il nous a fallu comprendre comment utiliser le port série du Devkit ESP.

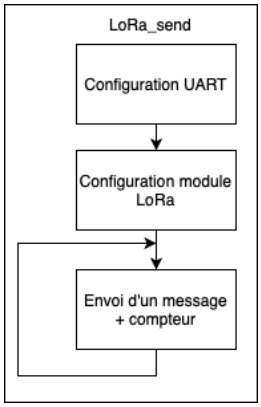
Pour cela, nous avons trouvé l'exemple uart_echo donné par Espressif.

Nous avons ensuite importé un partie du code de uart_echo dans notre application LoRa_send et utilisé les commandes AT que nous avions testé à la mains auparavant. Voici quelques commandes AT disponibles intéressantes ainsi que leur utilité:

Commande AT	Utilisation
AT	Répond AT+ pour dire que tout se passe bien
AT+UART=BR, baudrate	Permet de choisir le baudrate au prochain démarrage du module
AT+KEY= APPKEY, "KEY"	Permet de configurer l'AppKey utilisé lors de l'envoi de paquets LoRa
AT+MODE=LWOTAA	Permet de configurer le mode de connexion à la Gateway
AT+JOIN	Permet de rejoindre la Gateway
AT+MSG="CestNoe"	Permet d'envoyer un paquet LoRa contenant le message spécifié
AT+CMSG="CestNoe"	Envoi un paquet et demande une confirmation
AT+POWER=14	Permet de choisir la puissance d'émission en dB

Comme vous pouvez le voir dans la figure 3a, le programme est assez simple. Nous commençons par configurer la connexion UART requise pour communiquer avec le module LoRa, puis nous configurons ce dernier (voir figure 3b) et enfin nous avons écrit une boucle while qui envoi en boucle tous les X temps un message incrémenté.

[2] `https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/get-started/index.html#get-started-step-by-step`
[3] Voir notre répertoire Github



(a) Diagramme du logiciel LoRa_send

```
//Configure LoRa Module
//Test si tout est ok
sendLoraModule("AT\n", 3);
//On définit l'appkey
sendLoraModule("AT+KEY= APPKEY,\"14 46 2a 40 45 46 4c 7e a7 29 6c 07 5f c7 ba cc\"\n", 65);
//On passe en mode OTAA
sendLoraModule("AT+MODE=LWOTAA\n", 15);
//On met la pouissance au MAX
sendLoraModule("AT+POWER=24, FORCE\n", 19);
//On rejoint le réseau
joinNetwork();
```

(b) Configuration du module LoRa

Figure 3: LoRa_send

4 Configuration de ChirpStack

Nous avions vu précédemment que nous avions installé ChirpStack sur notre Raspberry Pi, il est maintenant temps de configurer ce stack d'applications pour pouvoir recevoir nos paquets LoRa. La configuration n'est pas compliqué, mais il faut quand même suivre un ordre précis sinon vous ne pourriez vous retrouver coincé. Cette dernière se fera dans l'interface web de ChirpStack disponible sur le port 8080 du Raspberry Pi.

4.1 Network Server

Dans un premier temps il va falloir configurer le Network Server, c'est lui qui est gère les paquets. Dans la figure 4 vous pouvez voir qu'on lui donne l'adresse localhost:8000 car ce dernier est installé en local. Si nous voulions mettre en place une plus grande architecture de Gateways, nous pourrions ainsi dédier un serveur unique comme étant le Network Server.

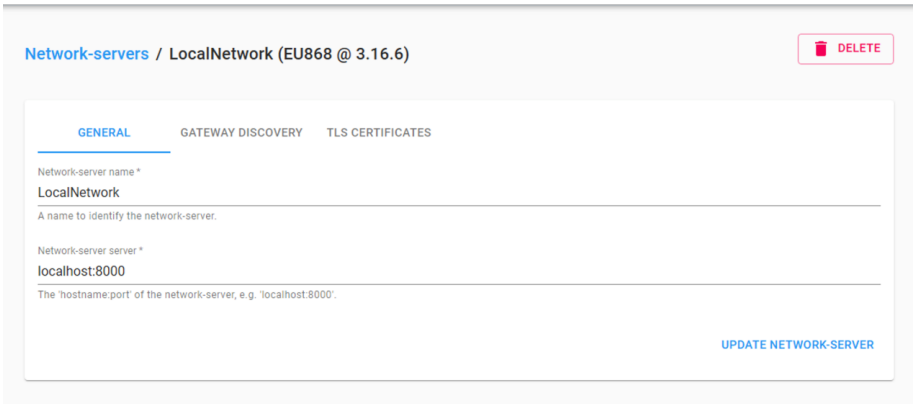


Figure 4: Network Server

4.2 Gateway Profile

Cette étape n'est pas une des plus intéressantes et cette option mériterait d'être configuré directement lors de la création de la Gateway dans l'interface de Chirpstack. Le profile de la Gateway permet de choisir le temps entre deux récupération de paquets du HAT Raspberry Pi.

On peut voir sur la figure 5 les options disponibles et comment nous les avons configurés.

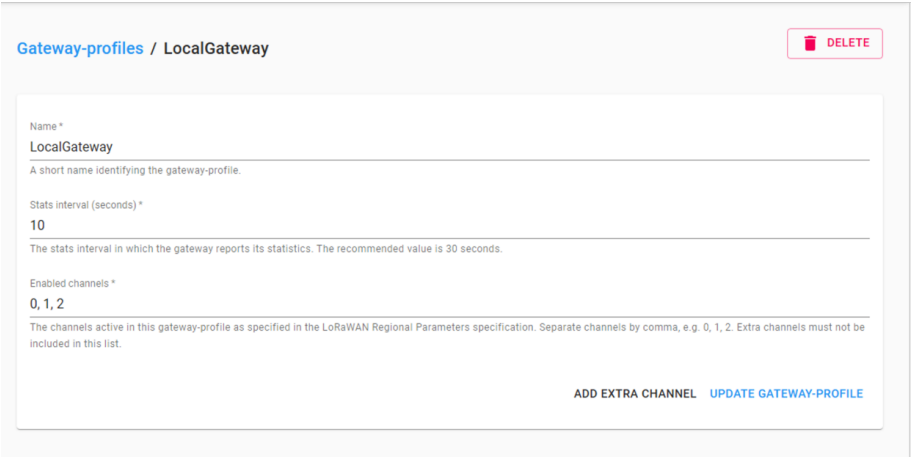


Figure 5: Gateway profiles

4.3 Device Profile

La configuration du Device profile permet de choisir comme son nom l'indique un profil pour les nodes LoRa que nous auront. On peut entre autre y choisir la version de LoRaWAN®supporté ainsi que la classe de la node et le mode de connection à la Gateway.

On peut voir sur la figure 6 que nous utilisons la version 1.0.4 de LoRaWAN et pour le mode de connection nous utilisons le LWOTAA comme spécifié plus haut dans le programme LoRa_send pour l'ESP32.

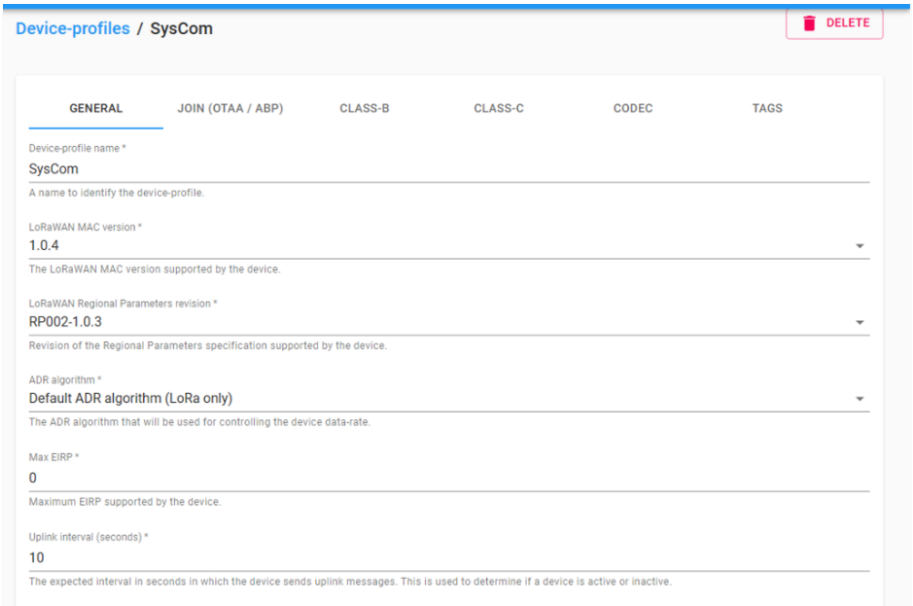


Figure 6: Device profile

4.4 Gateway

Nous pouvons maintenant configurer la Gateway:

Dans ce menu (voir figure 7), nous pouvons donner un nom à notre Gateway et un ID (généré aléatoirement en général). Il faut ensuite choisir le Network Server avec qui elle va communiquer et le Gateway profile.

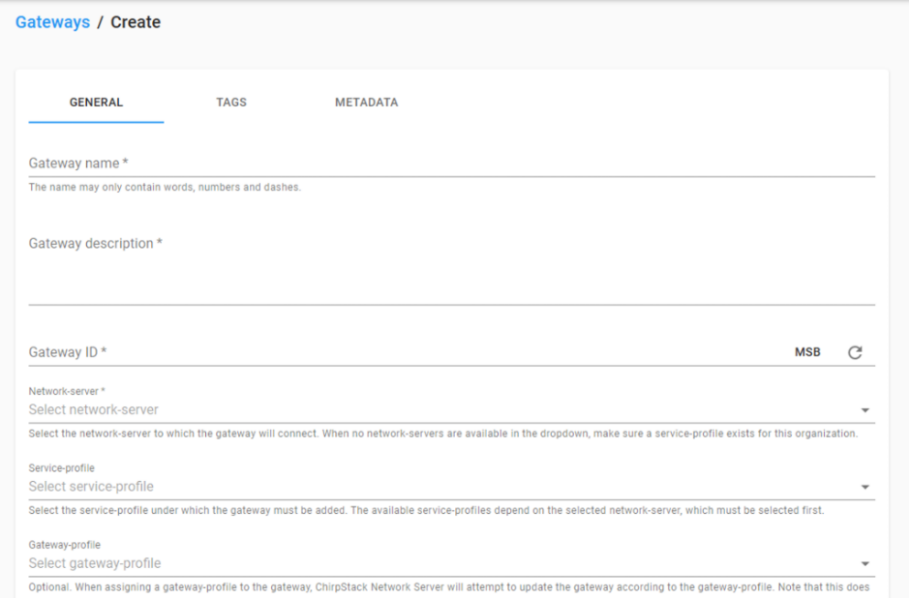


Figure 7: Configuration de la Gateway

4.5 Application

Une application est un groupement de nodes, une fois créée elle permettra d'ajouter des nodes qui pourront se connecter à notre Gateway et c'est aussi dans l'application que nous pourront choisir comment seront envoyé les paquets vers internet (voir le Chapitre 5).

Dans la figure 8, on peut donc choisir un nom pour notre application et une description.

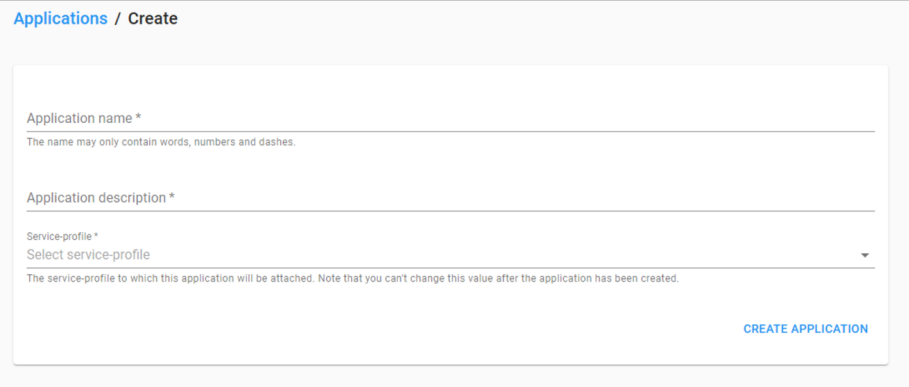


Figure 8: Configuration d'une application

4.6 Device

Enfin la dernière chose à configurer dans ChirpStack est la création d'une node.

Dans le menu de l'application nous pouvons ajouter un nouveau device, vous verrez alors apparaître la figure 9.

Ici il faut donc donner à un nom à notre node, un profil et un EUI qui correspond à l'identifiant de la node. Sur notre module LoRa Seeed, nous pouvons utiliser la commande série AT+ID=DevEui pour que le module nous donne son EUI.

Ensuite une fois que ces informations sont remplies, on peut créer le device et ChirpStack nous proposera alors de générer une APPKEY. Cette dernière est très importante car elle est la clé permettant à la node de se connecter à la Gateway, sans elle la Gateway refusera la node. On peut ensuite copier l'APPKEY et la coller dans le programme LoRa_send.

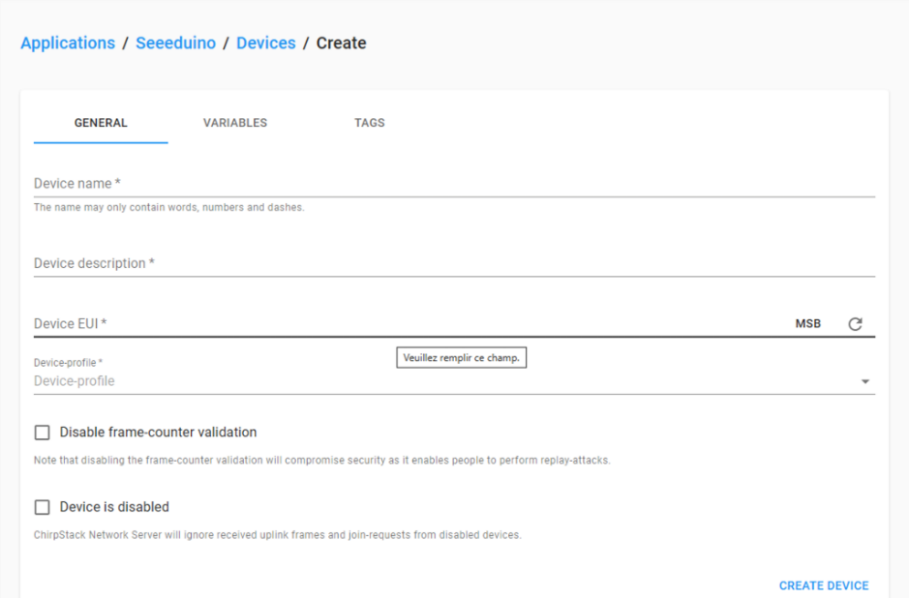


Figure 9: Configuration d'une node

Si la node après ça n'arrive pas à se connecter à la Gateway, il faudra voir si les versions de LoRaWAN sont les mêmes entre celle donnée dans le Device-Profile et celle configurée sur la node.

On peut aussi utiliser le site LoRaWAN 1.0.x packet decoder[4] pour voir si il y a un conflit d'APPKEY.

5 Envoi des paquets sur internet

La node arrive maintenant à envoyer des paquets LoRaWAN à la Gateway, cependant dans l'interface de ChirpStack nous recevons les messages sous formes de JSON et la donnée est en base64.

Nous voulions alors pouvoir traiter ces messages pour ne récupérer que les informations les plus importantes. ChirpStack nous permet alors d'utiliser d'autres applications comme MQTT ou Azure Service Bus (qui est une forme de MQTT propriétaire) pour envoyer les messages vers internet.

Nous avons testés plusieurs services, mais celui qui a fonctionné pour le mieux est Azure Service Bus. Vous pouvez voir sur la figure 10 un diagramme montrant comme les messages sont envoyés et transformés à l'aide d'un script python[5] vers Telegram.

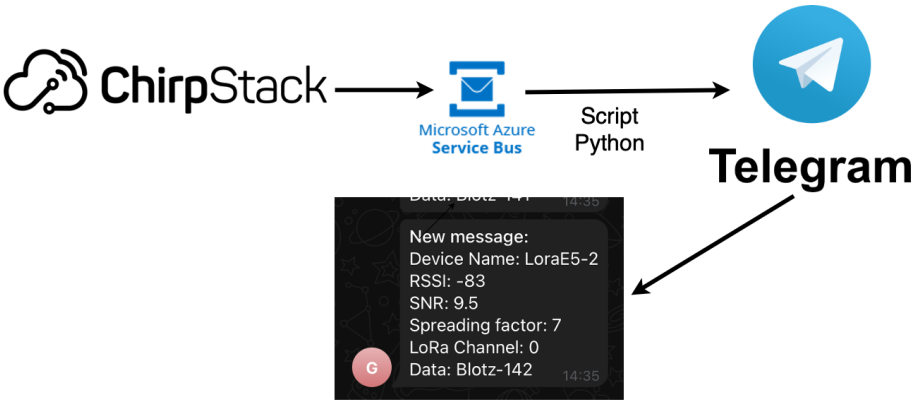


Figure 10: Script python

[4] `https://lorawan-packet-decoder-0ta6puiniaut.runkit.sh`

[5] Vous pouvez retrouver ce script dans notre répertoire Github

6 Versions

- Version 1.0 : 7 mai 2023 - Version originale
- Version 1.1 : 10 mai 2023 - Correction de fautes