

TP Déploiement d'un réseau LoRaWAN (3h)

Objectif :

- Infrastructure LoRaWAN
 - Transmission des données
-

1. Infrastructure	2
a. Introduction	2
b. VM IoT	4
2. Configuration du cœur de réseau Chirpstack	5
a. Introduction	5
b. Enregistrement d'un network server	5
c. Création d'un Service profile	5
d. Enregistrement d'une gateway	5
e. Création d'un Device profile	6
f. Création d'une Application	6
g. Création d'un Objet	6
3. Programmation du M5LoRaKit	6
a. Module LoRa	6
b. Bibliothèque LMIC	7
c. Join et Uplink	9
4. Envoi et visualisation des données ENVIII	10
a. Envoi	10
b. Réception	10
c. Décodage des données	10
5. Envoi d'un Downlink	10

1. Infrastructure

a. Introduction

Le matériel utilisé est le suivant :

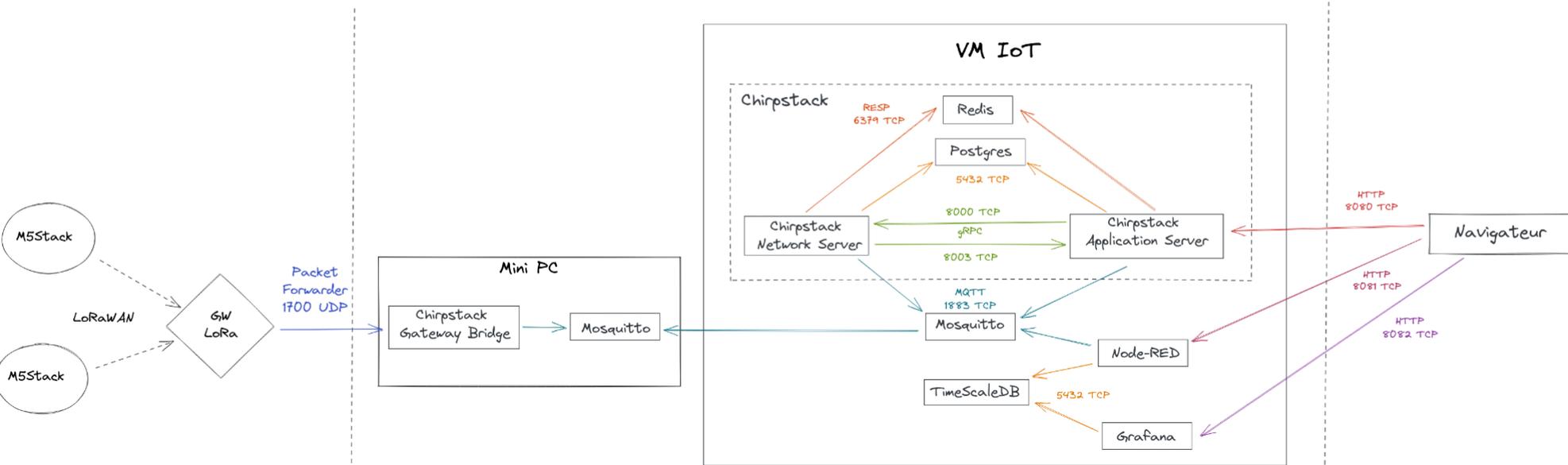
- IoT
 - M5LoRaKit
 - Mikrotik wAP LR8
- Serveurs
 - Mini-PC basé sur un Raspberry Pi CM 4
 - VM IoT Debian 11
- Terminaux
 - Navigateur web

L'architecture réalisée avec ce matériel est résumée par le schéma disponible en page suivante.

Iot

Serveurs

Terminaux



b. VM IoT

Une machine virtuelle pour héberger les services IoT est utilisée. Celle-ci est basée sur la distribution GNU/Linux Debian. Son utilisation est exclusivement en mode serveur, aucune IHM n'a été installée. Par conséquent, cette VM s'exécute aussi bien en local que sur une infrastructure dans le cloud.

L'ensemble de la configuration des services a été effectuée et ses derniers sont lancés automatiquement au démarrage (service systemd **docker-compose@iot**). Ceux-ci sont encapsulés dans des conteneurs Docker. Chaque conteneur possède son propre système de fichiers, ses propres connexions réseaux. Cela permet de faciliter le déploiement d'applications. La configuration des conteneurs est gérée par Docker compose au travers d'un fichier yaml, **docker-compose.yml** (situé sous `/etc/docker/compose/iot` sur la VM). Les différents services sont connectés sur le même réseau Docker et sont accessibles **sur ce réseau** avec les noms d'hôtes suivants :

- Chirpstack
 - Application server : chirpstack-application-server
 - Network server : chirpstack-network-server
- Broker MQTT : mqtt-broker
- Base de données séries temporelles : timescaledb
- Node-Red : nodered
- Grafana : grafana

Un compte **etudiant** avec pour mot de passe **etudiant** est disponible pour se connecter à la VM. Un serveur OpenSSH pour l'accès au terminal à distance a été déployé et est accessible sur le port **2222** de la machine hôte (redirection de port).

2. Configuration du cœur de réseau Chirpstack

a. Introduction

ChirpStack est une solution open-source pour gérer un cœur de réseau LoRaWAN et les applicatifs associés. Elle possède une interface Web intuitive pour la gestion du réseau, des passerelles, des objets et des applications.

- Ouvrir l'interface web : <http://localhost:8080>
- S'Identifier avec l'utilisateur **admin** et le mot de passe **admin**

b. Enregistrement d'un network server

Le network server prend en charge l'authentification et fait le lien entre notre serveur applicatif et nos devices.

- Dans l'onglet **Network-server**, renseigner le serveur déployé sur la VM
 - Nom d'hôte : chirpstack-network-server
 - Port : 8000

c. Création d'un Service profile

Le service profile est une section permettant la création de profils définissant les paramètres d'utilisation du network server.

- Dans l'onglet **Service-profiles**, ajouter un nouveau profil **Default** en vous aidant de [la documentation suivante](#) (RP002-1.0.3 LoRaWAN® Regional Parameters) et plus particulièrement des pages 21 et 26 pour définir le data rate minimum et maximum.

d. Enregistrement d'une gateway

- Dans l'onglet **Gateways**, ajouter la gateway présente pour la formation
 - ID : 3133303725003A00

e. Création d'un Device profile

Le device profile permet de définir les caractéristiques LoRaWAN du dispositif.

- Dans l'onglet **Device-profile**, configurer un device profile **M5LoRaKit**
 - À l'aide de [la documentation de la bibliothèque utilisée, LMIC](#), spécifier la version de la MAC LoRaWAN
 - Configurer une activation OTAA

f. Création d'une Application

Une application est un ensemble d'objets répondant à un cas d'usage.

- Dans l'onglet **Applications**, ajouter une application **tp-iot**

g. Création d'un Objet

Un **Device** correspond à un objet LoRaWAN. Il s'agit ici du M5LoRaKit. Pour chaque **Device**, il est possible de voir les données applicatives ainsi que les trames réseaux échangées.

Le **DEVEUI** est l'identifiant unique du nœud. Pour éviter des conflits, les kit sont numérotés (en hexadécimal). Le **DEVEUI** en hexadécimal sera 01 02 03 04 05 06 07 **XX, XX** étant le numéro du kit.

La clé d'application **APPKEY**, sur 16 octets, dépend elle aussi de chaque device. Dans ce TP, l'**APPKEY** en hexadécimal sera : 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 **XX, XX** étant le numéro du kit.

Provisionner un objet avec ces identifiants dans l'application.

3. Programmation du M5LoRaKit

a. Module LoRa

Pour ce TP nous allons utiliser [le module LoRa 868](#) qui est le module bleu stacké sous le module Core2. Ce module M5Stack embarque un [module Ra-01H](#) contenant le transceiver radio **SX1276**.

b. Bibliothèque LMIC

LMIC (LoRa MAC In C) est une bibliothèque en C permettant à un objet de se connecter à un réseau LoRaWAN. Celle-ci a été incluse dans le projet de base.

Une configuration de la bibliothèque est nécessaire. Pour cela, des `build_flags` ont été rajoutés dans le fichier ***platformio.ini*** :

- Identifier le define qui permet d'effectuer la configuration par le biais des `build_flags`
 - <https://github.com/mcci-catena/arduino-lmic#platformio>
- Identifier le define qui configure la région
 - <https://github.com/mcci-catena/arduino-lmic#selecting-the-lorawan-region-configuration>
- Identifier le define qui configure le module radio
 - <https://github.com/mcci-catena/arduino-lmic#selecting-the-lorawan-region-configuration>

Ouvrir maintenant le ***main.cpp*** et identifier le code configurant le pin mapping. Celui-ci définit les broches du microcontrôleur connectées au module radio LoRa. Il s'établit en croisant les schémas du module LoRa et Bus M5Core2 :

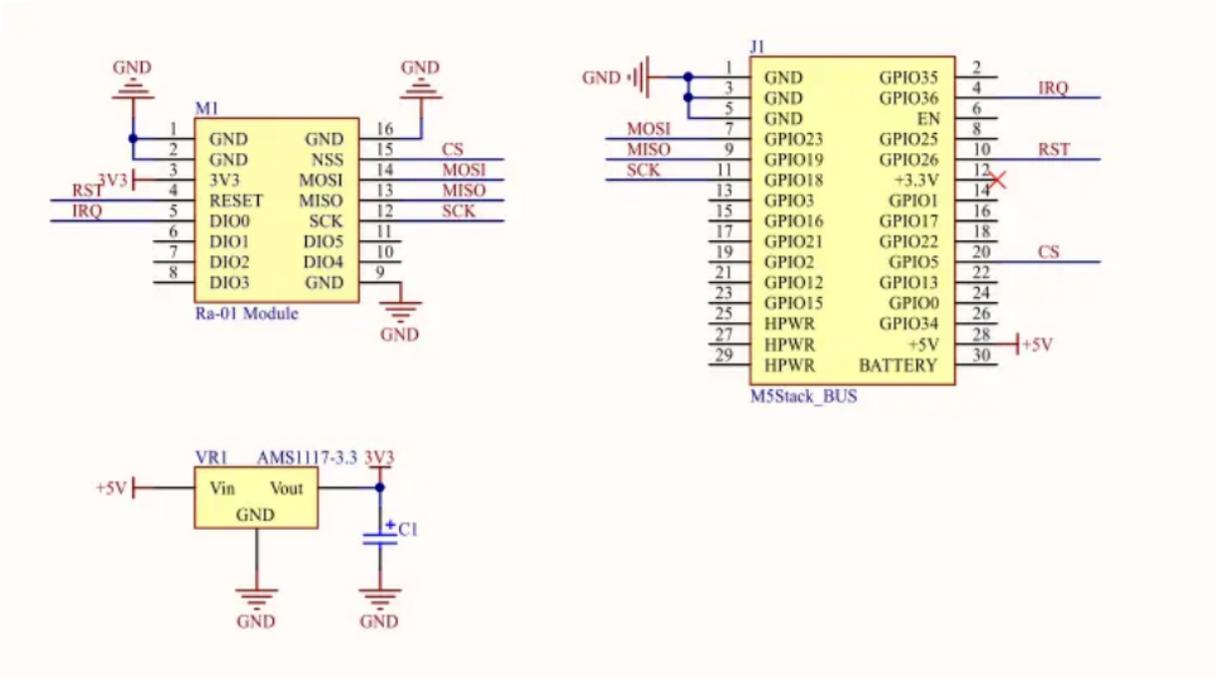


Schéma électrique module LoRa

M5Core2 Bus									
M5Stack Bus									
GND			GND	1	2	ADC1	GPIO35	GPIO35	ADC
GND			GND	3	4	ADC2	GPIO36	GPIO36	ADC
GND			GND	5	6	RESET	EN	EN	RST
MOSI	GPIO23	GPIO23	MOSI	7	8	AUDIO_L	GPIO25	GPIO25	DAC
MISO	GPIO38	GPIO19	MISO	9	10	AUDIO_R	GPIO26	GPIO26	DAC
SCK	GPIO18	GPIO18	SCK	11	12	3.3V			3.3V
RXD0	GPIO3	GPIO3	RXD1	13	14	TXD1	GPIO1	GPIO1	TXD0
RXD2	GPIO13	GPIO16	RXD2	15	16	TXD2	GPIO17	GPIO14	TXD2
intSDA	GPIO21	GPIO21	SDA	17	18	SCL	GPIO22	GPIO22	intSCL
PA_SDA	GPIO32	GPIO2	IO6	19	20	IO7	GPIO5	GPIO33	PA_SCL
GPIO	GPIO27	GPIO12	IIS_SCLK	21	22	IIS_WS	GPIO13	GPIO19	GPIO
I2S_OUT	GPIO2	GPIO15	IIS_OUT	23	24	IIS_MCLK	GPIO0	GPIO0	2S_LRCLK
NC			HPWR	25	26	IIS_EN	GPIO34	GPIO34	PDM_DAT
NC			HPWR	27	28	5V			5V
NC			HPWR	29	30	BATTERY			BAT

Schéma de correspondance des broches du bus M5Core2

En dernier, configurer les identifiants pour une activation OTAA

- DEVEUI
- APPKEY

c. Join et Uplink

Compiler et téléverser le programme

- A l'aide du moniteur série, vérifier le Join et identifier les clés de chiffrement utilisées pour la session (AppSkey et NwkSKey).
- Sur la page du device configuré dans Chirpstack, observer les trames échangées

Vous remarquez que les messages reçus de l'objet ne sont pas compréhensibles. En effet, ils sont codés en base64 (codage de l'information utilisant 64 caractères).

Son intérêt réside principalement pour la représentation de données binaires sous un format texte relativement compact.

- Décoder les messages observés sur Chirpstack via [un décodeur base64 vers ASCII en ligne](#)
- Identifier et personnaliser dans le code le message envoyé au réseau
- Confirmer la modification des messages reçus

4. Envoi et visualisation des données ENVIII

a. Envoi

- Intégrer l'acquisition des données du module ENVIII vue dans le TP1
- Remplacer le message envoyé en LoRa par les dernières données relevées

b. Réception

- Décoder les messages reçus via [un décodeur base64 vers hexadécimal en ligne](#)
- Assembler les octets pour reconstruire et valider les données

c. Décodage des données

Un décodeur peut être directement intégré dans le **Device-profile M5LoRaKit**

- Ouvrir l'onglet **Codec** et sélectionner **Custom JavaScript codec functions**
- Compléter la fonction JavaScript **Decode**

5. Envoi d'un Downlink

Il est possible d'envoyer un message du réseau à l'objet.

- Sur la page du device dans Chirpstack, configurer l'envoi d'un downlink
 - Le message doit être codé en base64
- Observer le nombre d'octets reçus par l'objet dans le moniteur série
- Envoyer sur le port série le contenu des octets reçus et les observer dans le moniteur série