Université Bretagne Sud

Wireless Attack Network Testbed for Embedded Devices

Livrable 3

Projet WANTED

GHARBI Meysoun

Encadrant : Philippe Tanguy

2025/2026

Table des matières

Table des figures

1 Introduction générale

Les réseaux LoRa/LoRaWAN sont aujourd'hui massivement déployés dans des domaines critiques tels que les smart cities, l'industrie, l'agriculture connectée ou encore la gestion d'infrastructures. Bien que LoRaWAN intègre des mécanismes de sécurité, de nombreuses vulnérabilités subsistent, notamment en raison de mauvaises configurations ou d'implémentations imparfaites.

Dans ce contexte, le projet **WANTED (Wireless Attack Network Testbed for Embedded Devices)** vise à fournir une plateforme expérimentale permettant d'orchestrer, d'exécuter et d'analyser des attaques sans fil sur des dispositifs embarqués, dans un environnement de laboratoire isolé.

Ce projet a pour but d'expérimenter différentes attaques sur un réseau LoRaWAN, notamment :

— **Sniffing** : interception des paquets LoRaWAN transmis sur le réseau.

— **Brouillage (jamming)** : perturbation volontaire du signal radio afin d'empêcher la communication entre les nœuds et le réseau.

— **Rejeu de paquets (replay attack)** : retransmission de paquets précédemment interceptés dans le but de tromper le réseau ou de provoquer des états incohérents.

— **Usurpation d'identité** : imitation d'un nœud légitime afin d'envoyer des données falsifiées ou malveillantes.

Au-delà de la reproduction d'attaques connues, ce travail vise également à analyser la littérature scientifique existante sur les attaques LoRa/LoRaWAN afin de tenter de reproduire ou d'adapter d'autres nouvelles attaques décrites dans des travaux de recherche récents. Cette approche permet d'inscrire le projet dans une démarche scientifique liant l'état de l'art aux expérimentations pratiques.

2 Rappel du cahier des charges

2.1 Objectifs du projet

L'objectif principal du projet est l'intégration d'attaques LoRa/LoRaWAN au sein de la plateforme **WANTED** afin de disposer d'un banc d'essai reproductible permettant :

— l'orchestration d'attaques sans fil,

— la génération de traces et de métriques exploitables,

— l'analyse de l'impact des attaques sur le réseau et les dispositifs.

2.2 Exigences fonctionnelles

Les exigences fonctionnelles définies pour le projet sont les suivantes :

— lancer, arrêter et superviser des attaques ,
— intégrer des attaques existantes dans WANTED,
— implémenter au moins deux attaques LoRa/LoRaWAN et essayer de les intégrer,
— générer des logs structurés au format JSON,
— fournir une documentation minimale.

2.3 Contraintes techniques et réglementaires

Le projet est soumis à plusieurs contraintes techniques et réglementaires :
— respect de la réglementation radio EU868,
— exécution des attaques dans un environnement isolé,

3 Démarche scientifique et technique

3.1 Analyse du problème et modèle de menace

Une analyse préalable du système LoRa/LoRaWAN a permis d'identifier les actifs critiques à protéger, notamment le *network server*, le processus de *join* OTAA, l'intégrité des données et les passerelles. Le modèle de menace établi dans les livrables précédents se concentre exclusivement sur des attaques sans fil, en excluant volontairement les attaques physiques.

Les profils d'attaquants considérés incluent l'attaquant opportuniste, le concurrentmalveillant et le saboteur ciblé. Les vecteurs d'attaque prioritaires incluent le sniffing, le rejeu de paquets, le brouillage radio (*jamming*) et l'usurpation d'identité.

Dans le cadre de ce projet, une première attaque issue de la littérature scientifique a été étudiée spécifiquement : **l'attaque par déni de service (DoS) via l'exploitation du DevNonce dans la procédure de jointure OTAA (Over-The-Air Activation)**. Cette attaque, documentée dans plusieurs travaux académiques portant sur la sécurité de LoRaWAN, cible le mécanisme de prévention des rejeus (replay attack protection) de la procédure OTAA. Elle permet à un attaquant d'inonder le serveur réseau d'un grand nombre de requêtes de jointure forgées, utilisant des valeurs DevNonce distinctes mais usurpant l'identifiant DevEUI d'un dispositif légitime.

Et une 2ème attaque :**Attaque par Déni de Service Énergétique (Energy Depletion Attack) via l'exploitation du paramètre RX2 et de la fenêtre de réception**. Cette attaque, exploite le mécanisme de réception bidirectionnelle de LoRaWAN. L'attaquant envoie de fausses trame "downlink" adressées à une victime spécifique pendant sa fenêtre de réception. Le dispositif cible, pensant recevoir un message légitime du serveur, active son récepteur et consomme de l'énergie pour le recevoir et le traiter.

3.2 Choix techniques et justification

Le choix de **LoRa/LoRaWAN** comme technologie cible s'explique par sa large adoption et par l'asymétrie importante entre le coût d'attaque et l'impact potentiel

La plateforme **ChirpStack** a été retenue comme *network server* en raison de son caractère open-source, de sa modularité et de sa large adoption dans la communauté académique.

Le choix des attaques issues de la littérature scientiques des attaques :

3.2.1 Faisabilité technique et matérielle

Cette attaque présente des **exigences matérielles minimales**, compatibles avec les contraintes d'un laboratoire pédagogique :

- Un serveur LoRaWAN local (`ChirpStack` sur un PC)
- Une passerelle LoRaWAN (Raspberry Pi 3 model B+ IC880A)
- Deux nœuds LoRa (un légitime (LoPy4), l'autre d'attaquant(ARDUINO))

3.2.2 Simplicité de mise en œuvre et reproductibilité

- **Version semi-simulée** : Un simple script Python peut inonder le serveur de requêtes de jointure forgées via le protocole UDP, sans même nécessiter d'émission radio. Cela permet de valider rapidement le principe de l'attaque.
- **Version "over-the-air" complète** : Nécessite uniquement de programmer un nœud attaquant pour émettre des trames `Join-Request` forgées, démontrant l'exploitation réelle de la vulnérabilité protocolaire.

Bien que légèrement plus exigeante que la précédente, cette attaque nécessite du matériel courant en laboratoire d'électronique :

3.2.3 Matériel accessible et pédagogiquement riche

- Une passerelle LoRaWAN
- Deux nœuds LoRa (légitime et attaquant)
- Un analyseur de consommation (multimètre ou oscilloscope, équipements standards)

Cette configuration permet d'introduire des **mesures quantitatives** (consommation énergétique) dans l'analyse de sécurité.

3.2.4 Défi technique réaliste et formateur

L'attaque présente un **défi de synchronisation temporelle** particulièrement instructif :

- **Complexité contrôlable** : Ce défi peut être résolu de plusieurs façons adaptées au contexte pédagogique :
  - Utilisation d'une **horloge commune** pour les deux nœuds (attaquant et victime)
  - Implémentation d'un **déclenchement par détection radio** de l'*uplink*
  - Synchronisation logicielle via le réseau local

3.3 Architecture du système

L'architecture globale du système repose sur les éléments suivants :

- *End-device* LoRa : un noeud légitime : LoPy4 et un noeud malveillant : Arduino dragino
- Passerelle LoRaWAN fournie par l'encadrant, incluant : Raspberry Pi 3 Model B + Module concentrateur LoRa IC880A
- Serveur réseau ChirpStack préconfiguré sur la passerelle.

3.4 Mise en œuvre technique

La mise en œuvre du projet s'est déroulée en plusieurs étapes :

— prise en main de l'architecture existante de WANTED,

— mise en place d'un environnement isolé incluant ChirpStack, une passerelle et des *end-devices* et sa configuration,

— exécution de scénarios d'attaque contrôlés (sniffing, brouillage simple).

Cette phase a présenté plusieurs défis, dont une méprise initiale sur la documentation à suivre, conduisant à explorer des configurations serveur non pertinentes (WalT, NFS) avant d'être réorienté par l'encadrant. Sur le bon chemin, des problèmes techniques sont survenus concernant la configuration de ChirpStack (identification de passerelle) et la synchronisation des équipements, problèmes qui ont été résolus par une étude documentaire approfondie et des tests itératifs.

4 Réalisation et Mise en œuvre du Banc de Test

Cette section détaille les étapes techniques pour établir la communication LoRaWAN entre le nœud capteur et le serveur réseau.

4.1 Configuration de la Passerelle (Gateway)

La passerelle est le cœur du réseau, faisant le pont entre la radio LoRa et l'IP.

— **Identification du matériel** : Nous avons utilisé un Raspberry Pi 3 couplé à un concentrateur IC880A.

— **Extraction de l'ID Gateway** : L'identifiant unique (EUI-64) est crucial pour ChirpStack. Nous l'avons généré à partir de l'adresse MAC de l'interface Ethernet `b8:27:eb:d5:b6:0e`.

— **Calcul de l'ID** : En insérant `FFFE` au centre de l'adresse MAC, nous avons obtenu le **Gateway ID** : `b827ebfffed5b60e`.

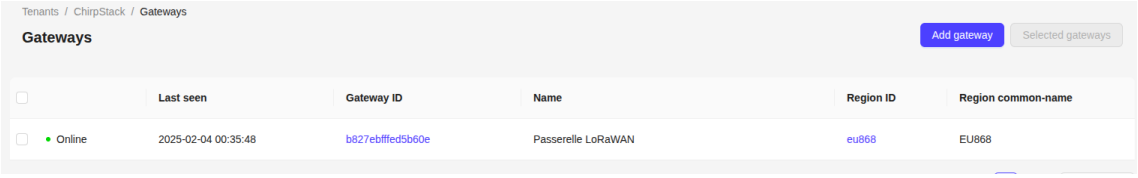

FIGURE 1 – gateway connectée

4.2 Configuration du Nœud Capteur (LoPy4)

Le LoPy4 agit comme le dispositif final (End-device) que nous allons ensuite attaquer.

— **Connexion au poste de travail** : Le module a été connecté via USB à un PC Lenovo. Il a été identifié sur le port série `/dev/ttyACM0` sous Linux.

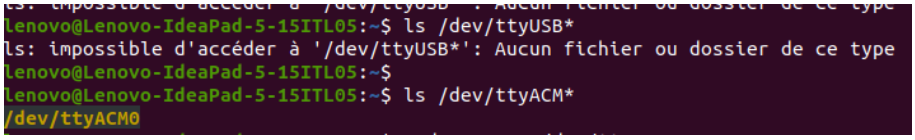

FIGURE 2 – /dev/ttyACM0

=> Afin de vérifier la bonne détection matérielle du nœud par le système d'exploitation Linux, une interrogation des périphériques série a été effectuée. Contrairement aux standards habituels (ttyUSB), le module a été identifié sur le port /dev/ttyACM0. Cette étape de diagnostic a été cruciale pour configurer l'environnement de développement VS Code et l'extension Pymakr.

— **Extraction du DevEUI** : Pour enregistrer le nœud dans ChirpStack, nous avons dû extraire son identifiant matériel unique via un script MicroPython utilisant `lora.mac()`.

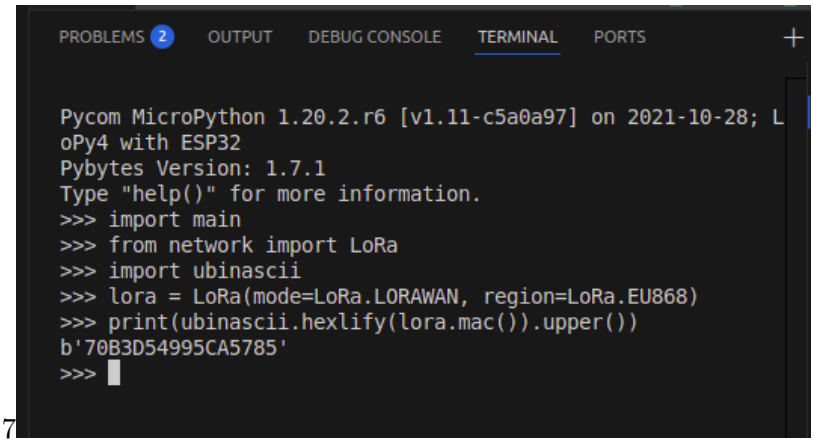

FIGURE 3 – DevEUI du loPy

Last seen	DevEUI	Name	Device profile	Battery
2025-02-04 03:44:07	70b3d54995ca5785	LoPy4	Pycom	

FIGURE 4 – End device configuré dans ChirpSTack

Une fois le DevEUI récupéré, nous avons implémenté le script de connexion utilisant la méthode d'activation OTAA. L'élément central de ce script est l'AppKey, générée sur le serveur ChirpStack et reportée dans le code pour permettre le chiffrement des échanges

```
# create an OTAA authentication parameters, change them to the provided creden
app_eui = ubinascii.unhexlify('0000000000000000') #app key non trouvée
app_key = ubinascii.unhexlify('bae64ce2ce34fd94c24f8ec55c233ecb')
#uncomment to use LoRaWAN application provided dev_eui
dev_eui = ubinascii.unhexlify('70B3D54995CA5785')
```

FIGURE 5 – AppKey et DevEUI configurés par ChirpStack

4.3 Validation de la connectivité LoRaWAN

Apès les changements des AppKey et DevEUI dans le code, nous avons supprimé le contenu du mémoire flash du LoPy et flashé le nouveau code.

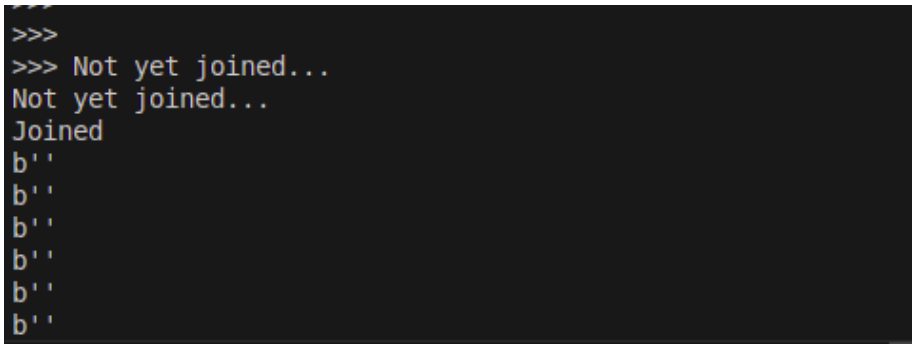

FIGURE 6 – End device joined

Dans le terminal de pymakr, nous voyons le message "Joined" qui signifie que le LoPy4 a reçu le JoinAccept du serveur et qu'il est maintenant officiellement sur le réseau.

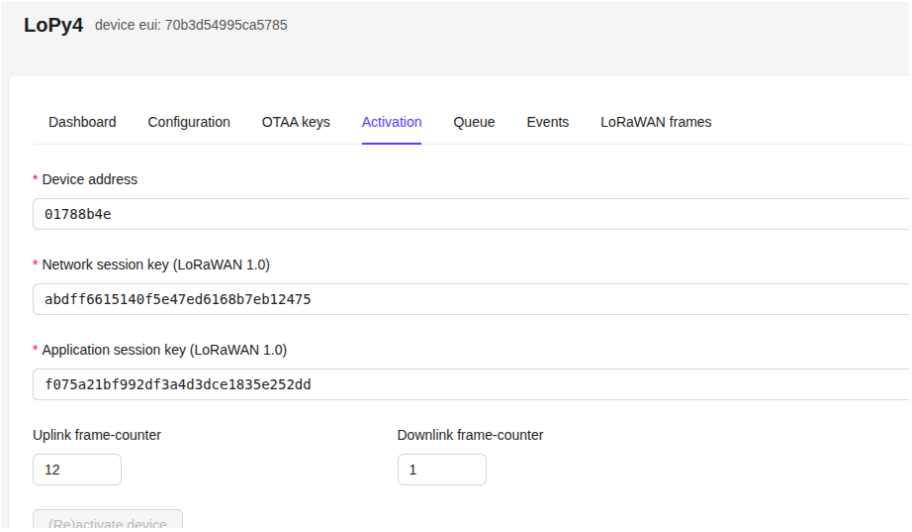

FIGURE 7 – End device activé

Dans l'onglet activation de ChirpStack, nous voyons que le end device est activé et il a maintenant une adresse dynamique (Device address) et des clés de session (Network session key), ce qui prouve que l'activation OTAA est terminée.

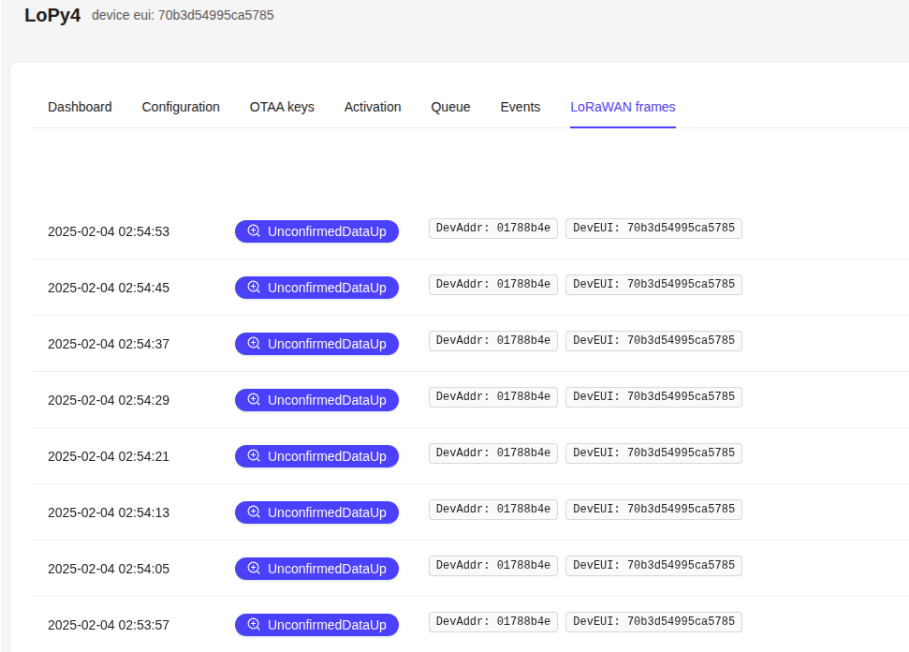

FIGURE 8 – End device joined

Et dans l'onglet LoRaWAN frames du device, nous voyons des messages UnconfirmedDataUp. Ce sont les données du capteur (Humidité) arrivent sur le serveur.

4.4 Difficultés Rencontrées et Solutions

— **Visibilité du périphérique USB** : Le port série n'était pas initialement visible avec `ls /dev/ttyUSB*`. La solution a été d'identifier que le module utilisait le pilote ACM (`/dev/ttyACM0`) et de configurer l'extension Pymakr dans VS Code.

— **Accès à la console (REPL)** : Lors de la phase de test, nous avons identifié que le module LoPy4 conservait en mémoire flash une ancienne version du script main.py. Cette persistance empêchait la prise en compte des nouveaux identifiants (AppKey et DevEUI), causant un échec systématique de l'authentification sur le serveur ChirpStack en saturant la console avec des messages "Not yet joined", empêchant la saisie de commandes ainsi qu'un échec de la procédure Join(OTAA) Le script pré-installé sur le LoPy4 saturait la console avec des messages "Not yet joined", empêchant la saisie de commandes

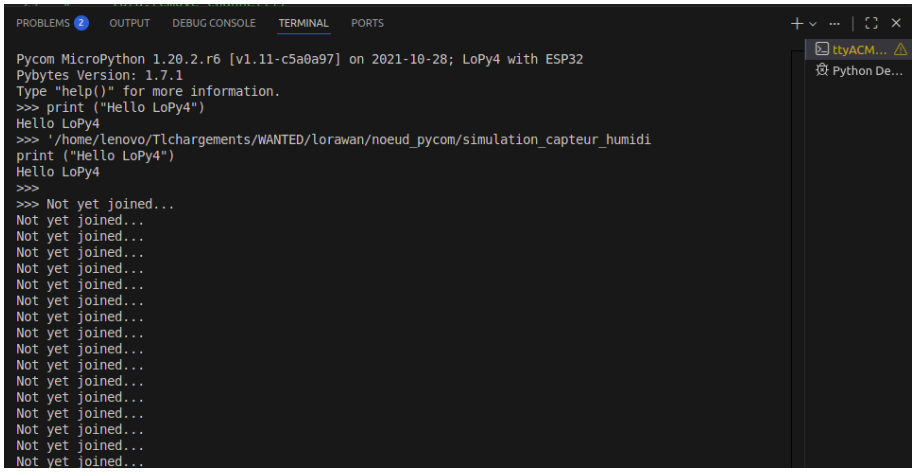

FIGURE 9 – LoPy saturait la console

Pour résoudre ce conflit, nous avons formaté la partition flash en utilisant la commande `os.fsformat('/flash')` , flashé le nouveau code qui contient les bons clés et redémarré le matériel

— **Échec de la procédure Join (OTAA)** : Bien que les trames `"Join Request"` soient visibles dans ChirpStack, l'activation restait bloquée. Cela a permis de souligner l'importance de la concordance parfaite entre le `DevEUI`, le `JoinEUI` (`AppEUI`) et l'`AppKey`.

— **Anomalie d'horodatage (Infrastructure)** : Une fois la connexion établie (statut `"Joined"`), nous avons constaté que les trames reçues sur ChirpStack affichaient une date erronée (février 2025). Le diagnostic a permis d'isoler ce problème au niveau de la passerelle LoRaWAN : celle-ci n'est pas synchronisée via le protocole NTP (*Network Time Protocol*). Bien que ce décalage temporel n'entrave pas la transmission des données en temps réel, il souligne un défaut de maintenance de l'infrastructure réseau utilisée.

4.5 Paramètres de Sécurité (Clés OTAA)

Pour finaliser la connexion, les paramètres suivants ont été configurés :

— **JoinEUI** : Fixé à `0000000000000000` par convention pour ce labo.

— **AppKey** : Clé de 128 bits générée aléatoirement sur ChirpStack et copiée dans le firmware du LoPy4 pour le chiffrement AES.

4.6 Synthèse de l'étape de mise en œuvre

La mise en œuvre de la communication entre le nœud légitime (LoPy4) et la passerelle (Raspberry Pi 3) a constitué la phase la plus délicate et la plus chronophage de ce projet. Cette étape a nécessité une compréhension profonde des interactions entre les couches matérielles (ports série /dev/ttyACM0) et les protocoles réseau (OTAA, gestion des clés AES-128).

Toutefois, ce temps investi a été déterminant : une fois la liaison établie et sécurisée, la reproduction des attaques (sniffing, brouillage) devient une procédure beaucoup plus directe. L'essentiel de ce travail réside dans la maîtrise de l'infrastructure LoRaWAN ; comprendre comment les trames sont générées et validées par le serveur ChirpStack permet désormais d'identifier précisément les vecteurs de vulnérabilité.

5 Configuration technique du nœud attaquant (Arduino)

Pour réaliser les attaques, nous avons utilisé un Arduino équipé d'un module radio LoRa, piloté par la bibliothèque MCCI LoRaWAN LMIC. Cette bibliothèque est critique car elle permet un contrôle bas niveau de la radio.

5.1 Préparation de l'environnement

— Configuration du fichier `lmic_project_config.h` pour forcer la fréquence sur EU868, correspondant à la zone géographique de notre passerelle Raspberry Pi.

— La communication entre l'Arduino et la puce radio (RFM95) avec un câblage spécifique (NSS sur pin 10, RST sur 9, et DIO sur 2, 6, 7) est déjà fournie

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};
```

FIGURE 10 – mapping

5.2 Configuration des formats de clés (Endianness)

Une difficulté technique majeure en LoRaWAN est l'ordre des octets. Pour que l'Arduino puisse "dialoguer" avec le serveur ChirpStack, il faut convertir les clés :

— **Little Endian (LSB)** pour le `DevEUI` et le `JoinEUI`.

— **Big Endian (MSB)** pour l'`AppKey`.

cette étape est déjà faite

```
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]={0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11};
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
static const u1_t PROGMEM APPKEY[16] = {0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11,
 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

FIGURE 11 – Configuration des formats de clés

5.3 Utilisation du nœud pour l'attaque

Une fois ce nœud Arduino Dragino configuré, il nous sert de base pour :

— **Le Sniffing** : En modifiant le code pour désactiver le filtrage des adresses, l'Arduino peut lire toutes les trames passant à sa portée.

— **Le Brouillage** : En modifiant la fonction `do_send()`, nous pouvons forcer l'Arduino à émettre en continu sur le canal utilisé par le LoPy4, créant ainsi un déni de service (DoS).

6 Objectifs et indicateurs de réussite des attaques (Perspectives immédiates)

L'implémentation prochaine des scénarios d'attaque sur ce banc de test aura pour but de démontrer la faisabilité de l'orchestration d'attaques LoRaWAN via le framework WANTED.

Les outils de capture et les scripts Arduino configurés précédemment permettront de générer des logs détaillés. Nous prévoyons d'extraire des métriques exploitables pour quantifier l'efficacité de l'attaque, notamment :

— Le nombre de paquets capturés lors des phases de **sniffing**.

— Le taux de paquets perdus (**Packet Loss**) côté passerelle lors des phases de **brouillage**.

— Le succès des tentatives de **rejeu de trames**.

Les résultats attendus devront confirmer les hypothèses formulées lors de l'étude théorique

7 Conclusion et bilan du projet

7.1 Atteinte des objectifs

Au regard des objectifs définis dans le cahier des charges, le projet a quasi permis d'atteindre une grande partie des attendus. L'implémentation d'attaques LoRa/LoRaWAN existantes ainsi que la recherche des nouvelles attaques pour les intégrer dans WANTED ont été réalisées avec succès.

7.2 Retards et problèmes rencontrés

Des retards ont été observés, principalement dus à la complexité de l'environnement radio. Ces difficultés ont néanmoins permis d'approfondir la compréhension des contraintes réelles liées aux attaques sans fil.

7.3 Analyse du diagramme de Gantt

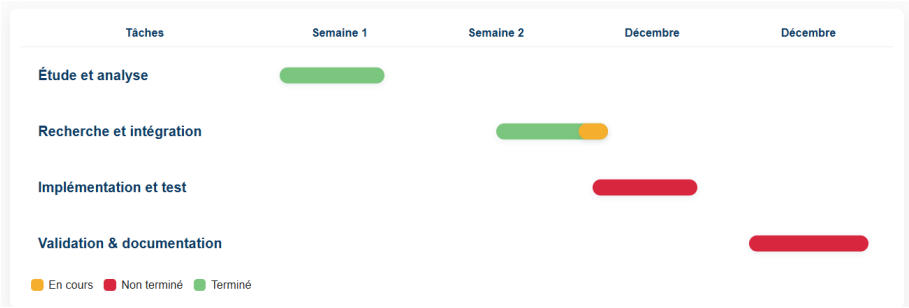

FIGURE 12 – diagramme de Gantt

L'analyse du diagramme de Gantt met en évidence des écarts entre les temps estimés et le temps réellement consacré à certaines tâches, notamment lors des phases d'intégration et de validation. Ces écarts s'expliquent par des itérations supplémentaires nécessaires pour stabiliser l'environnement expérimental.

8 Perspectives

Le travail réalisé a permis d'établir une base solide pour l'étude de la sécurité LoRaWAN. À court terme, les perspectives du projet incluent l'implémentation d'attaques supplémentaires et l'automatisation complète des scénarios. À long terme, l'intégration de mécanismes de détection et de contre-mesures directement au sein de la plateforme WANTED.