Documentation projet LoRa

Adrien LEBOURGEOIS

01/04/2020

Table des matières

1 Préambule

La documentation concerne un projet existant et récupéré en l'état. Les moyens de parvenir à cet état ne seront pas explicités. Seules les modifications et apports le seront.

Afin de faciliter la configuration, il est recommandé de configurer votre serveur DCHP pour qu'il attribue des IP fixes aux équipements.

2 Identifiants et mots de passe

Système	Accès	Identifiant	Mot de passe
RaspberryPi	SSH via 10.3.141.1	pi	raspberry
RaspberryPi	web via raspberrypi.local	admin	secret
RaspberryPi - Grafana	web via son adresse :3000	admin	admin
Passerelle Kerlink	web via son adresse	spn	iot_vision29
VM OVH	SSH via 51.83.79.109	root	gm0Gw80f
VM OVH	SSH/FTP via 51.83.79.109	lora	iot_vision29
VM OVH	SSH/FTP via 51.83.79.109	pycom	iot_pycom29
VM OVH - Grafana	web via 51.83.79.109 :3000	admin	iot_vision29
OpenWeathear	API	iot_vision	iotVisionWeather

OpenWeather API key : 98b0cfd0bfd462ed697ae60003716368

Mots de passe du VPN :

Type	Nom	Mot de passe
CA	projet	projetM1_2020
serveur	serverLora	serveurLora
client Kerlink	kerlink	kerlinkLora
p12 client	kerlink	kerlink
admin Kerlink	adminKerlink	kerlinkLoraAdmin
p12 admin	adminKerlink	adminExport

Première partie

RaspberryPi

1 Etat existant

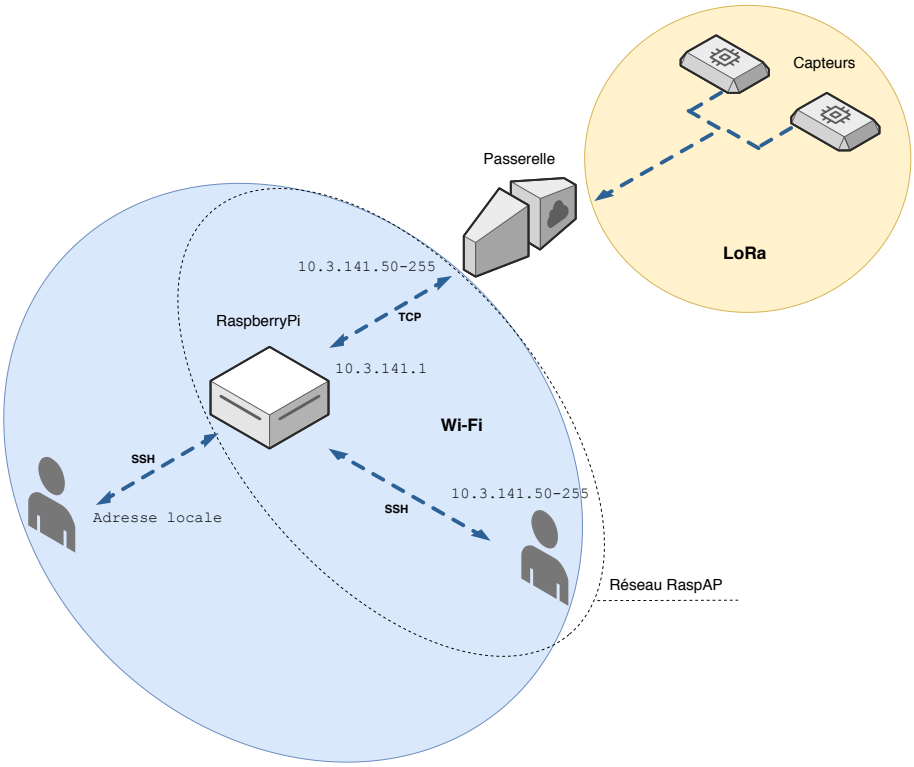

FIGURE 1 – Réseau existant

Sur la RaspberryPi est installé RaspAP [1] qui lui permet de créer un réseau privé auquel est connectée la passerelle. L'accès à l'interface de configuration de la passerelle se fait via ce réseau ; il est possible d'utiliser la commande `nmap -sP 10.3.141.0/24` pour trouver son adresse. Il est possible d'administrer le point d'accès crée par RaspAP via son interface graphique accessible via l'adresse `raspberrypi.local` lorsque l'on est connecté au réseau RaspAP. L'accès à la RaspberryPi via SSH peut se faire via le réseau RaspAP ou via le réseau local de l'utilisateur si la Raspberry est connectée au réseau local. Là encore `nmap` peut servir à trouver les adresses.

2 Passage du réseau RaspAP au réseau local

La carte RaspberryPi n'étant pas conçue pour servir de point d'accès Wi-Fi, sa température atteignait les 80°C... Le point d'accès a donc été désactivé et la communication se fait maintenant en Ethernet via le réseau local.
Les configurations à effectuer sont les suivantes :

2.1 Passerelle

Changer l'adresse "Remote FTP" via Configuration/Interfaces/Remote FTP :

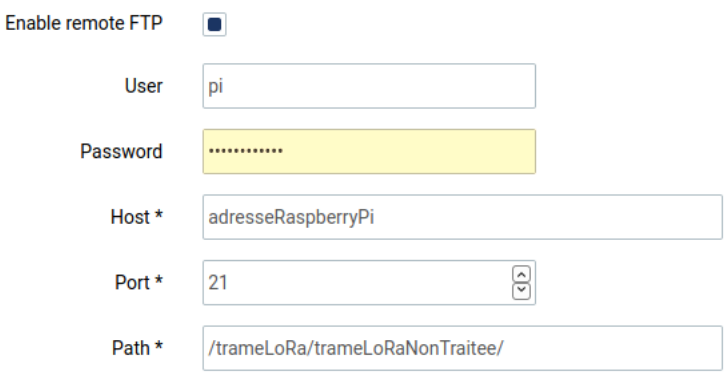

FIGURE 2 – Remote FTP

Ne pas oublier d'enregistrer.

2.2 RaspberryPi

Changez l'adresse à laquelle Grafana sera diponible dans le fichier `grafana.ini` :
`sudo nano /etc/grafana/grafana.ini`
`sudo service grafana-server restart`

Désactivez le démarrage automatique de RaspAP :
`sudo systemctl disable raspap.service`
`enable` pour activer.

Créez un fichier `/etc/modprobe.d/raspi-blacklist.conf` contenant :
```
#wifi
blacklist brcmfmac
blacklist brcmutil
```
pour désactiver le point d'accès wifi.

Deuxième partie
Machine virtuelle OVH

1 Connexion FTP

Un serveur FTP ProFTPd [2] est installé sur la machine. Pour des raisons de sécurité, l'accès à la machine OVH via FTP ne peut se faire avec l'utilisateur `root`, il faut passer par l'utilisateur `lora`. Dans le fichier de configuration `/etc/proftpd/proftpd.conf` ont été ajoutées les lignes suivantes :

```
<Global>
        RootLogin off
        UseFtpUsers on
</Global>
```

Elles permettent de désactiver l'accès FTP à l'utilisateur `root` et d'utiliser le fichier `/etc/ftpusers` qui contient la liste des utilisateurs n'ayant pas accès au serveur FTP.

2 Mise en place du VPN

Le VPN est réaliser avec OpenVPN [3]. Référence supplémentaire : [4]

2.1 Génération des clés

Les clés sont générées avec Easy-RSA version 3 développé par OpenVPN. S'agissant de sécurité, il est important d'avoir une version à jour. Pour cela, clonez le dépôt GitHub Easy-RSA [5] :
`git clone https://github.com/OpenVPN/easy-rsa.git`

Se déplacer dans `easy-rsa/easyrsa3`, copier le fichier `vars.example` et l'appeler `vars` (`cp vars.example vars`. Ajoutez dans le fichier vars les lignes suivantes :

```
set_var EASYRSA_REQ_COUNTRY     "France"
set_var EASYRSA_REQ_PROVINCE    "Finistère"
set_var EASYRSA_REQ_CITY        "Brest"
set_var EASYRSA_REQ_ORG         "ISEN"
set_var EASYRSA_REQ_EMAIL       "iot.vision.projet@gmail.com"
set_var EASYRSA_REQ_OU          "ISEN_Brest"
```

Elle serviront à générer le certificat d'autorité.

Initialisez la PKI (Public Key Infrastruture)
`./easyrsa init-pki`

Produire le certificat d'autorité :
`./easyrsa build-ca`
Un nom et un mot de passe sont demandés. Le mot de passe servira pour la suite. Le certificat se trouve dans `/root/easy-rsa/easyrsa3/pki/ca.crt`

Générez les clés serveur :
`./easyrsa build-server-full` *`nomDeLaPaireDeClés`*
Un nouveau mot de passe est demandé. Le mot de passe du certificat d'autorité est aussi demandé.

Générez les clés client :
`./easyrsa build-client-full` *`nomDeLaPaireDeClés`*
Un nouveau mot de passe est demandé. Le mot de passe du certificat d'autorité est aussi demandé.
Générez de la même façon les clés client qui permettront à un administrateur d'accéder à l'interface d'administration de la passerelle via le VPN.

Générez les clés Diffie-Hellman, elles permettent l'initialisation de la connexion entre un client et le serveur (leur génération prend du temps) :
`./easyrsa gen-dh`

Exportez les clés client et le `ca.crt` dans une archive sécurisée de type `.p12` :
`./easyrsa export-p12` *`nomDeLaPaireDeClésAExporter`*
Un mot de passe pour l'archive est demandé. Le mot de passe de la paire de clés est aussi demandé.

Nom fichier	**Requis par**	**Secret**	**Emplacement**
ca.crt	serveur + clients	NON	/easyrsa3/pki/
ca.key	machine signant les clés (ici le serveur)	OUI	/easyrsa3/pki/private/
dh.pem	serveur	NON	/easyrsa3/pki/
server.crt	serveur	NON	/easyrsa3/pki/issued/
server.key	serveur	OUI	/easyrsa3/pki/private/
client1.crt	client1	NON	/easyrsa3/pki/issued/
client1.key	client1	OUI	/easyrsa3/pki/private/
client1.p12	client1	NON	/easyrsa3/pki/private/

2.2 Création des fichiers de configuration

Dans `/usr/share/doc/openvpn/examples/sample-config-files/` se trouvent des exemples de fichiers de configuration, copiez-les avant modification.

2.2.1 Serveur

Le fichier doit être nommé `server.conf` et placé dans `/etc/openvpn/`.
Contenu du ficher :

```
port 443
proto tcp
dev tun
#fichiers crées précédemment
ca /root/easy-rsa/easyrsa3/pki/ca.crt
cert /root/easy-rsa/easyrsa3/pki/issued/serverLoRa.crt
key /root/easy-rsa/easyrsa3/pki/private/serverLoRa.key # This file should be
# kept secret
dh /root/easy-rsa/easyrsa3/pki/dh.pem

server 10.8.0.0 255.255.255.0
client-to-client
keepalive 10 120
cipher AES-256-CBC
comp-lzo
persist-key
persist-tun
status server/log/openvpn-status.log
log
server/log/openvpn.log
verb 6
```

Le TCP est préféré à l'UDP car il assure la correction des erreurs dans les données transmises. Malgré la correction qui peut prendre du temps en plus de l'encapsulation VPN et au vu de la quantité de données à transmettre, la différence ne se fait pas sentir.
Les nom des clés et certificats sont bien-sûr à changer selon ce qui a été fait lors de la génération de ces derniers.

2.2.2 Client

Le fichier doit être nommé `client.conf`.//
Contenu du fichier :

```
client
dev tun
proto tcp
remote 51.83.79.109 443
resolv-retry infinite
nobind
persist-key
persist-tun
```

```
pkcs12 kerlink_nopass.p12
cipher AES-256-CBC
comp-lzo
verb 6
```

2.3 Démarrage et connexion au VPN

2.3.1 Serveur VPN

La mise en place du VPN va créer une nouvelle interface réseau virtuelle `tun0`. Pour démarrer le VPN :

```
service openvpn stop
service openvpn start
```

S'il y a un message `"systemd-tty-ask-password-agent...."` [7], saisissez `systemd-tty-ask-password-agent` puis le mot de passe de la clé serveur vous sera demandé. Vérifiez que l'interface `tun0` est présente avec la commande `ifconfig`. Vérifiez également avec la commande `netstat -tulnp` que OpenVPN écoute bien en TCP sur le port 443.
Par défaut le serveur VPN aura toujours l'adresse `10.8.0.1`.
L'arrêt puis le démarrage de OpenVPN doit être faite à chaque redémarrage de la VM car la saisie du mot de passe ne peut se faire automatiquement.

2.3.2 Passerelle Kerlink

Le plus simple pour récupérer l'archive `.p12` et les fichiers `client.conf` et `ca.crt` est de le faire par SCP [6]. Le fichier `ca.crt` est déjà présent dans l'archive `.p12` mais il faut tout de même le télécharger sur la passerelle Kerlink.

```
scp root@51.83.79.109:/root/easy-rsa/easyrsa3/pki/private/kerlink.p12
\emph{votreRépertoire}
```

La passerelle Kerlink ne permet pas de saisir de mot de passe pour l'archive `p12`, il faut donc le supprimer avec OpenSSL [8] :

```
openssl pkcs12 -in kerlink.p12 -nodes -out temp.pem
entrez le mot de passe de l'archive
openssl pkcs12 -export -in temp.pem -out kerlink_nopass.p12
appuyez sur entrée quand on vous demande le mot de passe,
pareil pour la confirmation
rm -f temp.pm
```

Vous pouvez tester la connexion au VPN en vous y connectant vous-même : `sudo openvpn client.conf` et en envoyant un ping au serveur VPN (`10.8.0.1`). Le mot de passe est celui de l'archive `p12`.

Importez les fichiers `ca.crt`, `client.conf` et l'archive `p12` via l'interface Administration/OpenVPN de la passerelle. Changer l'adresse "Remote FTP" via Configuration/Interfaces/Remote FTP :

FIGURE 3 – Remote FTP

Ne pas oublier d'enregistrer.
Consultez les logs dans Logs/Public pour vérifier la connexion. Vous devriez voir le message : "`Initialization Sequence Completed`".

2.3.3 Administration

L'administration de la passerelle via son interface web peut se faire via le VPN avec la paire de clés dédiée à cet effet. Déterminez l'adresse de la passerelle : `nmap -sP 10.8.0.0/24`, puis connectez-vous au VPN : `sudo openvpn adminKerlink.conf`. Le mot de passe est celui de l'archive `p12`. Vous pouvez vous assurer de votre connexion au VPN en vérifiant la présence de l'interface `tun0` avec la commande `ifconfig`. Connectez-vous à l'interface d'administration via son adresse VPN.

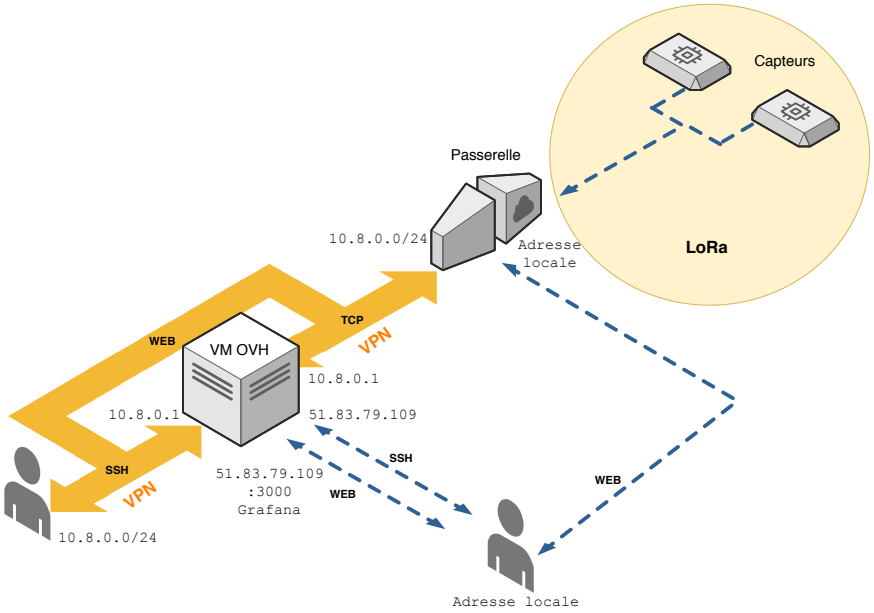

FIGURE 4 – Remote FTP

3 Sécurité

3.1 SSH

Actuellement Internet est plein de robots effectuant des attaques par force brute sur des services en tout genre, SSH n'y fait pas exception. Il existe plusieurs méthodes visant à parer ce genre d'attaque, la plus simple mais attention ni la plus sûre, ni la plus optimisée est l'utilisation de Fail2ban [9]. De plus, la version actuelle des dépôts Debian ne gère pas l'IPv6. Fail2ban fonctionne par analyse du fichier `/var/log/auth.log` via des expressions régulières.

```
apt install fail2ban
```

Modifiez le fichier `/etc/fail2ban/jail.d/defaults-debian.conf` :

```
[DEFAULT]
ignoreip = 127.0.0.1 # IP à ignorer, mettez la votre pour ne pas
# vous retrouver bloque
findtime = 7200 # duree sur laquelle les logs sont analyses
bantime = 172800 # temps de bannissement d'une IP
maxretry = 3 # nombre max de tentatives de connexion infructueuses

[sshd]
enabled = true
port = 22
logpath = /var/log/auth.log
maxretry = 3
```

Relancez le service : `service fail2ban restart` . Il est possible de consulter les logs de Fail2ban : `/var/log/fail2ban.log` .

Troisième partie
Transmissions LoRa

1 Caractéristiques de transmission

Dans l'interface d'administration de la passerelle Kerlink dans la section Fleet/Received Data une liste des paquets reçus est disponible :

ID	End-device ID	Gateway ID	Received time	Sequence number	Port	Radio ID	Channel	SNR	RSSI	Frequency	Modulation	Data Rate	Coding rate	Payload HEX
66196	A81758FFFE0352E0	7276FF0039030A46	04/03/2020 01:49:25 PM	2355	5	0	1	9.2 dB	-29 dBm	868.300 MHz	LoRa	SF7BW125	4/5	0100BF02320400690503070DBD
66195	A81758FFFE0352E0	7276FF0039030A46	04/03/2020 01:19:26 PM	2354	5	0	0	9.2 dB	-31 dBm	868.100 MHz	LoRa	SF7BW125	4/5	0100BF02320401570504070DBD
66194	1657344E78397A20	7276FF0039030A46	04/03/2020 12:52:18 PM	119	1	0	2	7.5 dB	-29 dBm	868.500 MHz	LoRa	SF7BW125	4/5	0B2298

FIGURE 5 – Transmissions reçues

ID Le numéro que la passerelle donne à la transmission reçue. Ne dépend pas du capteur émetteur.

End-device ID Le DevEUI (Device Extended Unique Identifier) du capteur émetteur.

Gateway ID Identifiant de la passerelle dans le cas où plusieurs seraient utilisées.

Received Time La date et l'heure de réception.

Sequence Number Numéro de transmission propre à chaque capteur. Incrémenté à chaque transmission.

Port NC

Radio ID La passerelle Kerlink possède deux bandes de fréquences de réception (en bleu) de 800 kHz chacune, *Radio 0* et *Radio 1*. Ces bandes peuvent être placées où on le souhaite entre 863 et 870 MHz.

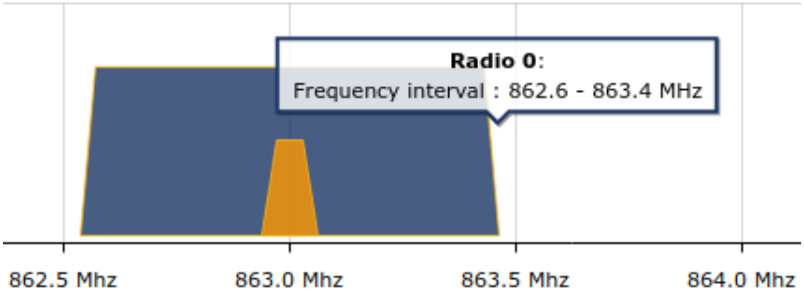

FIGURE 6 – Fréquences minimales de réception

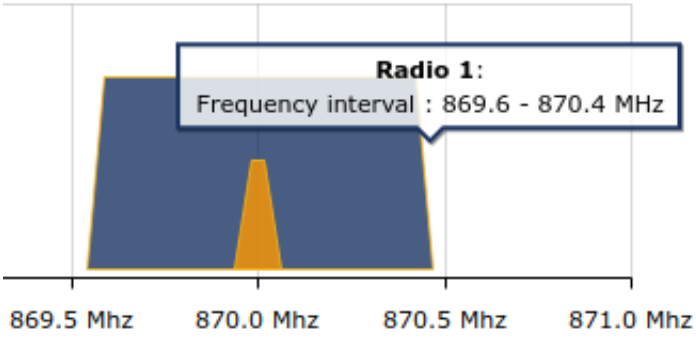

FIGURE 7 – Fréquences maximales de réception

Frequency Les bandes jaunes correspondent au bandes d'écoute utilisées, elles ont une largeur de 125 kHz. Même si la bande de réception peut descendre jusqu'à 862,6 MHz et monter jusqu'à 870 MHz, les bandes d'écoute sont au minimum centrées sur 863 MHz et au maximum sur 870 MHz. La passerelle peut écouter au maximum et en simultané sur 7 bandes de fréquences. Actuellement seules 3 sont utilisées et réparties sur une seule bande de réception. Elles sont identifiées sur la figure 5 par leur fréquence centrale. Le tout est paramétrable via l'interface d'administration dans la section Configuration/Rx configuration.

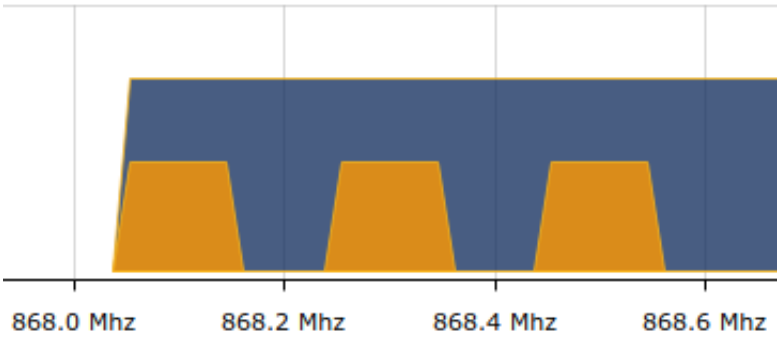

FIGURE 8 – Fréquences maximales de réception

Channel Les bandes d'écoute susmentionnées forment des canaux de communication. Numérotés de 0 à 2 sur la figure 5. Ces 3 canaux sont les canaux minimum dont une passerelle LoRa doit disposer.

SNR Le Signal Noise Ratio ou rapport signal sur bruit, est le rapport entre la puissance du signal reçu et la puissance du bruit de fond.

RSSI Le Received Signal Strength Indication ou indication d'intensité du signal reçu. Il est exprimé en dBm et exprime le rapport en la puissance du signal reçu et 1 mW. LoRa permet un RSSI minimum de -120 dBm.

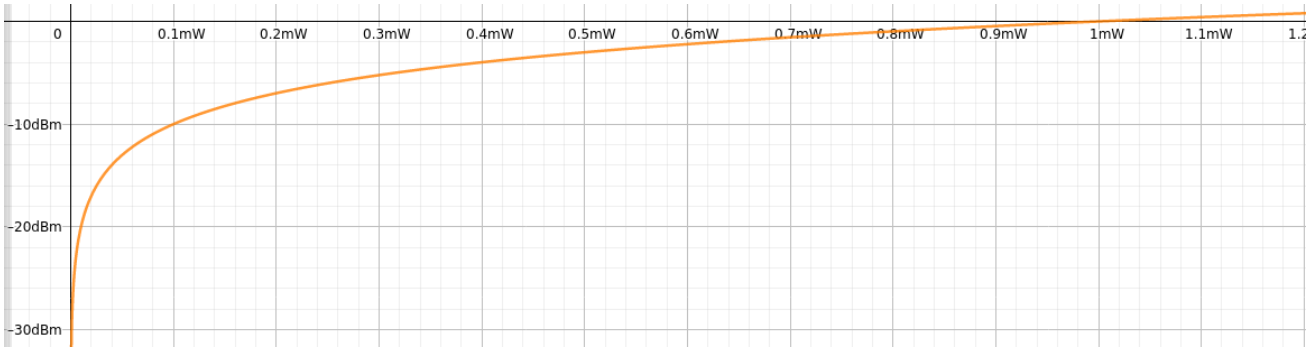

FIGURE 9 – RSSI - 1

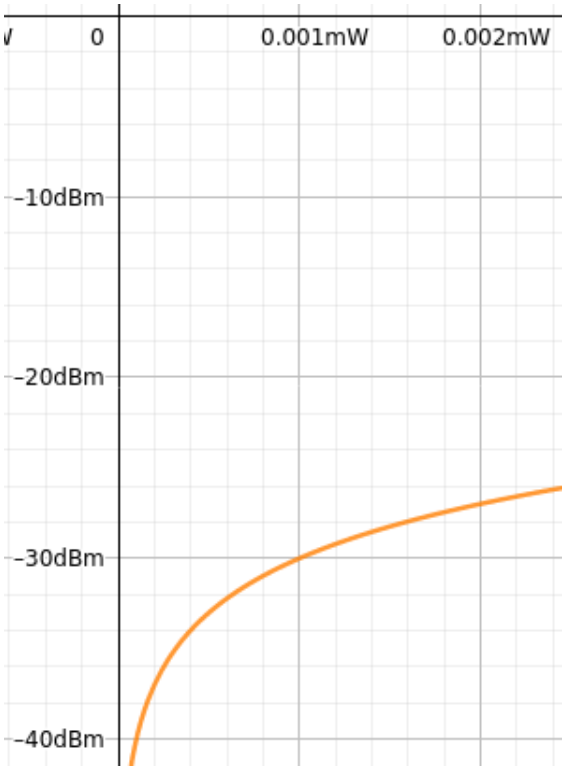

FIGURE 10 – RSSI - 2

Modulation Le type de modulation utilisé, LoRa ici.

Data Rate La donnée du tableau 5 est la concaténation du Spreading Factor ou facteur de propagation, et de la Band Width ou largeur de bande de fréquence utilisée par les capteurs pour la transmission des données. Le data rate ou débit de données dépend de la largeur de bande de fréquences aussi appelé bande passante et du facteur de propagation.

Data Rate (DR)	Modulation	Spreading Factor (SF)	Bande Passante	Débit Physique (bit/s)
0	LoRa	SF12	125 kHz	250
1	LoRa	SF11	125 kHz	440
2	LoRa	SF10	125 kHz	980
3	LoRa	SF9	125 kHz	1 760
4	LoRa	SF8	125 kHz	3 125
5	LoRa	SF7	125 kHz	5 470
6	LoRa	SF7	250 kHz	11 000
7	FSK	50kbit/s		50 000
8	Réservé pour utilisation future			

FIGURE 11 – Débits en fonction des SF et de la BP

Coding rate Le coding rate indique pour un nombre de bits transmis, le nombre de bits d'information. Un coding rate de 4/5 indique que pour 5 bits transmis, 4 sont des bits d'information et 1 est un bit de redondance servant à la correction d'erreurs.

Payload Le payload est les données directement issues des capteurs. Elle sont affichées en base 16 dans le tableau 5 mais sont en réalité encodée en base 64. Le passage en base 16 est par la suite nécessaire pour le déchiffrage.

2 Structure des données

Les données des payloads doivent être déchiffrées afin d'être interprétées. La méthode de chiffrement est différente selon les fabricants des capteurs. La passerelle envoie toutes les heures à la VM OVH sous la forme d'un fichier CSV les données qu'elle a reçues. Un fichier CSV peut être lu sous la forme d'un tableau.

2.1 Capteur On Yield

Le capteur On Yield de référence OY1100 [10] est un capteur de température et d'humidité. Il est actuellement programmé pour envoyer une prise de mesures toutes les 2 heures.

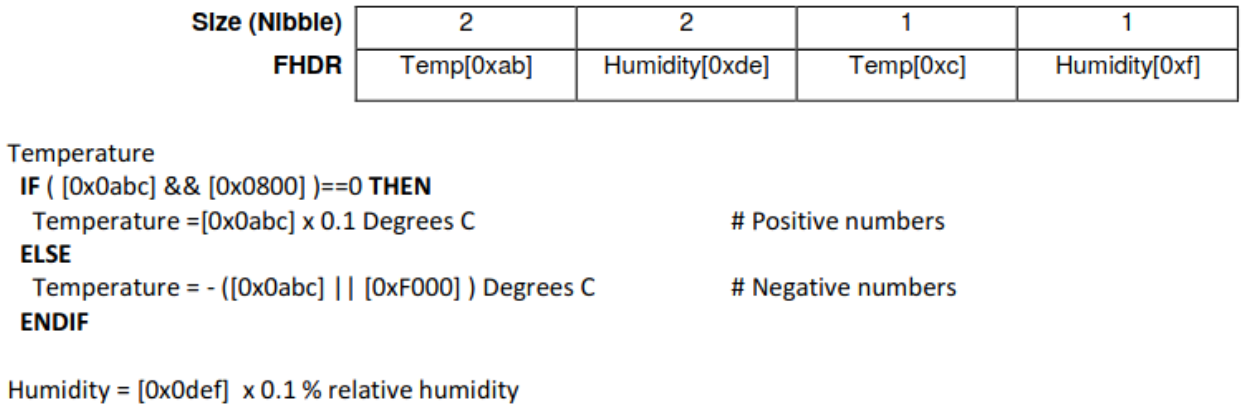

FIGURE 12 – Extrait documentation OY1100

La température et l'humidité sont codés sur 12 bits chacun. Ces bits sont codés en base 16, soit sur 3 symboles. Cependant les symboles codant la température et l'humidité sont entremêlés de la façon suivante : T_1 T_2 H_1 H_2 T_3 H_3
Une fois dans l'ordre il suffit d'effectuer les opérations indiquées dans l'extrait de documentation 12.

2.2 Capteur Elsys.se ERS

Le capteur Elsys.se ERS [11] est un capteur de température, d'humidité, de luminosité et de mouvement. Il est actuellement programmé pour envoyer une prise de mesures toutes les 30 minutes.

Data types			
Type value	*Type*	*Data size*	*Comment*
0x01	Temperature	2	-3276.5 °C → 3276.5 °C (Value of: 100 → 10.0 °C)
0x02	Humidity	1	0 – 100 %
0x04	Light	2	0 – 65535 Lux
0x05	Motion (PIR)	1	0 – 255 (Number of motion counts)
0x07	VDD (Battery voltage)	2	0 – 65535 mV
0x3D	Debug information	4	Data depends on debug information
0x3E	Sensor settings	n	Sensor setting sent to server at startup (first package). Sent on Port+1.

FIGURE 13 – Extrait documentation ELSYS.se ERS

Les données sont codés en base 16 dans l'ordre indiqué dans l'extrait de documentation 13. La taille indiquée par type de donnée est en octet.

3 Script de déchiffrage des données

3.1 Modifications apportées au script

Le script existant était écrit en Python 2.7 et comportait quelques défauts. Les deux scripts, nouvelle et ancienne version sont présents sur la VM.

Modifications :

- Changement du système de passage de la base 64 à la base 16 afin d'utiliser Python 3.8.
- Correction de l'inversion de dénomination des capteurs.
- Dans la boucle principale, suppression du premier tour de boucle `for` car les données ne commencent qu'à la deuxième ligne du fichier CSV.
- Passage de la listes des entêtes plutôt que de tout le fichier aux fonctions de déchiffrement.
- Le script tournait en permanence à la recherche de nouveaux fichiers en lançant continuellement des appels système et n'était utile que 0,03% du temps (un fichier CSV de 3 lignes toutes les heures). Ce qui consomme beaucoup de ressources 14 et induit une plus grande consommation électrique. Code couleur de `htop` [12].

FIGURE 14 – Capture `htop` avant changement

Le script comportait une boucle infinie et son lancement se faisait au démarrage de la VM via une ligne dans le fichier `/etc/rc.local`. Cette ligne est actuellement commentée.

Le lancement du script se fait maintenant une fois toutes 30 minutes via le `cron`, un planificateur de tâches. Une ligne a été ajoutée dans le fichier `/etc/crontab`.

```
# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*/30 *  * * *   root    python3.5 /home/lora/trameLoRa/scripts/decryptAndParse_2_0.py
#
```

FIGURE 15 – Capture `crontab`

FIGURE 16 – Capture `htop` après changement

3.2 Fonctionnement du script

La boucle principale est une boucle `for` qui pour chaque fichier présent dans un dossier donné, va ouvrir ceux portant l'extension `.csv`. La première ligne de chaque fichier `.csv` comporte les noms des données présentes dans les colonnes. Chaque ligne correspond à une prise de mesure d'un capteur. L'identifiant du capteur dans les données permet d'utiliser la bonne fonction de déchiffrement. Les données sont ensuite envoyées à la base de données InfluxDb locale via une requête POST. Format de la requête et fonctionnement : 2.3
Le script comporte également d'une fonction `Lire_Fichier` retournant le fichier `csv` sous la forme d'une liste, ainsi que d'une fonction `FindIndexByName` qui pour une liste, retourne l'index correspondant à une valeur donnée.

3.2.1 Déchiffrement On Yield

```
1  def DecryptageLigneOnYield(ligne, nameOfColumns):
2
3      time_rcvd = int(ligne[FindIndexByName("received_time", nameOfColumns)])
4        *1000000000
5
6      payload = str(base64.b16encode(base64.b64decode(str(
7        ligne[FindIndexByName("payload", nameOfColumns)]))))
8      tabPayload = list(payload)
9
10     # le format bits converti en str sont de la forme : b"abcdefg" il faut
       donc supprimer le b et les guillemets
11     del(tabPayload[0:2])
12     del(tabPayload[-1])
13
14     # les symboles de la temperature sont concatenes ensemble et convertis
       en base 10
15     tempBeforeConvert = (int(tabPayload[0],16)*16*16 +
16       int(tabPayload[1],16)*16 + int(tabPayload[4],16))
17
18     temp = 500
19
20     # calculs logiques issus de la documentation du capteur figure[12]
21     if tempBeforeConvert & 0x0800 == 0:
```

```
        temp = tempBeforeConvert / 10.0
    else:
        temp = - (tempBeforeConvert | 0xF000)

  # les symboles de l'humidite sont concatenes ensemble et convertis en base 10
    hum = ( int(tabPayload[2],16) * 16 * 16 + int(tabPayload[3],16) * 16 + int(tabPayload[5],16) )/10.0

    # recuperation du ssi et du SNR(db)
    snr_db = ligne[FindIndexByName("snr_db", nameOfColumns)]
    rssi_dbm = ligne[FindIndexByName("rssi_dbm", nameOfColumns)]

    return (time_rcvd, temp, hum, snr_db, rssi_dbm)
```

3.2.2 Déchiffrement Elsys.se ERS

```
def DecryptageLigneElsys(ligne, nameOfColumns):

    time_rcvd = int(ligne[FindIndexByName("received_time", nameOfColumns)])
      *1000000000
    payload = str(base64.b16encode(base64.b64decode(str(
      ligne[FindIndexByName("payload", nameOfColumns)]))))

    # le format bits converti en str sont de la forme : b"abcdefg" il faut donc supprimer le b et les guillemets
    tabPayload = list(payload)
    del(tabPayload[0:2])
    del(tabPayload[-1])

    # les symboles d'une meme mesure sont concatenes ensemble et convertis en base 10 suivant la figure[13]
    temp = (int(tabPayload[2],16) * 16*16*16 + int(tabPayload[3],16)
      *16*16 + int(tabPayload[4],16)*16 + int(tabPayload[5],16)*1)/10.0

    hum = (int(tabPayload[8],16) * 16 + int(tabPayload[9],16)*1)

    light = (int(tabPayload[12],16)*16*16*16 + int(tabPayload[13],16)*16*16
      + int(tabPayload[14],16)*16 + int(tabPayload[15],16)*1)

    motion = (int(tabPayload[18],16)*16 + int(tabPayload[19],16)*1)

    vdd = (int(tabPayload[22],16)*16*16*16 + int(tabPayload[23],16)*16*16
      + int(tabPayload[24],16)*16 + int(tabPayload[25],16)*1)/1000.0

    # Recuperation du ssi et du SNR(db)
    snr_db = ligne[FindIndexByName("snr_db", nameOfColumns)]
    rssi_dbm = ligne[FindIndexByName("rssi_dbm", nameOfColumns)]

    return (time_rcvd, temp, hum, light, motion, vdd, snr_db, rssi_dbm)
```

Quatrième partie
OpenWeather

Cette partie sert d'exemple pour l'usage d'InfluxDb et de Grafana.

1 API OpenWeather

1.1 Principe

OpenWeather [13] regroupe plusieurs API permettant de récupérer des données météorologiques. L'API utilisée est Current Weather qui permet jusqu'à 60 requêtes par minute dans sa version gratuite, sachant que les données disponibles sont mises à jour toutes les 10 minutes. La requête utilisée est de la forme suivante :
`api.openweathermap.org/data/2.5/weather?zip={zip code},{country code}&appid={your api key}&units=metric`

- zip code : code postale
- country code : identifiant du pays
- API key : identifiant lié au compte crée, `98b0cfd0bfd462ed697ae60003716368` dans notre cas
- `units=metric` : le système par défaut est le système impérial

Ce qui donne : `http://api.openweathermap.org/data/2.5/weather?zip=29200,fr&appid=98b0cfd0bfd462ed697ae60003716368&units=metric` . Vous pouvez essayer dans votre navigateur. Les données renvoyées sont au format JSON.

1.2 Script de requêtage

Les deux parties du code sont expliquées séparément mais sont dans un seul et même fichier. Le script se trouve dans `/home/lora/openWeather.py`. Les données de l'API étant mises à jour tout les 10 minutes, une ligne est ajoutée au `cron` afin de lancer le script toutes les 10 minutes, cf. 15.

```
1  import socket
2  import json
3  import requests
4  #import time
5
6  api = "api.openweathermap.org" # OpenWeather prefere qu'on utilise son NDD
       plutot que son IP
7  api_port = 80 # port HTTP conventionnel
8
9  # AF_INET : IPv4 ; SOCK_STREAM : TCP
10 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11
12 client.connect((api, api_port)) # connexion au serveur
13
14 request = "GET /data/2.5/weather?zip=29200,fr&appid=98
       b0cfd0bfd462ed697ae60003716368&units=metric\n" # ne pas oublier le
       caractere '\n' a la fin
15
```

```
client.send(request.encode()) # envoi de la requete de type GET

response = client.recv(4096) # taille du buffer de reception, doit etre une
    puissance de 2

jsonResponse = json.loads(response.decode()) # formatage JSON
```

2 InfluxDb - Stockage des données

Les données sont stockée dans une base de données de type Time Series. Une telle BDD permet de stocker de nombreuses données dont la particularité est que chaque enregistrement ou point est indexé par rapport au temps.

2.1 Concepts clés

Base de données : regroupe un ensemble de mesures ayant un rapport entre elles. Mesures de capteurs LoRa ou d'OpenWeather par exemple.

Measurement : table qui comme son nom l'indique, regroupe les mesures ou point. Similaire à une table dans une BDD classique.

Timestamp : chaque **measurement** dispose d'une colonne par défaut nommée "time", elle comporte un timestamp en nanoseconde. InfluxDb l'attribut automatiquement s'il n'est pas précisé lors de l'ajout de données.

Tag set : ensemble des **tag key** et **tag value**. Informations optionnelles, le plus souvent des constantes prédéfinies, fonctionnant par clé-valeur et servant à qualifier le **field set**. Pour une mesure de température par exemple, "ville" est la **tag key** et "Brest" la **tag value** associée. Les **tag sets** sont indexés et permettent d'affiner une recherche de données sans efforts.

Field set : ensemble des **field keys** et **field values**. Fonctionnant par clé-valeur, ce sont les valeurs mesurées. Les **field sets** ne sont pas indexés, ce qui signifie que pour effectuer une recherche par **field value**, il faut parcourir tout le **measurement**.

Retention policy : règle fixant la durée pendant laquelle InfluxDb va conserver les données d'une BDD, la durée est infinie par défaut. Cela permet de gérer l'espace de stockage.

Source : [14]

2.2 Commandes de base

Un des intérêts d'InfluxDb est que son langage de requêtage est similaire au langage SQL [15]. Pour lancer la CLI saisissez `influx`.

Afficher les bases de données présentes :
`show databases`

Créer une BDD :
`create database nomDeLaNouvelleBDD`

Utiliser une BDD :
`use nomDeLaBDD`

Afficher les measurements :
`show measurements`
Une measurement est crée automatiquement lors de l'ajout d'un premier point appartenant à cette dernière.

Afficher tous les points ajoutés depuis X minutes :
`select * from nomDeLaMeasurement where time > now() - Xm`

Supprimer tous les point ajoutés avant il y a X heures :
`delete from nomDeLaMeasurement where time < now() - Xh`

Supprimer une measurement :
`drop nomDeLaMeasurement`
Même principe pour les BDD.

Afficher l'historique des commandes saisies :
`history`

Afficher les retention policies :
`show retention policies`

Mise en place d'une retention policy d'une durée de X jours :
ATTENTION ! Il vous est vivement conseillé de lire la documentation [16] avant de mettre en place ce genre de règle.
`create retention policy nomDeLaRententionPolicy on nomDeLaBDD duration Xd replication 1 default`

2.3 Script de requêtage

L'ajout de données à la BDD se fait par requêtage de l'API de la BDD [17]. La requête est de la forme suivante : `http://localhost:8086/write?db=`*nomDeLaBDD* comme adresse puis les données *nomDeLaMeasurement*`,`*tagKey1*`=`*tabValue1*`,`*tagKeyN*`=`*tagValueN* *fieldKey1*`=`*fieldValue1*`,`*fieldKeyN*`,`*fieldValueN* `timestamp` .

```
data = "weatherByCity,city="+str(jsonResponse["name"])
  +" "
  +"temperature="+str(jsonResponse["main"]["temp"])
  +",feels_like="+str(jsonResponse["main"]["feels_like"])
  +",pressure="+str(jsonResponse["main"]["pressure"])
  +",humidity="+str(jsonResponse["main"]["humidity"])
  +",wind="+str(jsonResponse["wind"]["speed"])
  # +" "+str(int(time.time())*100000000) conversion en nanoseconde
# le timestamp fourni par le script est moins precis que celui ajoute
   automatiquement par InfluxDb
```

```
10
11 r = requests.post(url="http://localhost:8086/write?db=openWeather", data=
       data)
```

3 Grafana - Affichage des données

Grafana est une plateforme opensource et simple d'usage de visualisation de données sous forme de graphiques en tout genre.

3.1 Création d'un dashboard

Commencez par ajouter une nouvelle source de données via la barre de gauche : Configuration/Data Sources puis cliquez sur le bouton vert "Add data source". Sélectionnez InfluxDb. Remplissez les champs suivants :

— Name : nom que vous voulez
— URL : adresse de la BDD, ici `http://localhost:8086`
— Database : nom de la BDD crée précédemment

Les données de la BDD ne sont pas protégée donc le user et le mot de passe ne sont pas utiles.

Via la barre de gauche, sélectionnez Create/Dashboard/Add Query. Sélectionnez dans le menu déroulant Query la source de données que vous venez d'ajouter puis créez votre requête à partir des champs présents. Vous pouvez personnaliser votre dashboard via les menus présents à gauche. Le reste est instinctif. L'axe Y de droite ne fonctionne pas, cela vient d'un bug de Grafana.
N'oubliez pas d'enregistrer vos modifications en cliquant sur le bouton "Save dashboard" en haut à droite.

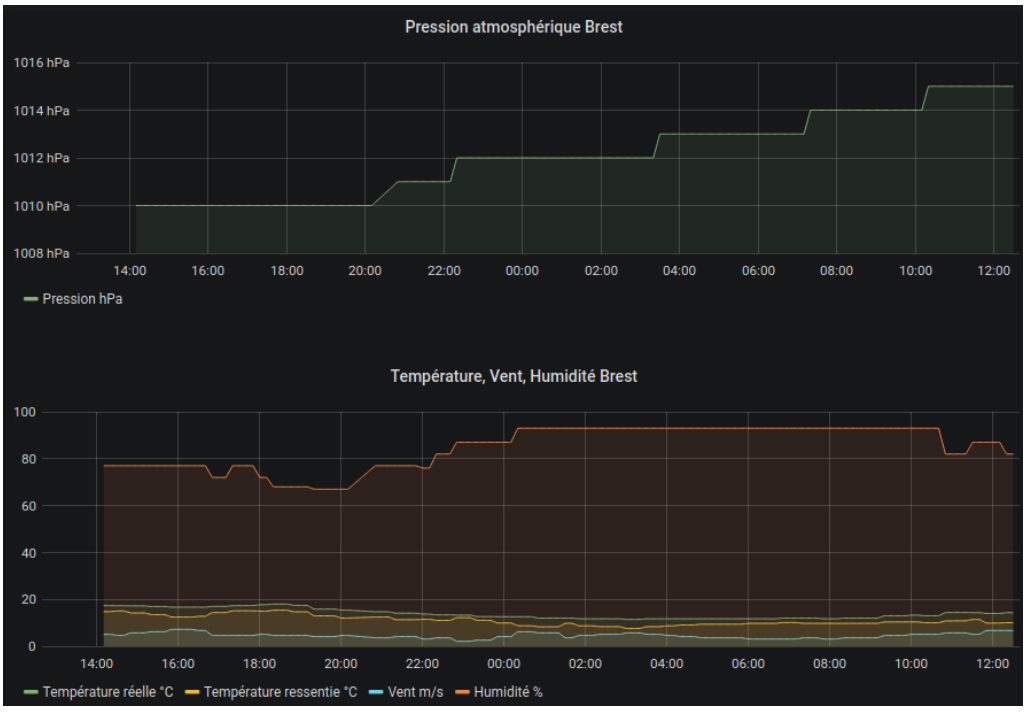

FIGURE 17 – Exemple de dashboard

Références

[1] https://raspap.com/

[2] https://www.linuxtricks.fr/wiki/ftp-installer-et-configurer-un-serveur-proftpd

[3] https://openvpn.net/community-resources/how-to/#openvpn-quickstart

[4] http://www.hydrogen18.com/blog/your-own-pki-tls-golang.html

[5] https://github.com/OpenVPN/easy-rsa

[6] https://www.it-connect.fr/chapitres/transfert-de-fichier-via-ssh/

[7] https://askubuntu.com/questions/765951/starting-openvpn-error-please-enter-password-with-the-systemd-tty-ask-password

[8] https://serverfault.com/questions/515833/how-to-remove-private-key-password-from-pkcs12-container

[9] https://doc.ubuntu-fr.org/fail2ban

[10] http://www.dataprint.fr/support/talkpool/Manuel_OY1100.pdf

[11] https://elsys.se/public/datasheets/ERS_datasheet.pdf

[12] https://serverfault.com/questions/180711/what-exactly-do-the-colors-in-htop-status-bars-mean

[13] https://openweathermap.org/

[14] https://docs.influxdata.com/influxdb/v1.7/concepts/key_concepts

[15] https://docs.influxdata.com/influxdb/v1.7/concepts/crosswalk/

[16] https://docs.influxdata.com/influxdb/v1.7/query_language/database_management/#create-retention-policies-with-create-retention-policy

[17] https://docs.influxdata.com/influxdb/v1.7/guides/writing_data/