# Manuale d’installazione e d’uso

L’installazione di tutte le componenti necessarie per il corretto funzionamento del codice richiede una lunga serie di passaggi che devono essere svolti principalmente da linea di comando.

I primi passi di questa installazione verranno svolti su *Ubuntu*, un programma scaricabile da *Microsoft Store* che consiste in un sistema operativo basato su *Linux*, e, successivamente, ci si sposterà sulla *Windows Subsystem for Linux* del *Prompt dei comandi* di *Windows*.

## Requisiti software

I seguenti passi sono necessari nel momento in cui si sta lavorando con un sistema operativo *Windows*: l’obiettivo è quello di installare un sistema dotato di sistema operativo *Linux*.

### Installazione di Ubuntu

Per prima cosa bisogna scaricare *Ubuntu* dal *Microsoft Store*.

Una volta fatto ciò, si deve avviare l’applicazione e, alla prima apertura, verrà richiesto di creare una partizione *Linux* tramite un *username* e una *password*.

Questa *password* sarà fondamentale perché verrà richiesta ogni volta che si vorrà entrare nella partizione *Linux* al fine di avviare l’*application server*.

### Installazione wsl 2.0

Successivamente bisogna installare *WSL 2.0* sulla propria macchina. Si tratta di un layer di compatibilità che permette l’esecuzione di file binari *Linux* in modo nativo su *Windows*.

Prima di fare ciò, è necessario aprire il *Prompt dei comandi* in modalità *Esegui come amministratore* ed eseguire il comando

* wsl.exe --install

per abilitare le funzioni opzionali di WSL.

Fatto questo, bisognerà riavviare il sistema, aprire *Ubuntu* ed eseguire i seguenti due comandi

* sudo apt update
* sudo apt upgrade

A questo punto si può passare all’installazione di WSL 2.0. Per eseguire l’upgrade da WSL 1.0 basta aprire nuovamente il *Promp dei comandi* con i permessi d’amministratore ed eseguire il comando

* wsl.exe --set-version Ubuntu 2

In questo modo si installerà e si setterà di default WSL 2.0 sul proprio computer.

## Requisiti per le componenti del ChirpStack LoRaWAN Network Server

Prima di iniziare con l’installazione dell’*application server*, è necessario installare tre software seguendo la guida che si può trovare al link

“ *https://www.chirpstack.io/project/install/requirements/* “

Per prima cosa bisogna aprire *Ubuntu* ed eseguire i seguenti 3 comandi

* sudo apt install mosquitto

Questo comando installa *Mosquitto*, un *broker* dei messaggi che implementa il protocollo MQTT.

* sudo apt install postgresql

Questo comando installa *PostgreSQL*, un DBMS (*DataBase Management System*) che verrà usato dall’*application server* per gestire il suo database.

* sudo apt install redis-server

Questo commando installa *Redis*, un archivio di strutture dati in memoria *open source* usato come database, cache e *broker* di messaggi.

## Installazione application server

A questo punto si può passare all’installazione dei componenti messi a disposizione dallo *stack ChirpStack*. Per farlo è stata seguita la guida presente al link

“ *https://www.chirpstack.io/project/guides/debian-ubuntu/* ”.

### Creazione del database per l’application server

Si deve aprire *Ubuntu* con i permessi d’amministratore, entrare nella partizione *Linux* mediante *username* e *password* ed eseguire i seguenti comandi:

* sudo -u postgres psql

Per creare un database per l’*application server* e spostarsi nel prompt di *PostgreSQL*

* create role chirpstack\_as with login password ‘dbpassword’

Per creare l’*user chirpstack\_as* (inserire una password al posto di *‘dbpassword’*).

* create database chirpstack\_as with owner chirpstack\_as

Per creare il database *chirpstack\_as*.

* \c chirpstack\_as
* create extension pg\_trgm
* create extension hstore
* \q

Questi comandi creano delle estensioni nell’*application server* e, infine, consentono di uscire dal *prompt* di *PostgreSQL*.

Per verificare il corretto setup, si può eseguire il comando

* psql -h localhost -U chirpstack\_as -W chirpstack\_as

### Installazione delle componenti

I successivi comandi sono usati per installare, avviare ed abilitare le varie componenti:

* sudo apt install chirpstack-gateway-bridge

sudo systemctl start chirpstack-gateway-bridge

sudo systemctl enable chirpstack-gateway-bridge

* sudo apt install chirpstack-network-server

sudo systemctl start chirpstack-network-server

sudo systemctl enable chirpstack-network-server

* sudo apt install chirpstack-application-server

sudo systemctl start chirpstack-application-server

sudo systemctl enable chirpstack-application-server

*start* serve per avviare la particolare componente. Se al suo posto si scrive *restart* o *stop* o *status*, il comando, rispettivamente, esegue un riavvio o termina l’esecuzione o fornisce lo stato del componente.

Quando si crea un componente, viene generato il suo corrispondente *Configuration file*. Per potervi accedere bisogna eseguire i comandi che seguono:

* sudo su

Per ottenere i permessi di amministratore su *Ubuntu*.

* cd /etc

Per accedere alla *directory* contenente le cartelle relative ai vari componenti.

Dentro questa cartella bisogna eseguire il comando

* cd /’componente’

dove al posto di ‘componente’ bisogna scrivere o *chirpstack-gateway-bridge* o *chirpstack-network-server* o *chirpstack-application-server* per entrare nella cartella di quello specifico componente. Infine, per aprire il file contenente la configurazione di quel particolare componente bisogna eseguire il comando

* sudo nano ‘componente’.toml

Si deve accedere ad ognuno di questi file in quanto bisogna apportare delle modifiche:

* In quello del *gateway* bisogna modificare i seguenti due campi
  + Nella sezione *Integration configuration* bisogna scrivere marshaler="json", in quanto *JSON* sarà il formato utilizzato per rappresentare i dati (e non *Protobuf*).
  + Nella sezione *Integration configuration / MQTT authentication* bisogna mettere username="chirpstack\_gw"
* In quello del *Network server* le operazioni da fare sono molteplici
  + Tra i *PostgreSQL Settings* bisogna modificare la stringa “postgres://user:password@hostname/database?sslmode=disable” sostituendo i *placeholder* con le stringhe opportune.

Un esempio di quanto scritto potrebbe essere il seguente:

dsn="postgres://chirpstack\_ns:MyPassword@localhost/chirpstack\_ns?sslmode=disable"

* + Nella sezione *Network-server settings / LoRaWAN regional band configuration* bisogna settare name="EU868". Questo valore corrisponde alla zona geografica in cui il sistema è stato installato e richiede di creare dei canali con determinate frequenze derivanti dalle specifiche *LoRaWAN*
  + Nelle *Extra channel configuration* bisogna inserire i seguenti sei insiemi di parametri, corrispondenti a frequenze diverse per i canali del *network server*:
    - [[network\_server.network\_settings.extra\_channels]]

frequency=867100000

min\_dr=0

max\_dr=5

* + - [[network\_server.network\_settings.extra\_channels]]

frequency=867300000

min\_dr=0

max\_dr=5

* + - [[network\_server.network\_settings.extra\_channels]]

frequency=867500000

min\_dr=0

max\_dr=5

* + - [[network\_server.network\_settings.extra\_channels]]

frequency=867700000

min\_dr=0

max\_dr=5

* + - [[network\_server.network\_settings.extra\_channels]]

frequency=867900000

min\_dr=0

max\_dr=5

* + - [[network\_server.network\_settings.extra\_channels]]

frequency=868100000

min\_dr=0

max\_dr=5

* + Nella sezione *Backend defines the gateway backend settings / MQTT gateway backend settings* bisogna settare username="chirpstack\_ns"
* Nell’Application server bisogna modificare le seguenti voci:
  + Tra i *PostgreSQL Settings* bisogna modificare la stringa “postgres://user:password@hostname/database?sslmode=disable” sostituendo i *placeholder* con le stringhe opportune.

Un esempio di quanto scritto potrebbe essere il seguente:

dsn="postgres://chirpstack\_as:MyPassword@localhost/chirpstack\_as?sslmode=disable"

* + Nella sezione *Application server settings / Integration configures the data integration* bisogna scrivere marshaler="json"
  + Infine, sempre negli *Application server settings*, bisogna inserire

username="chirpstack\_as"

A questo punto si devono eseguire i seguenti tre comandi:

* cd /tmp
* wget --content-disposition \https://gist.githubusercontent.com/djfdyuruiry/6720faa3f9fc59bfdf6284ee1f41f950/raw/952347f805045ba0e6ef7868b18f4a9a8dd2e47a/install-sg.sh chmod +x /tmp/install-sg.sh
* /tmp/install-sg.sh && rm /tmp/install-sg.sh

Fatto ciò si deve aprire il *Prompt dei comandi* tramite *Esegui come amministratore* ed eseguire questi altri tre comandi:

* wsl --shutdown

Per terminare l’esecuzione di *Ubuntu*.

* wsl genie -s
* sudo systemctl status time-sync.target

In questo modo l’interfaccia grafica dell’*application server* sarà visibile all’indirizzo *localhost:8080.*

Una volta spento il computer, per poter visualizzare nuovamente l’*application server* in locale sarà necessario eseguire il comando

* wsl genie -s

dal *Prompt dei comandi* (avviato sempre con *Esegui come amministratore*), altrimenti cercando direttamente *localhost:8080* il motore di ricerca dirà che a tale indirizzo non è possibile trovare nulla.

### Controllare lo stato delle componenti

I seguenti comandi non servono per l’installazione delle componenti, ma permettono di visualizzare a video l’output di quella particolare componente, contenente diverse informazioni:

* journalctl -u chirpstack-gateway-bridge -f -n 50
* journalctl -u chirpstack-network-server -f -n 50
* journalctl -u chirpstack-application-server -f -n 50

## Configurazione dell’interfaccia grafica

Una volta inseriti *username* e *password* ed entrati nell’*application server,* bisogna passare alla configurazione della rete.

### Organization

Per creare una nuova *Organization* (se non già presente o se si vuole configurare tutto da capo), cliccare su *Organizations* sulla sinistra dell’interfaccia ed inserire *Name* e *Display name*, mantenendo a 0 *Number of devices* e spuntando la voce *Organization can have gateways*.

Immagine che contiene testo

Descrizione generata automaticamente

### Network-servers

Il passo successivo è quello di creare un network server. Per farlo basta inserire un *Network-server name* ed un *Network-server server*.

Nella sezione *gateway discovery* è possibile spuntare *Enable gateway discovery*, una funzione che gestisce i ping automatici tra *gateway*. Questa funziona molto bene con *gateway* fisici interconnessi tra loro, mentre la componente software di questo progetto non è stata programmata per farlo.

Immagine che contiene testo

Descrizione generata automaticamente

### Gateway-profiles

Successivamente si deve creare un *Gateway-profile* collegato al *network server* appena generato. Per farlo bisogna inserire *Name*, *Stats interval(seconds)*, *Enabled channels* e *Network-server*, quest’ultimo scelto da un menù a tendina che propone quelli esistenti.

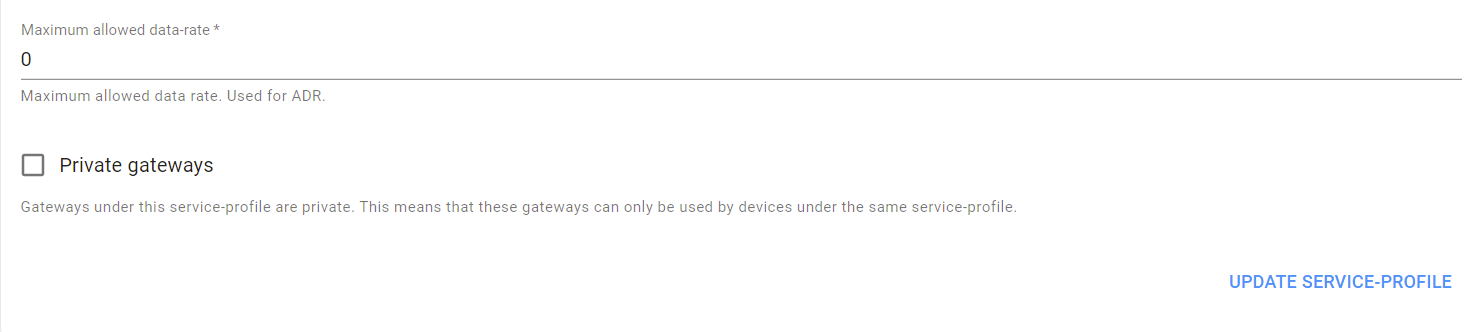
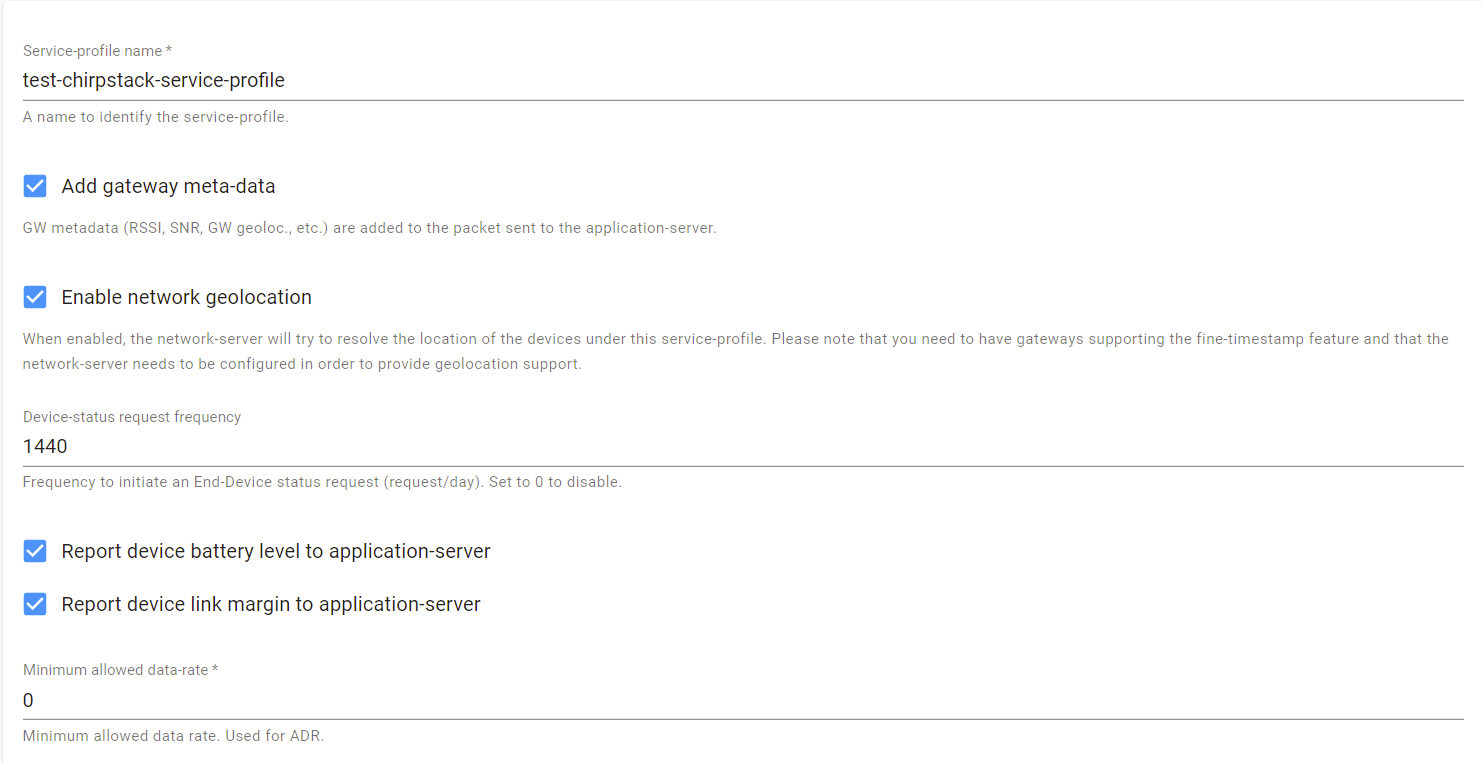
Immagine che contiene testo

Descrizione generata automaticamente

### Service profile

Passo successivo è quello di creare un *Service-profile*. Bisogna inserire un *Service-profile name*, il *Network-server* creato in precedenza selezionandolo dal menù a tendina, spuntare *Add gatewat meta-data* e *Enable network geolocation* e settare *Device-status request frequency* a 1440.

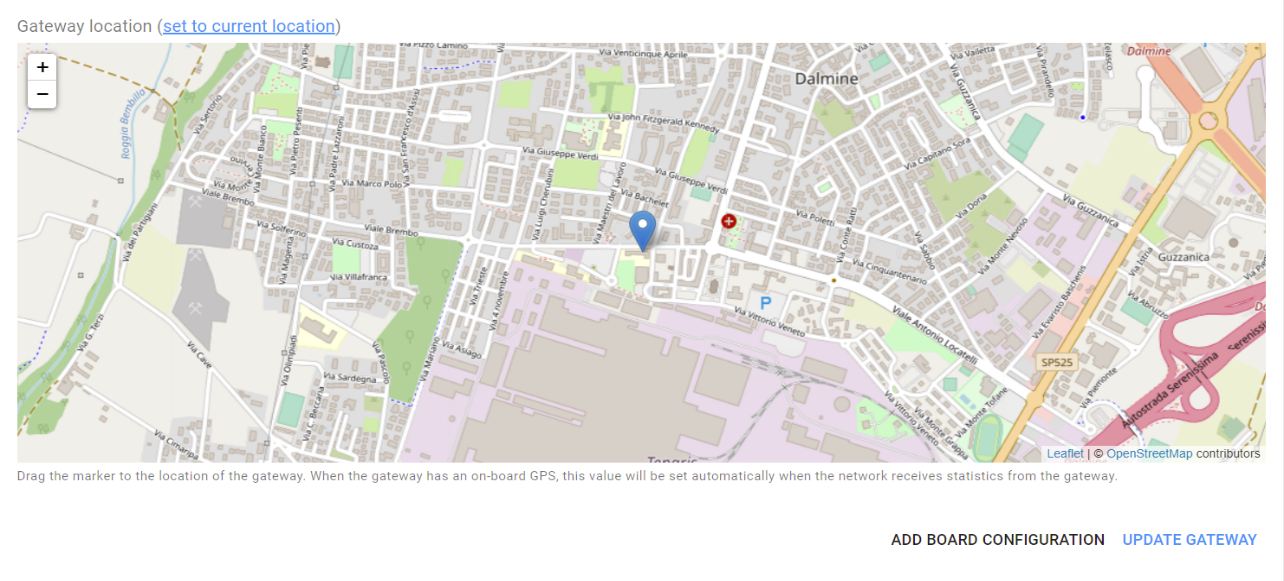
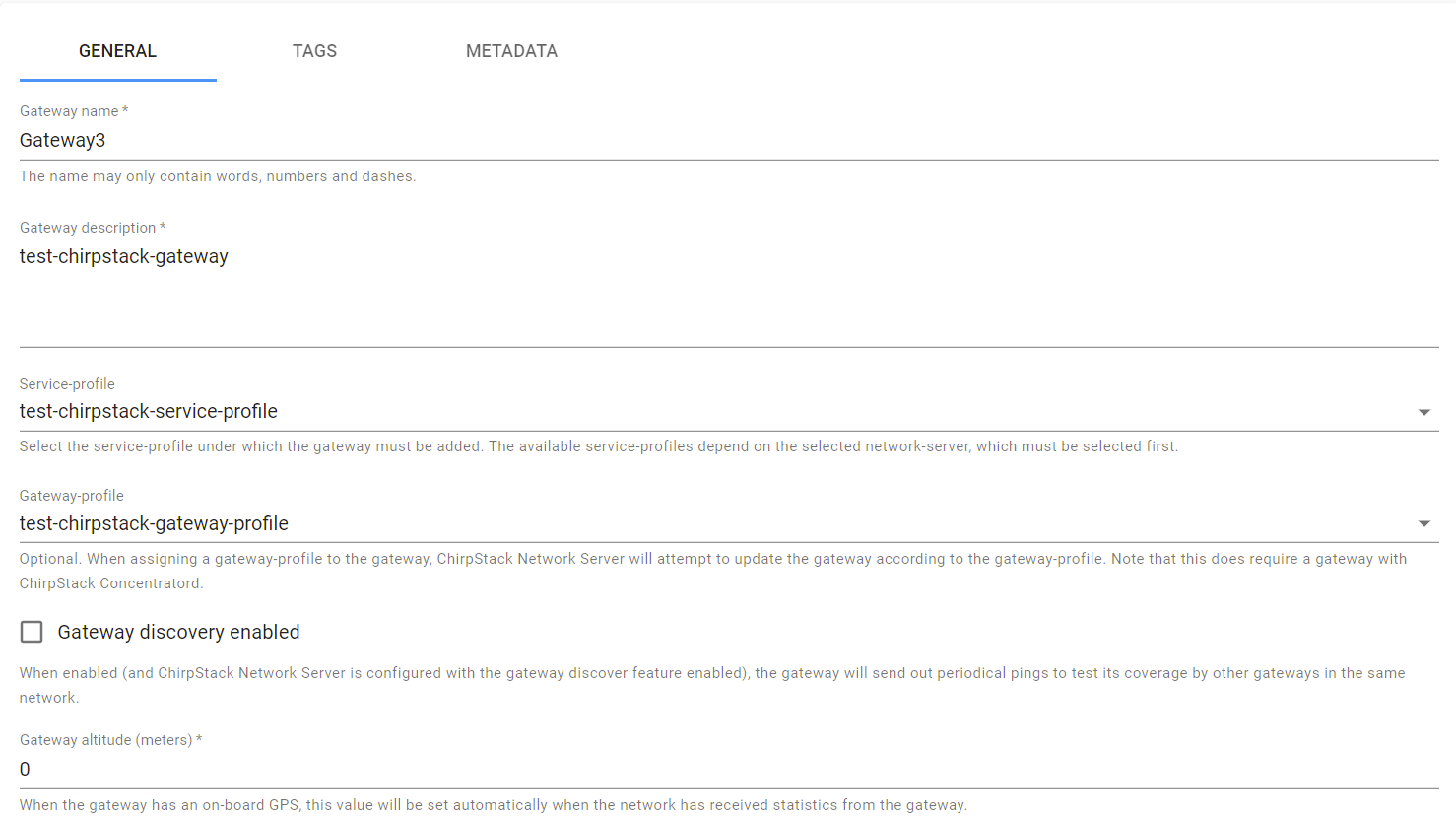
Una volta creato sarà necessario, inoltre, spuntare le due caselle di report presenti nel particolare *Service-profile*, *Report device battery level to application-server* e *Report device link margin to application-server*.



### Gateway

Immagine che contiene mappa

Descrizione generata automaticamenteOra si passa creare il *gateway* della rete. Inseriti *Gateway name* e *Gateway description*, si deve generare un *Gateway ID* valido (anche automaticamente) e selezionare il *Network-server*, il *Service-profile* ed il *Gateway-profile* creati in precedenza dai rispettivi menù a tendina. Infine si deve scegliere la posizione nella mappa nella quale il gateway verrà messo.



### Applications

Si passa ora alla creazione dell’*Application*, inserendo *Application name*, *Application description* e selezionando il *Service-profile* creato in precedenza dal menù a tendina.

Immagine che contiene testo

Descrizione generata automaticamente

### Device-profile

Arrivati a questo punto bisogna creare i dispositivi che si connetteranno ai *gateway* della rete. Prima di fare ciò bisogna generare un *Device-profile*. Per prima cosa si inserisce un *Device-profile name* e si seleziona il *Network server* creato prima dal menù a tendina. I successivi campi vanno riempiti nel modo seguente:

* *LoRaWAN MAC version* a *1.0.0*
* *LoRaWAN Regional Parameters revision* ad *A*
* *ADR algorithm* a *Default ADR algorithm*
* *Uplink interval (seconds)* a 30

Infine bisogna spuntare il campo *Device supports OTAA* nella sezione *JOIN (OTAA/ABP)*.

Immagine che contiene testo

Descrizione generata automaticamente

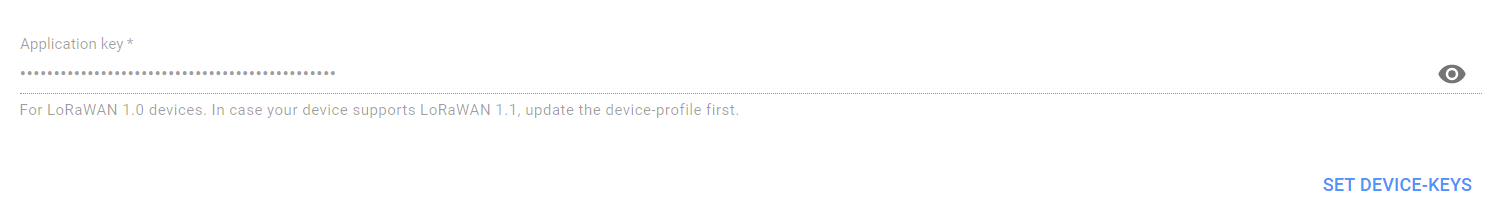
### Device

A questo punto si possono creare i *device* veri e propri. Entrando in *Applications* e selezionando l’*application* creata, si entra nella sezione *Device*. Qui è possibile creare un device inserendo *Device name*, *Device description*, *Device EUI* (l’identificativo univoco del device che può anche essere generato automaticamente), *Device profile* selezionato dal menù a tendina e spuntando la voce *Disable frame-counter validation*.

Immagine che contiene testo

Descrizione generata automaticamente

Quando si crea un *device*, esso non viene immediatamente attivato. Bisogna settare un *Application key* nella sezione *KEYS (OTAA)*, la quale rimarrà fissa per tutta la vita del *device*.



Inoltre, ogni volta che si fa una *join* tra *device* e *application server*, vengono generati dinamicamente, nella sezione *Activation, Device address*, *Network session key* e *Application session key*.

Immagine che contiene tavolo

Descrizione generata automaticamente

## Modifiche da apportare al codice sorgente

Configurato il tutto, se si prova a far girare il codice sorgente esso non funzionerà. Infatti bisogna apportare una modifica al codice ogni qual volta che si esegue il comando per avviare l’*application server*.

### Prelevare l’indirizzo della virtual machine

Innanzitutto bisogna accedere alla macchina virtuale, che gira su WSL 2.0, dal codice: questo creerà la connessione tra le componenti del codice e quelle create da linea di comando.

Su *Ubuntu* eseguire i seguenti comandi

* wsl -l -v

Per vedere la versione di *Ubuntu* in esecuzione.

* ip addr | grep eth0

Per prelevare l’indirizzo IP della *virtual machine* (indirizzo IP dopo la voce *inet*).

Per vedere l’indirizzo IP di Windows, eseguire su *Prompt dei comandi*

* ipconfig

e guardare sotto la sezione *Scheda LAN wireless Wi-Fi indirizzo IPv4*. Si noterà che l’indirizzo IP di Windows e quello della *virtual machine* sono diversi.

### Creazione della connessione

Arrivati a questo punto bisogna consentire la connessione da *Ubuntu* a *Windows*. Questo viene fatto attraverso due passaggi.

Il primo consiste nel creare una nuova regola sul *firewall* per gestire la connessione di dispositivi diversi alla macchina virtuale Ubuntu. Infatti, a questo punto, l’unico dispositivo che conosce della *virtual machine* che gira su *Ubuntu* è il computer sul quale *Ubuntu* è in esecuzione. Bisogna rendere quella *virtual machine* visibile anche ad altri dispositivi. Per fare ciò bisogna seguire questi passaggi:

* Entrare nel pannello di controllo di Windows
* Rete internet
* Centro connessioni di rete
* Windows defender firewall
* Impostazioni avanzate
* Regole connessioni in entrata
* Nuova regola (con tasto dx)
* Selezionare porta, avanti
* mettere porta 3390(o una qualsiasi libera)
* avanti fino alla fine e confermare

Il secondo passo, invece, deve essere eseguito da *Windows PowerShell* dotato dei permessi d’amministratore. Bisogna eseguire il comando

* netsh interface portproxy add v4tov4 listenport=3390 listenaddress=0.0.0.0 connectport=’port’ connectaddress=’ubuntu\_addr’

Dove bisogna sostituire ‘port’ con il numero di porta che si è inserito al passo precedente (es.: 3390) e ‘ubuntu\_addr’ con l’indirizzo della *virtual machine*.

Questi passi andranno eseguiti tutti una sola volta.

### Collegamento tra codice e application server

Arrivati a questo punto bisogna prelevare l’indirizzo della *virtual machine* da *Ubuntu* ed inserirlo all’interno del codice, al fine di creare la connessione tra il codice ed il *broker* MQTT presente sulla *virtual machine*. Questa operazione va eseguita prima di ogni avvio dell’*application server* in quanto tale indirizzo potrebbe variare.

Bisogna recarsi nella classe *main.py* e scrivere l’indirizzo della propria *virtual machine* nel parametro *broker = “ubuntu\_addr”* (dove al posto di ubuntu\_addr va inserito il particolare indirizzo IP citato prima).

Se tutti questi passaggi sono stati fatti correttamente, il codice comunica con l’interfaccia grafica dell’application server e sarà possibile vedere i messaggi.

WHOOPSIDAISIES