# Guide d’utilisation de la plateforme edge impulse

## I. Définition :

Edge Impulse est une plateforme en ligne conçue pour faciliter la création de modèles d’intelligence artificielle (IA) à destination des appareils embarqués (microcontrôleurs, capteurs, objets connectés, etc.).
Elle permet de :

- Collecter des données (vibrations, sons, signaux…),
- Prétraiter les signaux (filtrage, FFT, statistiques),
- Entraîner un modèle d’IA (classification, régression),
- Évaluer ses performances,
- Déployer ce modèle sur un appareil embarqué de manière simple et optimisée.

Son objectif principal est de rendre accessible le développement de l’IA embarquée, même sans expertise avancée en Machine Learning ou traitement du signal.

Ce guide s'adresse aux personnes souhaitant reprendre ou comprendre l'utilisation de la plateforme Edge Impulse dans le cadre de ce projet. Il détaille pas à pas toutes les étapes nécessaires pour entraîner un modèle d'intelligence artificielle embarquée sur un microcontrôleur Arduino, à partir de données vibratoires issues d'un moteur électrique. Ce guide suppose une connaissance de base d’Arduino, du terminal et de la logique d’acquisition de données.

## II. L’utilisation de Edge dans notre projet

Dans ce projet, l’objectif est d’exploiter les vibrations d’un moteur pour détecter des défauts mécaniques (déséquilibre, balourd, désalignement…). Ces vibrations sont analysées localement sur l’Arduino Nano RP2040, puis utilisées pour entraîner une IA avec Edge Impulse.

Edge Impulse est utilisé ici car il permet de :

- Gérer l’envoi des données FFT transformées depuis l’Arduino vers le cloud.
- Étiqueter facilement les différents états du moteur (normal, déséquilibre…).
- Appliquer un traitement statistique léger pour l’embarqué
- Entraîner un modèle optimisé pour être **exécuté localement** sur l’Arduino Opta
- Exporter directement une bibliothèque compatible avec l’environnement Arduino

## III. Prérequis (matériel, logiciel, compte)

- **Matériel**
  - Arduino Nano RP2040 Connect (capteur LSM6DS3 intégré)
  - Câble USB-C pour connexion PC
  - Ordinateur Windows/macOS/Linux
- **Logiciels à installer**
  - Arduino IDE
  - Node.js + Edge Impulse CLI :
    npm install -g edge-impulse-cli
  - Pilotes USB Arduino si nécessaire
- Créer un compte edge impulse gratuit sur : https://studio.edgeimpulse.com

## IV. Connexion de l'arduino Nano RP 2040 à edge impulse :

1. Connecter la carte au PC via USB
2. Ouvrir un terminal (CMD, PowerShell ou terminal bash)
3. Lancer la commande suivante : "edge-impulse-data-forwarder" ou "edge-impulse-data-forwarder --frequency 1"

A noter : L'option --frequency 1 force un échantillon/seconde (utile pour signaux FFT déjà prétraités localement).

4. Sélectionner le port série (ex : COM4/Nano RP 2040)
5. Le terminal affiche alors les informations de connexion, la détection automatique des champs, et la confirmation de connexion à Edge Impulse, comme illustré ci-dessous :

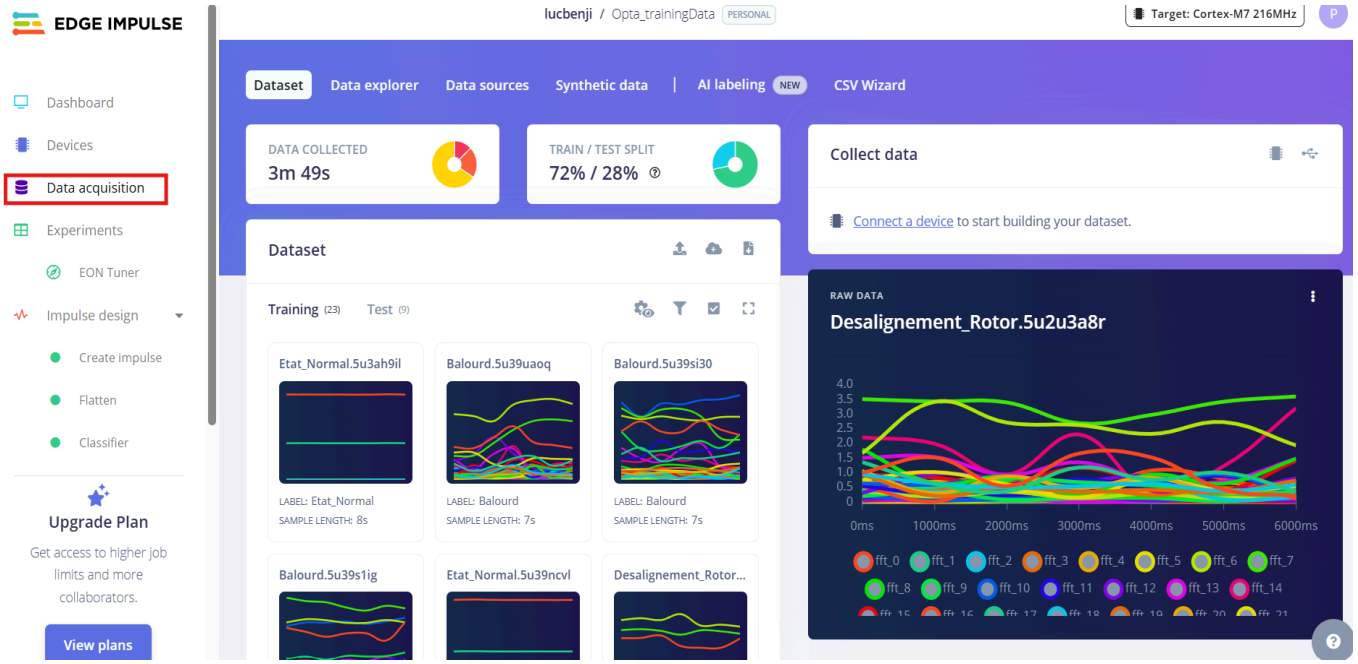
```
C:\WINDOWS\system32\cmd.

C:\Users\kaman>edge-impulse-data-forwarder --frequency 1
Edge Impulse data forwarder v1.32.1
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

? Which device do you want to connect to? (🔍 type to search) COM4
[SER] Connecting to COM4
[SER] Serial is connected (00:93:15:50:1C:78:A5:70)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com
[SER] Detecting data frequency...
[SER] Detected data frequency: 0.8Hz
[SER] Overriding frequency to 1Hz (via --frequency)
[WS ] Device "RP2040" is now connected to project "Opta_trainingData". To connect to another project, run `edge-impulse-
data-forwarder --clean`.
[WS ] Go to https://studio.edgeimpulse.com/studio/707024/acquisition/training to build your machine learning model!
[WS ] Incoming sampling request {
  path: '/api/training/data',
  label: 'Moteur_au_repos',
  length: 10000,
  interval: 1000,
  hmacKey: 'ae6588f8ed7c2b7c8e4a1410951958e3',
  sensor: 'Sensor with 64 axes (fft_0, fft_1, fft_2, fft_3, fft_4, fft_5, fft_6, fft_7, fft_8, fft_9, fft_10, fft_11, ff
t_12, fft_13, fft_14, fft_15, fft_16, fft_17, fft_18, fft_19, fft_20, fft_21, fft_22, fft_23, fft_24, fft_25, fft_26, ff
t_27, fft_28, fft_29, fft_30, fft_31, fft_32, fft_33, fft_34, fft_35, fft_36, fft_37, fft_38, fft_39, fft_40, fft_41, ff
t_42, fft_43, fft_44, fft_45, fft_46, fft_47, fft_48, fft_49, fft_50, fft_51, fft_52, fft_53, fft_54, fft_55, fft_56, ff
t_57, fft_58, fft_59, fft_60, fft_61, fft_62, fft_63)',
```

➔ On voit ici que le projet Edge Impulse actif est `Opta_trainingData`, que le label utilisé est `Moteur_au_repos`, et que 64 champs FFT sont bien transmis au serveur.

6. Une fois connecté, les données sont automatiquement envoyées dans l'onglet **"Data acquisition"** de ton projet Edge Impulse.
7. Les données sont maintenant visibles dans "Data Acquisition" sur Edge Impulse et l'interface se présente comme suit :



## V. Acquisition des données vibrations transformés en FFT

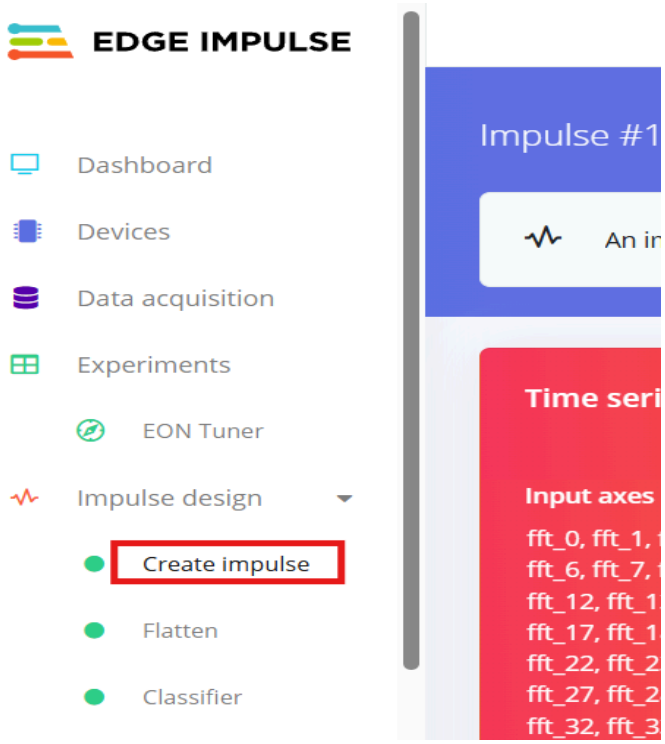
- Le traitement du signal est effectué **localement sur la carte Arduino** :
  - Acquisition des données brutes axe Z
  - Filtrage passe-bas
  - Fenêtrage de Hamming
  - FFT (128 points) → spectre sur 64 valeurs FFT
- Ces valeurs FFT sont envoyées via port série sous forme CSV ligne par ligne

## VI. Étiquetage et organisation des données

- Chaque spectre FFT doit être envoyé avec un label clair et nous avons choisi les trois labels décrivant les potentiels pannes : `Etat_Normal`, `Balourd`, `Desalignement`
- Edge Impulse répartit automatiquement les données en "**Training**" et "**Testing**"
- Bonnes pratiques :
  - Pas d'espaces ni caractères spéciaux dans les labels
  - Classes équilibrées en nombre d'échantillons

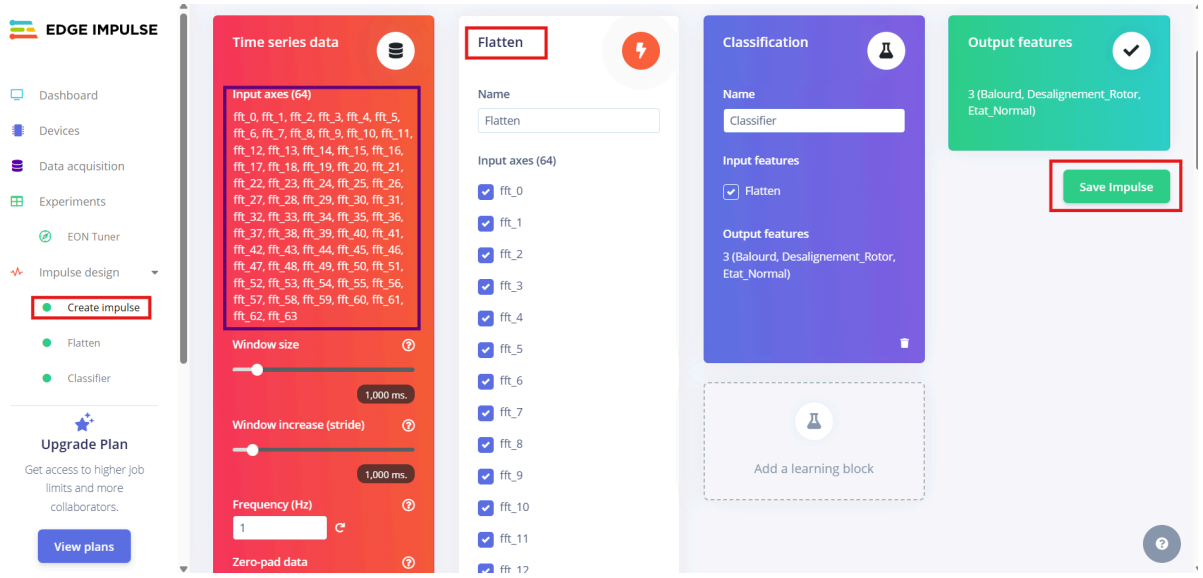
## VII. Création du pipeline (Impulse)

1. Aller dans l'onglet "Create Impulse"



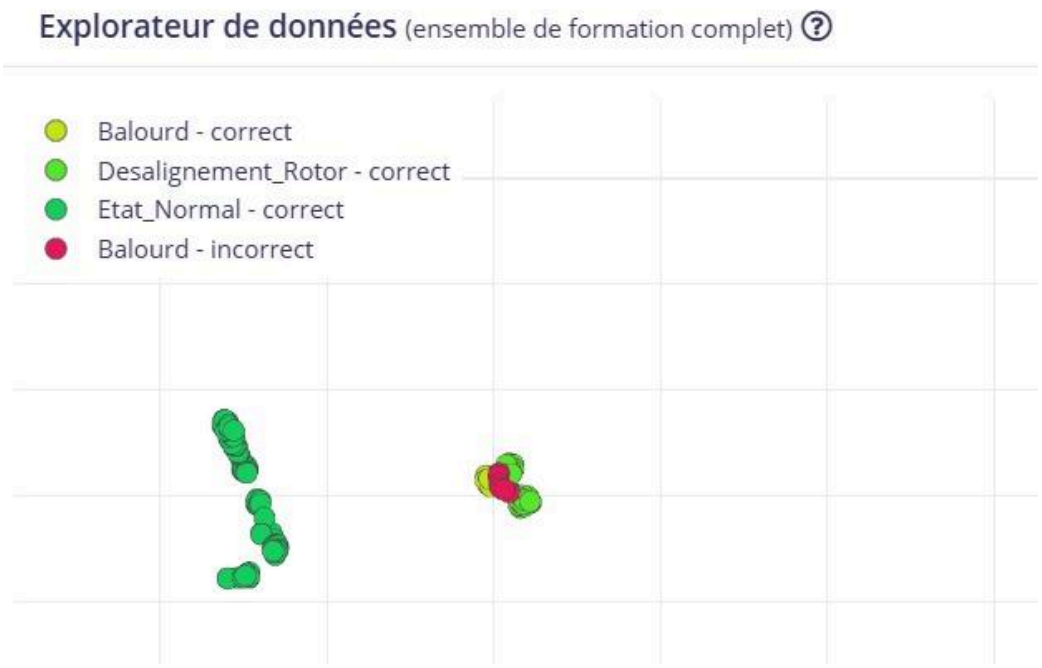

2. Ajouter les blocs suivants :
   - **Input (64 values)**
   - **Flatten** : aplatit les données
   - **Statistical Features** : extrait 7 caractéristiques/stat par valeur FFT (448 features totales)
   - **NN Classifier** : réseau de neurones dense (MLP)

3. Sauvegarder le pipeline



## VIII. Génération des caractéristiques (feature extraction)

- Aller dans "Generate features"
- Lancer la génération sur les échantillons
- Visualiser la projection 2D des features pour vérifier la séparation des classes




## IX. Entraînement du modèle IA

➔ Aller dans "NN Classifier"

➔ Régler les paramètres :

- Epochs : 50
- Batch size : 32
- Learning rate : auto

➔ Lancer l'entraînement

➔ Observer les courbes de perte (loss) et précision (accuracy)

## X. Évaluation et test du modèle

➔ Aller dans "**Model testing**"

➔ Tester le modèle sur le set de test

➔ Consulter :

- Taux de précision
- Matrice de confusion
- Cas mal classés

## XI. Déploiement sur microcontrôleur (Arduino Opta)

➔ Aller dans l'onglet "Deployment"

➔ Choisir le format **Arduino Library**

➔ Cliquer sur "Build" puis télécharger la bibliothèque `.zip`

➔ Importer cette librairie dans l'IDE Arduino : *Sketch > Include Library > Add .ZIP Library...*

➔ Intégrer le code dans un sketch (voir annexe)

➔ Utiliser `run_classifier(&signal, &result);` pour lancer l'inférence sur les doées FFT reçues

## XII. Annexe technique : Code et librairie

### ➔ Sketch Arduino principal (Opta)

Le fichier `Opta_Wifi_Edge.ino` contient le code permettant :

- De recevoir les données FFT depuis le Nano RP2040
- De les interpréter et lancer la classification via `run_classifier()`
- De transmettre le résultat par port série ou sur le réseau local

### ➔ Librairie Edge Impulse

Le fichier `ei-opta_trainingdata-arduino-1.0.6.zip` est la librairie générée automatiquement par Edge Impulse. Elle contient :

- Le modèle IA optimisé (quantifié)
- Le header avec les fonctions d'inférence (`run_classifier`, `ei_impulse_result`, etc.)

A noter : Importer cette bibliothèque dans l'IDE Arduino avant de compiler ton sketch final.

## XIII. Astuces, bonnes pratiques et pièges à éviter

- Bien fixer le capteur Nanoconnect RP 2040 sur le moteur
- Tester dans un environnement proche des conditions réelles
- Vérifier les labels avant acquisition
- Supprimer les valeurs anormales
- Toujours utiliser la quantification en `int8` pour alléger le modèle

## XIV. Perspectives d'amélioration

- Ajouter des données vibratoires sur d'autres axes (X, Y)
- Étendre les classes : bruit, faux positifs, défauts plus complexes
- Intégration dans l'environnement MES Lina Pro
- Évolution vers détection multimodale (vibration + son + énergie)

Pour en savoir plus sur la plateforme cliquer sur https://docs.edgeimpulse.com/docs