**Documentation du Projet TinyML**

**Présentation**

Bienvenue dans le projet TinyML ! 🎯 Ce projet a pour objectif de développer des solutions d’intelligence embarquée pour deux applications principales : L**a classification des vibrations** 🌀 sur une carte Arduino Nano 33 BLE Sense pour détecter différentes intensités de vibrations.

Ce document vous guidera à travers toutes les étapes pour comprendre, installer et utiliser ce projet passionnant. 🚀

**Structure du Projet**

TinyML\_project/

│

├── PARTIE\_1\_ClassificationVibrations/

│ ├── 1-DataGeneration\_Arduino/ # Collecte des données via Arduino

│ ├── 2-Training/ # Entraînement du modèle TinyML

│ │ ├── dataset/ # Données d'entraînement

│ │ ├── notebooks/ # Notebooks Jupyter

│ │ ├── models/ # Modèle enregistré

│ ├── 3-Inference\_Arduino/ # Code Arduino pour inférence

│ ├── doc/ # Documentation

│

└── README.md # Explication générale du projet

**Partie 1 : Classification des Vibrations**

Les vibrations sont partout autour de nous, mais comment peut-on les détecter et les analyser de manière intelligente ? Cette partie du projet utilise un capteur IMU intégré à un Arduino Nano 33 BLE Sense pour classer différents types de vibrations et permettre une prise de décision automatique.

**1. Acquisition des Données**

L'objectif ici est de récupérer des données brutes provenant du capteur IMU (accélération sur les axes X, Y et Z). Ces données seront ensuite utilisées pour entraîner un modèle de machine learning capable de reconnaître les vibrations spécifiques.

* **Matériel utilisé** : Arduino Nano 33 BLE Sense.
* **Code Arduino** : data\_generation.ino

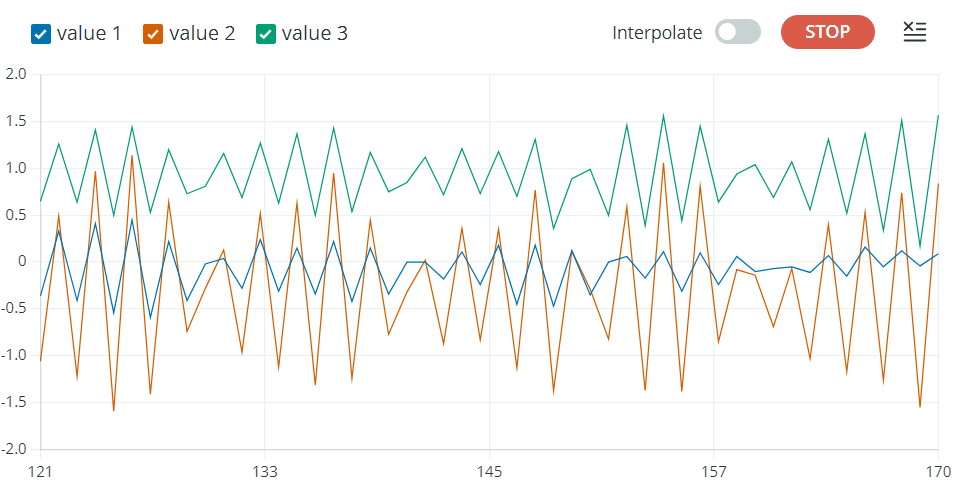
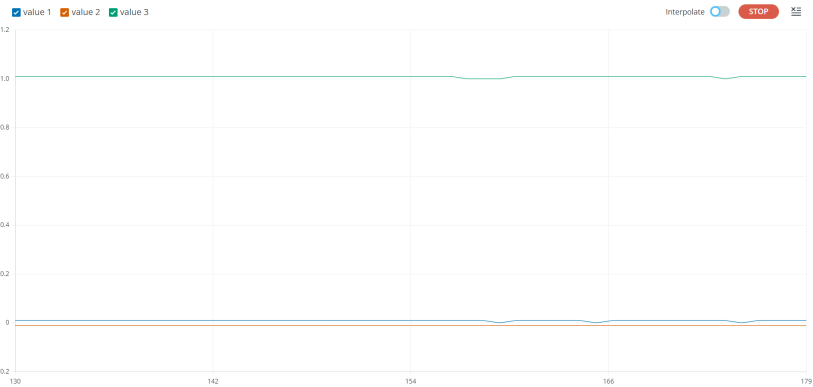
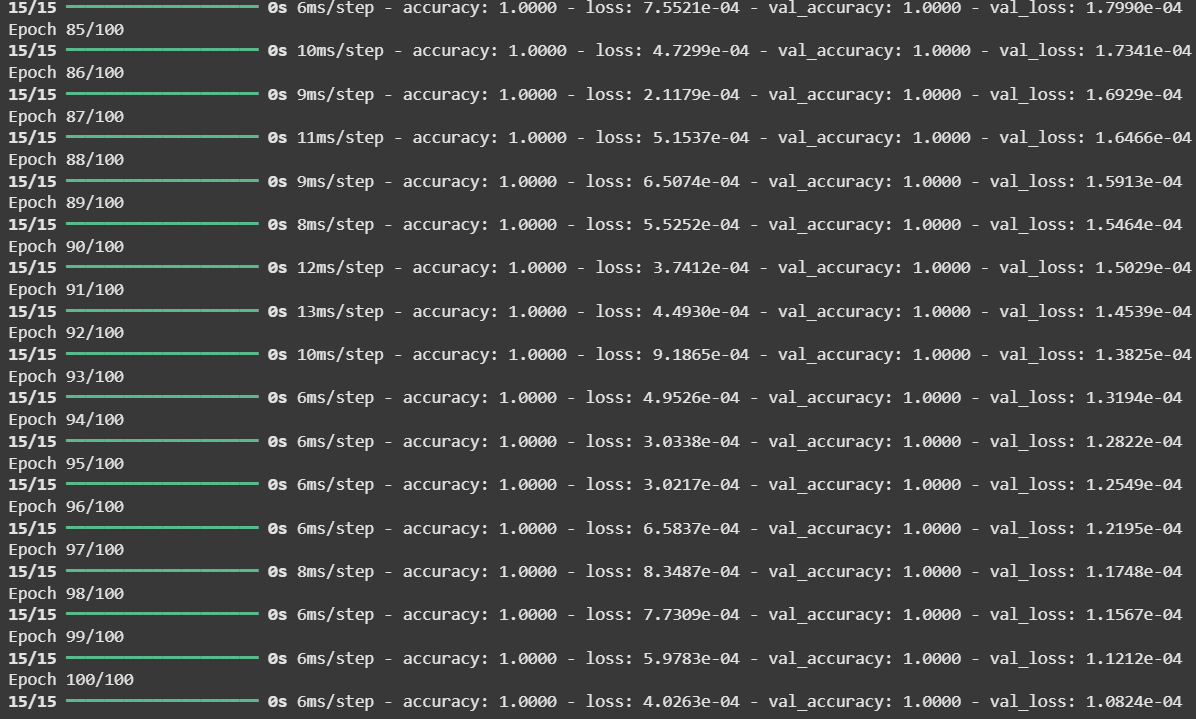
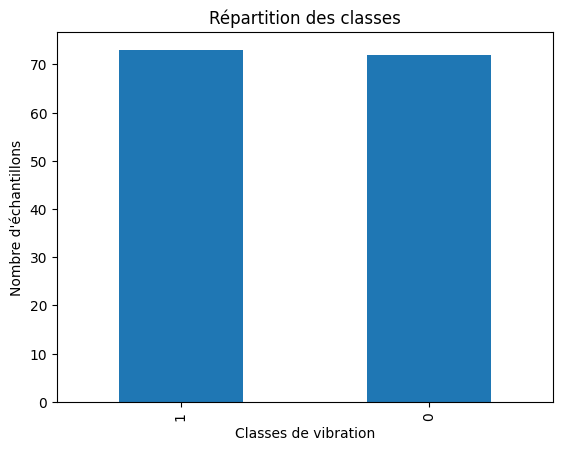


Figure :Etat du système en stabilité

Figure : Etat du système avec les vibrations

**2. Entraînement du Modèle**

Maintenant que nous avons des données, nous allons les utiliser pour entraîner un modèle de machine learning. Ce modèle sera capable de distinguer différentes vibrations.

* **Dataset utilisé** : vibrations\_dataset.csv
* **Notebook d'entraînement** : training\_vibrations.ipynb
* **Modèle obtenu** : vibrations\_model.tflite

Ce modèle a atteint une précision de 100% sur les données d'entraînement et de validation, avec une perte très faible (de l'ordre de 10⁻⁴). Cela indique que le modèle a parfaitement appris à classer les vibrations.

**📌 3. Inférence sur Arduino**

Une fois le modèle entraîné, nous allons le déployer sur notre Arduino pour qu’il puisse prédire en temps réel le type de vibration détecté.

* **Code Arduino pour inférence** : inference\_vibrations.ino
* **Affichage des résultats** sur le moniteur série.

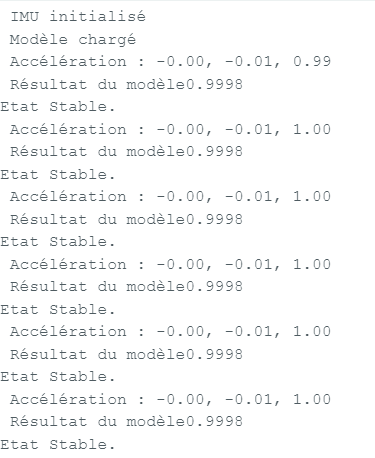
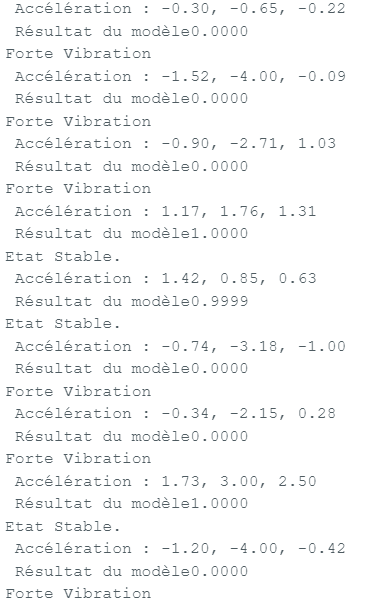


Figure : systeme en vibration

Figure : systeme stable

Le modèle détecte correctement l'état stable, mais il a parfois des difficultés à identifier les vibrations, les classant à tort comme stables. Cela suggère un besoin d'ajuster le seuil de classification ou d'améliorer le jeu de données pour mieux capturer les cas de vibrations. L'état stable, en revanche, est toujours bien détecté