

ENSEIRB-MATMECA



**TINYML :
IA EMBARQUEE SUR
MICROCONTROLEUR**

Patrice KADIONIK
kadionik.enseirb-matmeca.fr

TABLE DES MATIERES

1.	<i>But des travaux pratiques.....</i>	3
2.	<i>Présentation de l'environnement.....</i>	4
2.1.	<i>Carte cible tflite-duino.....</i>	4
2.2.	<i>Environnement de développement</i>	6
3.	<i>EX 1 : contrôle direct sur microcontrôleur d'une sortie en fonction d'une entrée</i>	8
4.	<i>EX 2 : création d'un modèle d'IA avec Tensorflow/Keras.....</i>	11
5.	<i>EX 3 : génération du modèle Tensorflow Lite for Micro</i>	13
6.	<i>EX 4 : contrôle par IA sur microcontrôleur d'une sortie en fonction d'une entrée</i>	14
7.	<i>EX 5 : visualisation de l'IA sur le traceur série.....</i>	16
8.	<i>Conclusion</i>	17
9.	<i>Références.....</i>	18
10.	<i>Annexe 1 : schéma électronique de la carte tflite-duino</i>	19

1. BUT DES TRAVAUX PRATIQUES

Ces Travaux Pratiques ont pour but de présenter la mise en œuvre de *TinyML* (*Tiny Machine Learning*) sur microcontrôleur.

TinyML est apparu vers 2019 pour implanter de l'intelligence artificielle embarquée sur carte possédant un microcontrôleur. La consommation en courant est ici en dizaines de mA et non pas en centaines d'Ampère comme dans un *data center* avec des IA comme ChatGPT et consorts...

TinyML est donc en soi utile car on est dans un cas d'*edge computing* et *TinyML* est plus respectueux des ressources de notre planète notamment en termes de consommation électrique... encore faut-il que l'application développée soit utile et non futile pour l'humanité...

Il s'agit plus précisément de voir la mise en œuvre de *Tensorflow Lite for Micro* (TFLM) qui est un canevas de programmation (*framework*) développé par Google pour mettre en œuvre *TinyML* sur un microcontrôleur.

La mise en œuvre de TFLM sera fera à l'aide d'une carte Arduino Nano 33 BLE Sense (version 2) pour la partie matérielle et de l'atelier de développement logiciel IDE (*Integrated Design Environment*) *Arduino IDE*.

L'exemple que nous allons aborder est un peu le « *Hello World* » avec TFLM.

Il s'agit de faire l'acquisition d'une tension analogique à l'entrée de la carte Arduino Nano 33 BLE Sense puis d'injecter sa valeur numérique comme entrée d'une inférence d'IA qui nous donnera en sortie une valeur numérique correspondant à un sinus. Cette valeur sera appliquée ensuite comme sortie analogique pour piloter l'intensité d'une led.

La mise en œuvre de TFLM a fait l'objet d'un sujet de « projets avancés » de l'option Systèmes Embarqués SE. Je tiens ainsi à remercier Salma Arouch, Zakaria Tber, Elouan Thierry et Maysa Msallem ainsi que Mohamed El Hassani, Nour Louati, Zakaria Souhail et Jawad Ali de la promotion SE 2024-2025 pour leur travail et leur contribution à l'amélioration constante de l'enseignement de l'option SE...

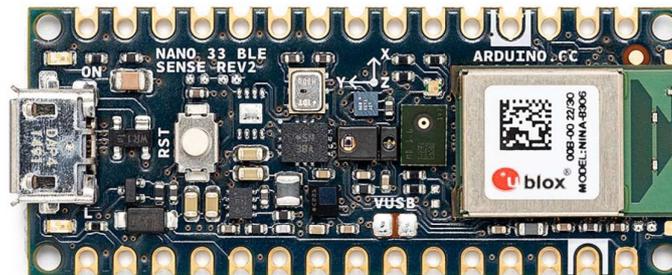
Mots clés : *Machine Learning*, Intelligence Artificielle, *Tiny Machine Learning*, *TinyML*, *Tensorflow*, *Keras*, *Tensorflow Lite*, *Tensorflow Lite for Micro*, TFL, TFLM, Arduino Nano 33 BLE Sense, *Arduino IDE*, *Spyder*

2. PRESENTATION DE L'ENVIRONNEMENT

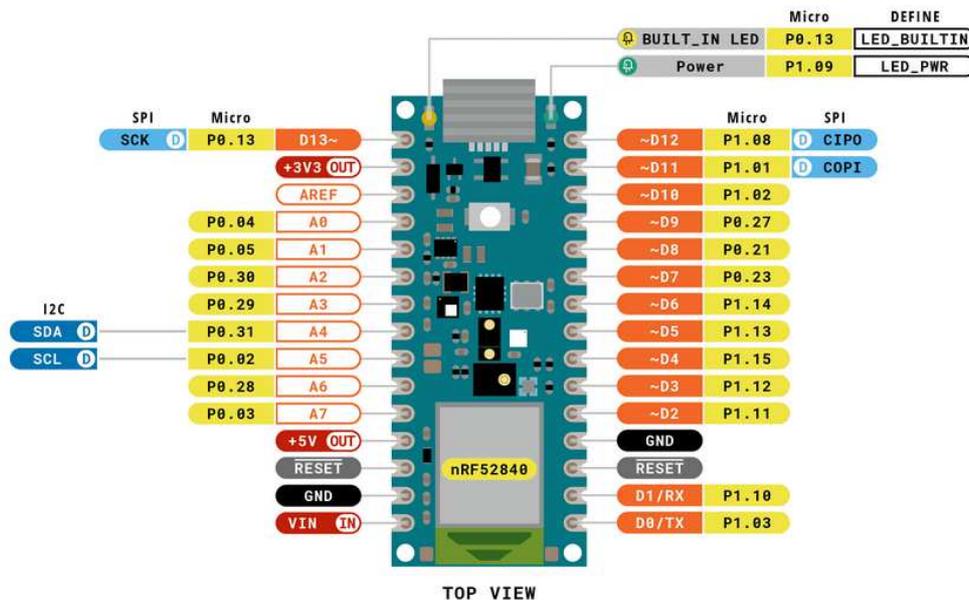
2.1. Carte cible tflite-duino

La carte cible tflite-duino est une carte « maison » construite autour d'une carte Arduino Nano 33 BLE Sense. Elle est complétée par un potentiomètre linéaire d'1 kΩ connecté sur l'entrée analogique A0 de la carte Arduino Nano 33 BLE Sense.

La carte Arduino Nano 33 BLE Sense est une carte bon marché largement utilisée pour le DIY (*Do It Yourself*) afin de développer de petits systèmes embarqués ou des objets connectés. Intégrant de multiples capteurs, elle permet de mettre en œuvre facilement de l'IA embarquée.



Carte cible Arduino Nano 33 BLE Sense



Brochage de la carte cible Arduino Nano 33 BLE Sense

La carte Arduino Nano 33 BLE Sense possède ainsi les éléments suivants :

- Un SoC (*System on Chip*) Nordic nRF52840 avec un processeur ARM Cortex-M4 à 64 MHz.
- 256 ko de RAM et 1 Mo de mémoire Flash.
- 1 port USB.
- 8 entrées analogiques.
- 14 E/S GPIO.
- Bluetooth BLE.
- Accéléromètre et gyroscope 3 axes BMI270.
- Magnétomètre 3 axes BMM150.
- Microphone MP34DT06JTR.
- Capteur de geste, de lumière ambiante et de proximité APDS9960
- Capteur de pression et baromètre LPS22HB.
- Capteur de température et humidité HS3003.
- 1 UART, 1 bus I2C et 1 bus SPI.

La photo suivante présente la carte cible tflite-duino à base d'une carte cible Arduino Nano 33 BLE Sense.



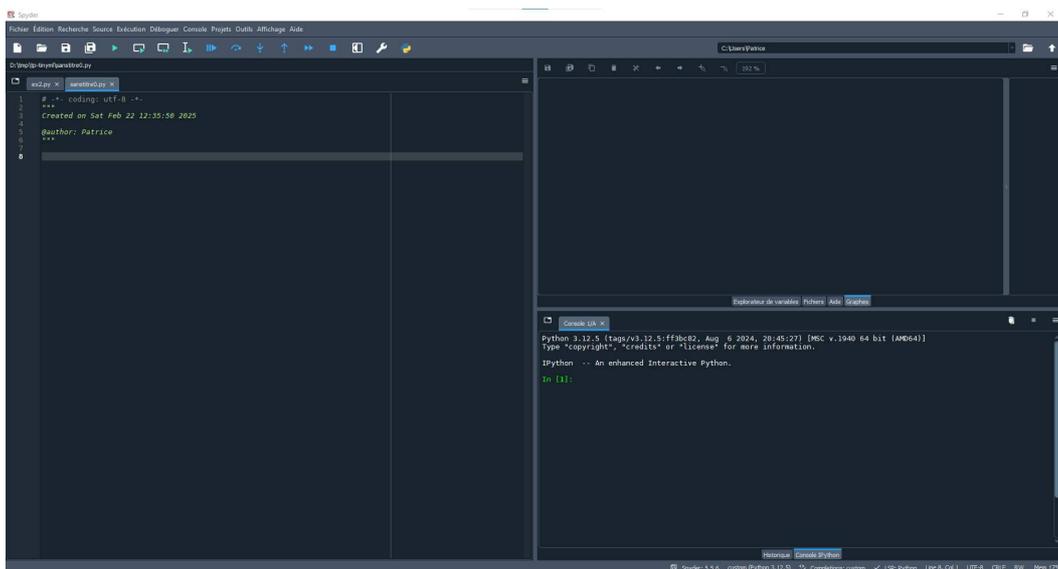
Carte cible tflite-duino

2.2. Environnement de développement

Le développement TFLM se fait en plusieurs étapes :

- Développement d'un modèle d'IA en langage Python avec le *framework* *Tensorflow/Keras* soit de *Machine learning* soit de *Deep Learning* : récupération ou génération d'un jeu de données (*dataset*), définition du modèle, entraînement du modèle, test du modèle et validation du modèle. Sauvegarde de l'inférence *Tensorflow*. L'IDE utilisé est *Spyder*.
- Transformation de l'inférence *Tensorflow* en inférence *Tensorflow Lite*. Optimisation de l'inférence (quantification, *pruning*...). L'IDE utilisé est *Spyder*.
- Transformation de l'inférence optimisée en fichier .h (tableau de valeurs hexadécimales).
- Développement d'un programme C/C++ mettant en œuvre l'inférence avec le *framework* TFLM. L'IDE utilisé est *Arduino IDE*.

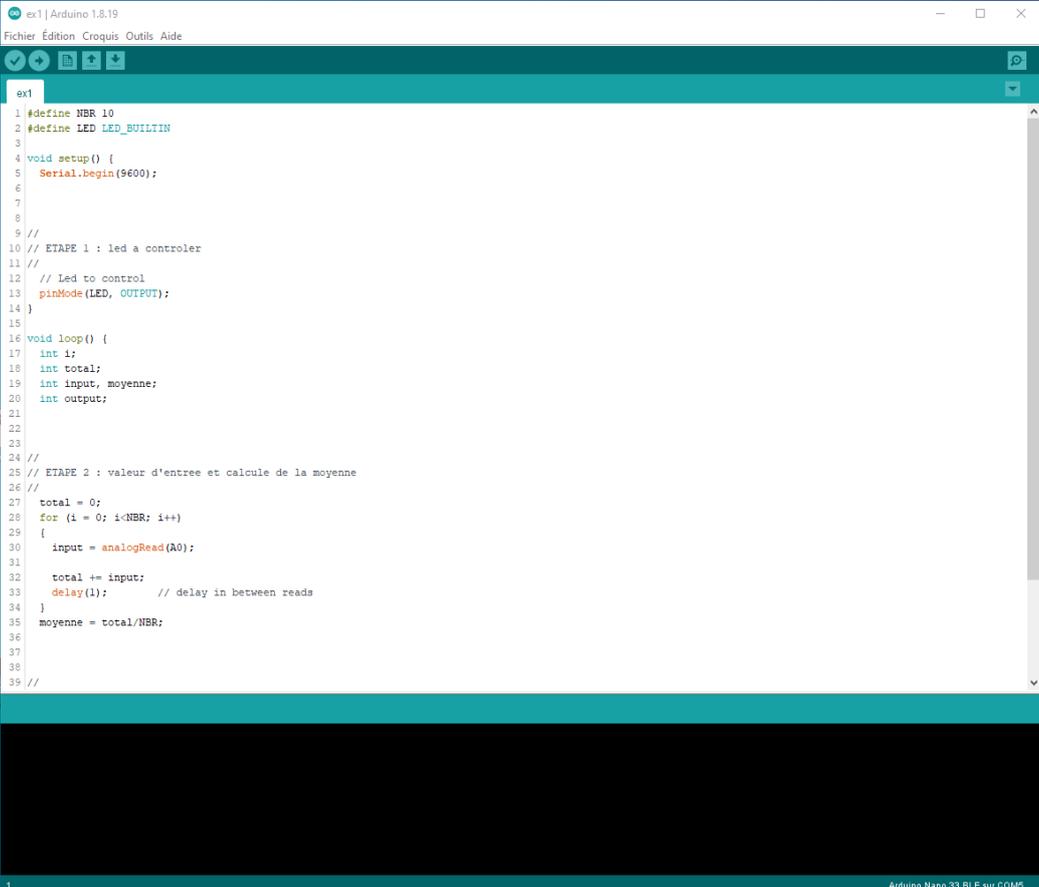
L'environnement *Spyder* est présenté ci-après. Il est simple d'utiliser et intuitif conçu pour des développements en langage Python.



IDE *Spyder*

L'environnement *Arduino IDE* permet de développer des programmes en langage C/C++ pour des cartes Arduino.

La prise en main est simple et intuitive.



```
ex1 | Arduino 1.8.19
Fichier Edition Croquis Outils Aide

ex1
1 #define NBR 10
2 #define LED LED_BUILTIN
3
4 void setup() {
5   Serial.begin(9600);
6
7
8
9 //
10 // ETAPE 1 : led a controler
11 //
12 // Led to control
13 pinMode(LED, OUTPUT);
14 }
15
16 void loop() {
17   int i;
18   int total;
19   int input, moyenne;
20   int output;
21
22
23
24 //
25 // ETAPE 2 : valeur d'entree et calcul de la moyenne
26 //
27   total = 0;
28   for (i = 0; i<NBR; i++)
29   {
30     input = analogRead(A0);
31
32     total += input;
33     delay(1); // delay in between reads
34   }
35   moyenne = total/NBR;
36
37
38
39 //
```

1 Arduino Nano 33 BLE sur COM5

IDE *Arduino IDE*

3. EX 1 : CONTROLE DIRECT SUR MICROCONTROLEUR D'UNE SORTIE EN FONCTION D'UNE ENTREE

Le but de ce TP est de lire la tension analogique appliquée sur l'entrée analogique A0 de la carte tflite-duino que contrôle le potentiomètre via un pont diviseur de tension qui sera appliquée après transformation sur la sortie analogique de la carte tflite-duino qui contrôle la led de la carte.

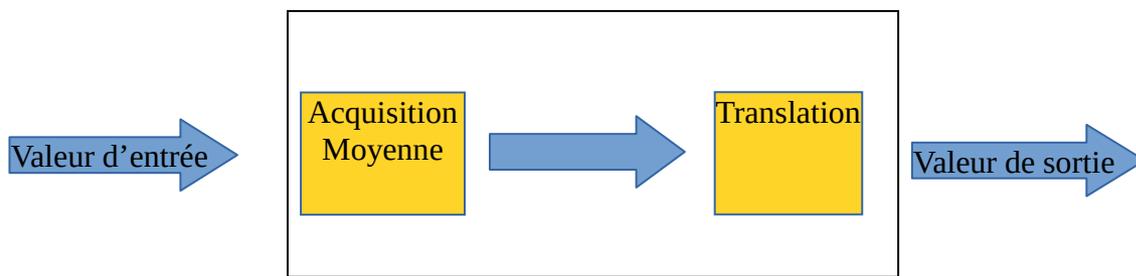
Cet exercice simple pose les bases de notre projet d'IA embarquée.

Nous avons :

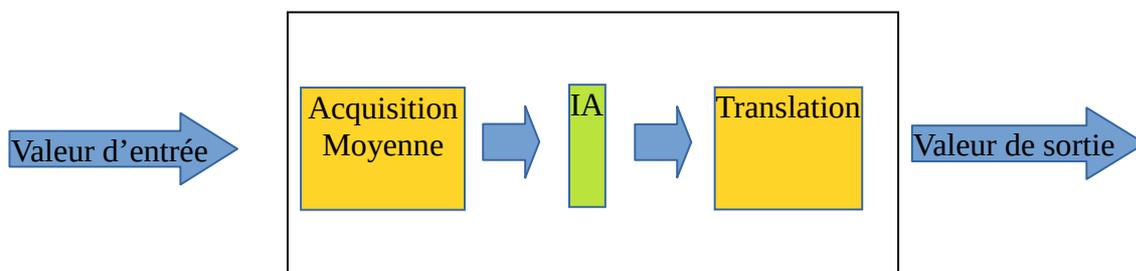
- Une grandeur d'entrée : tension analogique.
- Une grandeur de sortie : tension analogique.

Nous allons entre les deux appliquer une transformation exécutée par la carte tflite-duino :

- Sans IA : acquisition de 10 grandeurs d'entrée, calcul de la valeur moyenne, translation de la valeur moyenne et affecction comme grandeur de sortie.
- Avec IA : acquisition de 10 grandeurs d'entrée, calcul de la valeur moyenne, exécution d'une inférence prenant en entrée cette valeur moyenne et renvoyant son sinus, translation de la valeur issue de l'inférence et affecction comme grandeur de sortie.



Contrôle sans IA embarquée



Contrôle avec IA embarquée

L'IA que nous allons développer sera entraînée pour donner la valeur du sinus d'une grandeur d'entrée.

On pourrait plus simplement et plus efficacement appeler la fonction sinus de la bibliothèque du compilateur C. Ce n'est bien sûr pas le but de ces TP.

Mais l'on comprend alors bien l'intérêt de l'IA qui permettra de remplacer toute relation complexe sur une ou plusieurs grandeurs d'entrée par une inférence de *Machine Learning* ou de *Deep Learning* suivant le besoin avec bien sûr des limitations à connaître a priori (précision du calcul, performances Temps Réel...).

La tension analogique A0 est convertie en interne par un convertisseur analogique/numérique de 10 bits en une valeur numérique comprise dans l'échelle [0, 1023].

On fera l'acquisition de NBR (10) valeurs numériques et l'on en tirera ensuite la moyenne comme valeur finale pour la grandeur d'entrée.

Cette valeur finale sera enfin translatée ensuite dans l'échelle [0, 255] pour être appliquée sur la sortie analogique qui pilote la led de la carte tflite-duino (LED_BUILTIN). C'est la led de couleur jaune près du connecteur USB de la carte tflite-duino.

Cette valeur finale sera aussi envoyée sur le port série.

On utilisera pour cet exercice l'IDE *Arduino IDE*.

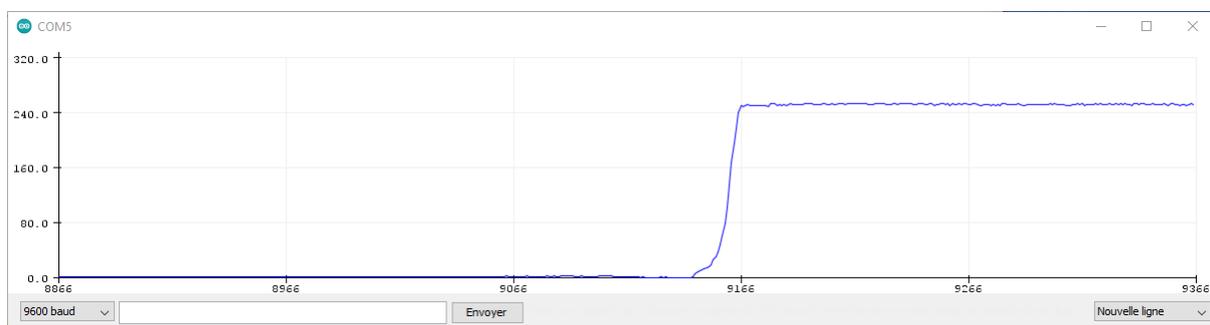
Par la suite, on adoptera les conventions suivantes :

Commande Linux PC hôte :

```
host% commande Linux
```

- Démarrer le PC sous Linux. Se connecter sous le nom **se01**, mot de passe : **se01** ☺ pour le groupe 1 et sous le nom **se02**, mot de passe : **se02** ☺ pour le groupe 2.
- Se placer dans son répertoire de travail :
host% cd
- Dans son répertoire à son nom, recopier le fichier `tp-TinyML.tgz` sous `~kadionik/` :
host% cp /home/kadionik/tp-TinyML.tgz .
- Décompresser et installer le fichier `tp-TinyML.tgz` :
host% tar -xvzf tp-TinyML.tgz
- Se placer ensuite dans le répertoire `TinyML/`. **L'ensemble du travail sera réalisé à partir de ce répertoire ! Les chemins seront donnés par la suite en relatif par rapport à ce répertoire...**
host% cd TinyML

- Se placer ensuite dans le répertoire `Arduino/ex1/`. Ouvrir le projet Arduino `ex1.ino` avec *Arduino IDE* :
host% cd TinyML
host% cd Arduino/ex1
- **Etape 1.** Préciser le rôle de cette étape. Modifier le code source pour piloter la led `LED_BUILTIN` en sortie.
- **Etape 2.** Préciser le rôle de cette étape. Modifier le code source pour lire 10 échantillons de la valeur de la tension analogique de la broche d'entrée `A0` et un faire la moyenne sauvegardée dans la variable `moyenne`.
- **Etape 3.** Préciser le rôle de cette étape. Modifier le code source pour translater la variable `moyenne` de `[0, 1023]` vers `[0, 255]` et écrire la valeur obtenue `output` sur la broche de sortie `LED_BUILTIN`.
- Compiler le programmer et téléverser le dans la carte `tflite-duino`. Tester.
- Depuis *Arduino IDE*, visualiser la sortie de la liaison série via le menu `Outil > Traceur série`. Nous avons accès à un oscilloscope. Si tout va bien, une action constante sur le potentiomètre du minimum au maximum doit créer une rampe de tension sur l'oscilloscope comme montré sur la figure suivante :



Oscilloscope *Arduino IDE*

Indications :

Pour la lecture de l'entrée analogique `in`, on peut utiliser la fonction `Arduino analogRead(in)`.

Pour translater une valeur de `[0, 1023]` dans `[0, 255]`, on peut utiliser la fonction `Arduino value = map(input, 0, 1023, 0, 255)`.

Pour piloter la sortie analogique `out`, on peut utiliser la fonction `Arduino analogWrite(out, value)`.

4. EX 2 : CREATION D'UN MODELE D'IA AVEC TENSORFLOW/KERAS

On va ici créer un modèle d'IA pour générer une variable y en fonction d'une variable d'entrée x et d'une expression mathématique (ici un sinus).

- Ouvrir avec *Spyder* le fichier Python `ex2.py` :
`host% cd TinyML`
- **Etape 1.** Préciser le rôle de cette étape. A quoi servent les constantes `SAMPLES` et `EPOCH` ? Quelle distribution statistique utilise-t-on pour générer les valeurs aléatoires ? Quel est le type de x ? Entre quelles valeurs varie x ? Modifier le code source avec `SAMPLES` initialisée à une valeur supérieur à 2000 et `EPOCH` initialisée à une valeur supérieure à 500.
- Etape 2. Préciser le rôle de cette étape.
- **Etape 3.** Préciser le rôle de cette étape. Modifier le code source pour calculer le sinus de $\pi \cdot x$. Quel est le type de y ? Quelle est la relation mathématique formelle qui existe entre x et y ? Entre quelles valeurs varie y ?
- **Etape 4.** Préciser le rôle de cette étape. Modifier le code source pour ajouter un bruit de 20 % sur y .
- **Etape 5.** Préciser le rôle de cette étape. Modifier le code source pour avoir un pourcentage de données d'entraînement de 80 % Quel est alors le pourcentage de données de test ? Y-a-t-il des données de validation ? Comment s'appelle la variable qui contient le jeu de données d'entraînement ? Comment s'appelle la variable qui contient le jeu de données de test ?
- Etape 6. Préciser le rôle de cette étape. Quel type de modèle d'IA a-t-on créé ? Quel est le type de problème d'IA essaye-t-on traiter ? A quoi sert la méthode `summary()` ? Dessiner le réseau de neurones. Est-il dense ? Quelle métrique est utilisée pour la mesure de la convergence lors de l'apprentissage ?
- Etape 7. Préciser le rôle de cette étape.
- Etape 8. Préciser le rôle de cette étape.
- Etape 9. Préciser le rôle de cette étape.
- Etape 10. Préciser le rôle de cette étape. Le modèle *Tensorflow* sauvegardé est-il adapté pour l'IA embarquée ?
- Etape 11. Préciser le rôle de cette étape. Comparer la taille du modèle *Tensorflow* `model.h5` et celle du modèle *Tensorflow Lite* `model.tflite`. Le modèle *Tensorflow Lite* est-il adapté mieux pour l'IA embarquée ?

- Etape 12. Préciser le rôle de cette étape. Quelle est le nom de la variable d'entrée de l'inférence *Tensorflow Lite* ? Quelle est le nom de la variable de sortie de l'inférence *Tensorflow Lite* ? Quelle méthode Python est appelée pour invoquer l'inférence *Tensorflow Lite* ? Ce code Python est-il transposable sur un microcontrôleur avec le *framework Tensorflow Lite for Micro* ?
- Faire varier la valeur de la constante EPOCH pour voir son influence sur les valeurs prédites.

5. EX 3 : GENERATION DU MODELE TENSORFLOW LITE FOR MICRO

On va créer le modèle *Tensorflow Lite for Micro* à partir du modèle *Tensorflow Lite*. On utilise pour cela la commande Linux `xxd`.

- Générer le fichier `model.cc` à partir du fichier `model.tflite` avec la commande `xxd` :

```
host% cd TinyML
host% xxd -i model.tflite > model.cc
```
- Analyser la structure du fichier `model.cc`. Que représente le fichier `model.cc` ?
- A quoi correspond la variable `model_tflite_len` à la fin du fichier `model.cc` ?

Nous allons enfin adapter le fichier `model.cc` au *framework Tensorflow Lite for Micro* que nous allons utiliser avec *Arduino IDE*.

- Copier le fichier `model.cc` dans un fichier `model.cpp` :

```
host% cp model.cc model.cpp
```
- Editer le fichier `model.cpp` pour ajouter les modifications suivantes :
 - Le tableau `model_tflite` est renommé `g_model`
 - La variable `model_tflite_model` est renommée `g_model_len`. La valeur de cette variable est conservée (taille du tableau) :

```
#include "model.h"

// Keep model aligned to 8 bytes to guarantee aligned 64-bit accesses.
alignas(8) const unsigned char g_model[] = {

    0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
    0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00, 0x0c, 0x00, 0x00, 0x00,
    0x08, 0x00, 0x04, 0x00, 0x14, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00,
    0x98, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x00, 0xd4, 0x04, 0x00, 0x00,
    . . .
    0x10, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x0c, 0x00, 0x0b, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x04, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x09

};
const int g_model_len = . . .;
```

- Copier le fichier `model.cpp` dans le répertoire `TinyML/Arduino/ex4` :

```
host% cd TinyML
host% cp model.cpp Arduino/ex4
```

6. EX 4 : CONTROLE PAR IA SUR MICROCONTROLEUR D'UNE SORTIE EN FONCTION D'UNE ENTREE

Nous allons maintenant développer l'application *Tensorflow Lite for Micro* avec l'IDE *Arduino IDE* pour le contrôle par IA de la led de la carte tflite-duino.

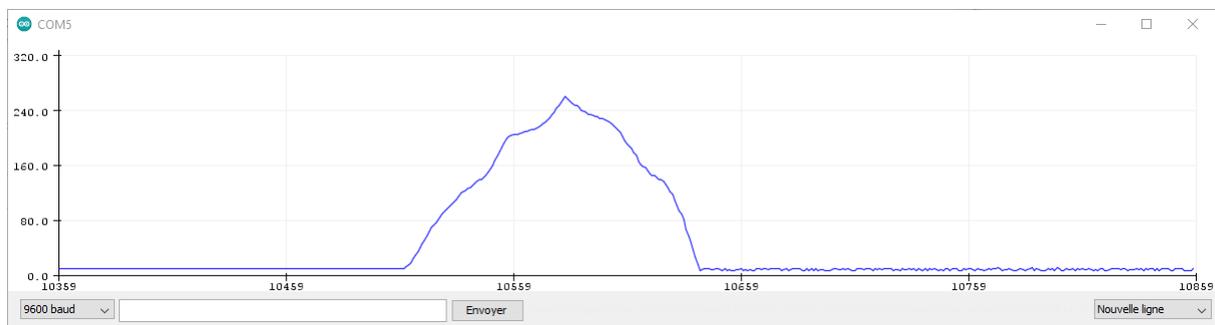
La bibliothèque TFLM utilisée est la bibliothèque C/C++ `Havard_TinyMLx`.

Le développement se fait en C/C++.

- Se placer ensuite dans le répertoire `Arduino/ex4/`. Ouvrir le projet Arduino `ex4.ino` avec *Arduino IDE* :

```
host% cd TinyML
host% cd Arduino/ex4
```
- Etape 1. Préciser le rôle de cette étape. A quoi correspondent les variables, `error_reporter`, `model`, `interpreter`, `model_input`, `model_output`, `tensor_arena` et `kTensorArenaSize`? A quoi sert la zone (tableau) `tensor_arena`? Quelle est sa taille?
- **Etape 2.** Préciser le rôle de cette étape. Modifier le code source pour piloter la led `LED_BUILTIN` en sortie.
- Etape 3. Préciser le rôle de cette étape. Quel est le lien entre les variables `error_reporter` et `micro_error_reporter`?
- Etape 4. Préciser le rôle de cette étape. Que charge-t-on en mémoire? Quel est le lien avec l'exercice 3?
- Etape 5. Préciser le rôle de cette étape. Quel est le lien entre les variables `interpreter` et `static_interpreter`?
- Etape 6. Préciser le rôle de cette étape. Où est allouée la mémoire pour le calcul sur les tenseurs?
- Etape 7. Préciser le rôle de cette étape. A quoi servent les variables `model_input` et `model_output`?
- **Etape 8.** Préciser le rôle de cette étape. Modifier le code source pour lire NBR échantillons de la valeur de la tension analogique de la broche d'entrée A0 et un faire la moyenne sauvegardée dans la variable `moyenne`.
- **Etape 9.** Préciser le rôle de cette étape. Modifier le code source pour translater la variable `moyenne` de `[0, 1023]` vers `[0, 1]` et écrire la valeur obtenue dans la variable `x_val`.
- Etape 10. Préciser le rôle de cette étape.

- Etape 11. Préciser le rôle de cette étape.
- **Etape 12.** Préciser le rôle de cette étape. Modifier le code source pour translater la variable `y_val` de `[0, 1]` vers `[0, 255]` et écrire la valeur obtenue `output` sur la broche de sortie `LED_BUILTIN`.
- Compiler le programmer et téléverser le dans la carte tflite-duino. Tester.
- Depuis *Arduino IDE*, visualiser la sortie de la liaison série via le menu Outil > Traceur série. Si tout va bien, une action constante sur le potentiomètre du minimum au maximum doit créer une arche en forme de sinusoïde sur l'oscilloscope comme montré sur la figure suivante :



Oscilloscope *Arduino IDE*

- Constater que la led passe de l'état éteint – allumé – éteint lors d'une action constante sur le potentiomètre du minimum au maximum.

7. EX 5 : VISUALISATION DE L'IA SUR LE TRACEUR SERIE

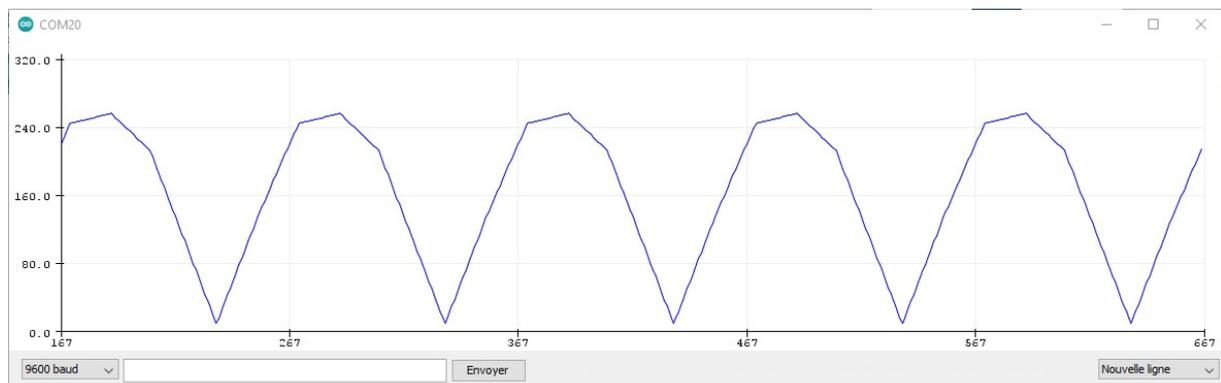
Nous n'allons plus utiliser le potentiomètre et la led de la carte tflite-duino.

Nous allons faire varier continuellement la variable d'entrée de l'inférence `x_val` entre 0 et 1 et récupérer à chaque fois la variable de sortie `y_val` de l'inférence. La valeur de la variable `y_val` sera ensuite envoyée sur le port série pour pouvoir utiliser l'oscilloscope *d'Arduino IDE*.

- Recopier le répertoire `Arduino/ex4/` dans le répertoire `Arduino/ex5/` :

```
host% cd TinyML
host% cd Arduino/ex4
host% cp -r ex4 ex5
```
- Renommer le projet Arduino `ex4.ino` en `ex5.ino` :

```
host% mv ex5/ex4.ino ex5/ex5.ino
```
- Ouvrir le projet Arduino `ex5.ino` avec *Arduino IDE* et modifier le fichier source pour répondre à l'exercice.
- Depuis *Arduino IDE*, visualiser la sortie de la liaison série via le menu `Outil > Traceur série`. Si tout va bien, on doit obtenir la figure suivante :



Oscilloscope *Arduino IDE* (l'ordonnée a été translatée de l'intervalle [0, 1] vers [0, 255])

8. CONCLUSION

On a pu voir la mise en œuvre de l'IA embarquée sur microcontrôleur en utilisant le *framework Tensorflow Lite for Micro*.

Le développement d'un modèle d'IA embarquée reste classique et ne diffère en rien avec le développement d'un modèle d'IA classique avec le langage Python.

Le modèle ainsi créé doit ensuite être adapté en utilisant des techniques de quantification ou d'autres techniques qu'offre le *framework Tensorflow Lite*.

Le modèle Tensorflow Lite ainsi créé est adapté à la programmation avec le *framework Tensorflow Lite for Micro* qui utilise le langage C/C++.

La mise en œuvre de *Tensorflow Lite for Micro* a enfin été étudiée sur une carte cible Arduino Nano 33 BLE Sense.

9. REFERENCES

- Carte Arduino Nano 33 BLE Sense (V2) : <https://docs.arduino.cc/hardware/nano-33-ble-sense/>
- Tensorflow Lite for Micro : <https://www.tensorflow.org/lite/microcontrollers?hl=fr>
- Hands-on TinyML. R. Banerjee. Editions BPB Online. 2023
- Consommation d'énergie des data centers : chiffres en 2025 : <https://opera-energie.com/consommation-energie-datacenter/>

10. ANNEXE 1 : SCHEMA ELECTRONIQUE DE LA CARTE TFLITE-DUINO

CARTE TFLITE-DUINO

enseirb/pk/2025

