des données de capteurs aux **modèles d'apprentissage automatique**

Détection de gestes avec Edge Impulse et un accéléromètre

Koen Vervloesem (Belgique)

Nous sommes habitués à interagir avec nos appareils via des claviers, des souris ou des écrans tactiles. Toutefois, il peut être intéressant de rechercher des alternatives à ces interfaces classiques. La reconnaissance des gestes représente une telle alternative prometteuse. Pour cela, nous pouvons utiliser Edge Impulse afin de développer un modèle d'apprentissage automatique qui détecte les gestes en exploitant les données provenant d'un accéléromètre.

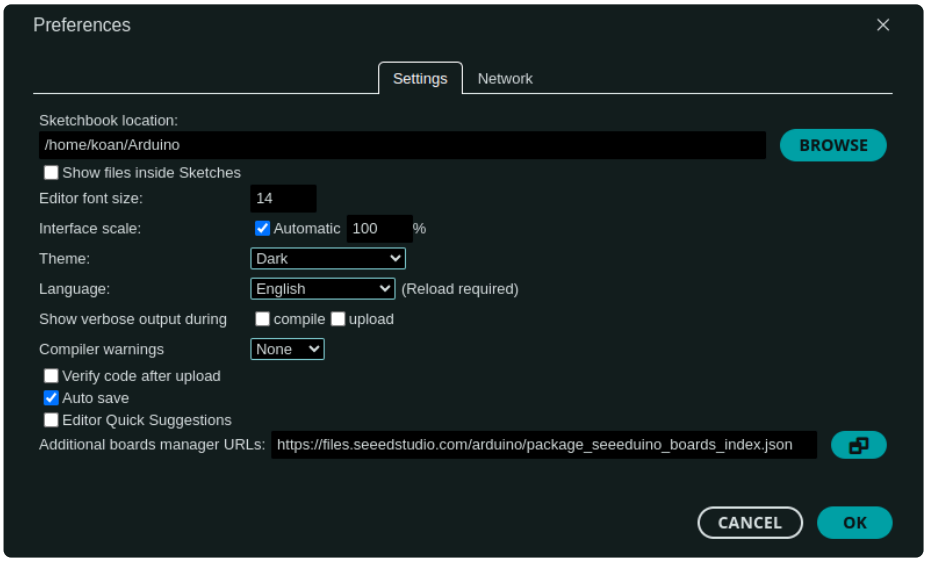

Figure 1. Ajouter l'URL des cartes Seeed à l'EDI Arduino.

L'une des méthodes de reconnaissance gestuelle utilise une caméra qui enregistre les mouvements de la main [1]. Toutefois, une méthode alternative consiste à intégrer un accéléromètre dans un appareil porté au poignet ou tenu en main, et il est possible d'utiliser l'apprentissage automatique pour identifier les gestes à partir des données de mouvement, réduisant ainsi le volume de données nécessaires au traitement.

Dans cet article, je vais créer un système de reconnaissance gestuelle fonctionnant sur une carte de microcontrôleur XIAO nRF52840 Sense de Seeed Studio, équipée d'un accéléromètre. En utilisant l'EDI Arduino [2], je développerai d'abord un croquis Arduino pour lire les données du capteur d'accéléromètre. Je réaliserai les gestes spécifiques que je souhaite que le système reconnaisse et je transmettrai ces données à la plateforme Edge Impulse [3] pour entraîner le modèle.

Sur Edge Impulse, je créerai un modèle d'apprentissage automatique qui reconnaît les gestes. Ce modèle sera ensuite déployé sur le microcontrôleur en tant que modèle TensorFlow Lite qui peut être facilement utilisé comme bibliothèque Arduino. Vous pouvez ensuite vous appuyer sur ce modèle pour contrôler votre ordinateur par gestes.

Configuration de l'EDI Arduino

Tout d'abord, installez l'EDI Arduino, disponible pour les systèmes d'exploitation Windows, macOS et Linux. La carte Seeed nécessite un package de carte spécifique pour fonctionner avec l'EDI Arduino. Accédez à *Fichier / Préférences* et ajoutez l'URL *https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json* au champ *URL du gestionnaire de cartes supplémentaires* (**figure 1**). Cliquez sur *OK* pour charger les informations nécessaires sur le package de cartes depuis ce fichier en ligne.

Ensuite, ouvrez le *gestionnaire de cartes* en cliquant sur l'icône de la carte dans la barre latérale gauche et recherchez « seeed nrf ». Sélectionnez le paquet nommé Seeed nRF52 mbed-enabled Boards et cliquez sur Installer.

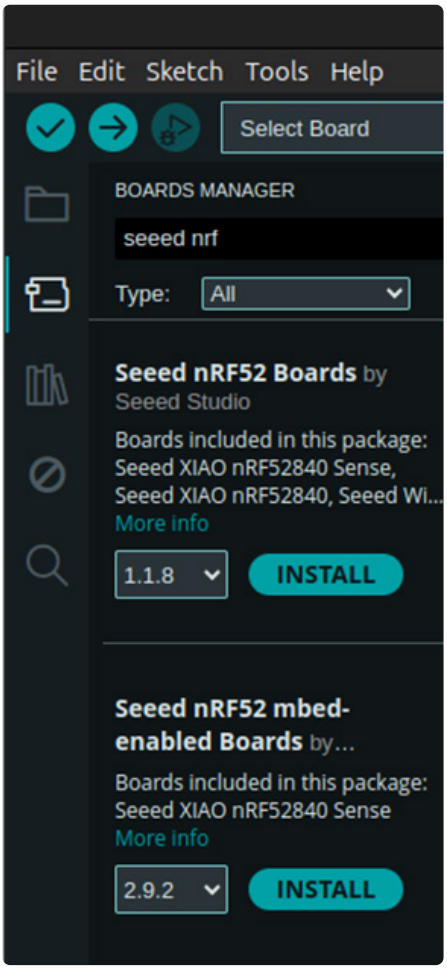

Figure 2. Installer le paquet Seeed nRF52 mbed-enabled Boards pour Arduino.

Ce paquet est optimisé pour les applications d'apprentissage automatique embarquées (**figure 2**). Après l'installation, connectez l'appareil au port USB de votre ordinateur, allez dans *Outils / Carte / Seeed nRF52 mbed-enabled Boards*, et choisissez *Seeed XIAO BLE Sense-nRF52840*. Ensuite, naviguez vers *Outils / Port* et sélectionnez le port série de votre appareil.

Pour vérifier que vous pouvez flasher un croquis Arduino sur l'appareil, ouvrez *Fichier / Exemples / 01.Basics / Blink*. Cela ouvre un croquis d'exemple qui fait simplement clignoter la LED intégrée. Cliquez sur l'icône *Téléverser* (la flèche orientée vers la droite) pour construire et flasher le croquis. Si tout se passe bien, la LED sur la carte commence à clignoter après environ 1 minute.

Essai de l'accéléromètre

Le nRF52840 Sense de Seeed Studio XIAO est équipé d'une unité de mesure inertielle (IMU) de haute précision à six axes LSM6DS3 qui inclut un accéléromètre à trois axes et un gyroscope à trois axes. Pour l'utiliser dans votre croquis Arduino, allez sur l'icône *Library Manager* dans la barre latérale gauche et recherchez « seeed lsm ». Installez la bibliothèque *Seeed Arduino LSM6DS3*.

La bibliothèque propose un code d'exemple, naviguez donc vers *Fichier / Exemples / Seeed Arduino LSM6DS3 / HighLevelExample*. Compilez et exécutez ce code, puis ouvrez le *Serial Monitor* dans le coin supérieur droit de l'EDI Arduino pour visualiser les coordonnées X, Y et Z de l'accéléromètre et du gyroscope, ainsi que la température du capteur interne (**figure 3**). Déplacez maintenant la carte Seeed dans votre main et observez les valeurs changeantes.

Modification du code de l'accéléromètre

Je souhaite utiliser les composantes X, Y et Z de l'accéléromètre pour développer un modèle d'apprentissage automatique, mais l'appareil un format de transmission des données plus compact. De plus, au lieu d'envoyer un échantillon toutes les secondes, comme c'est le cas avec l'instruction `delay(1000);`, je veux établir une fréquence d'échantillonnage de 50 Hz pour capturer avec précision les

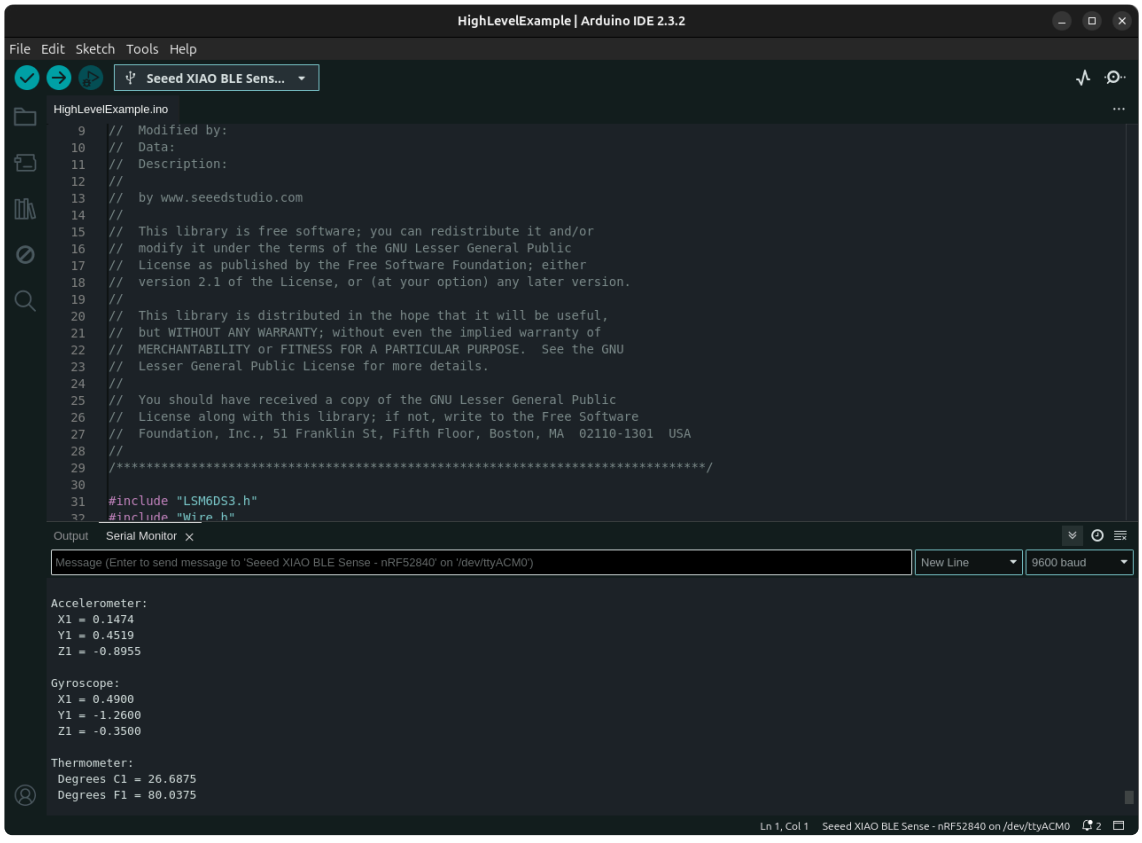

Figure 3. Les relevés de l'accéléromètre lors du déplacement de la carte Seeed.

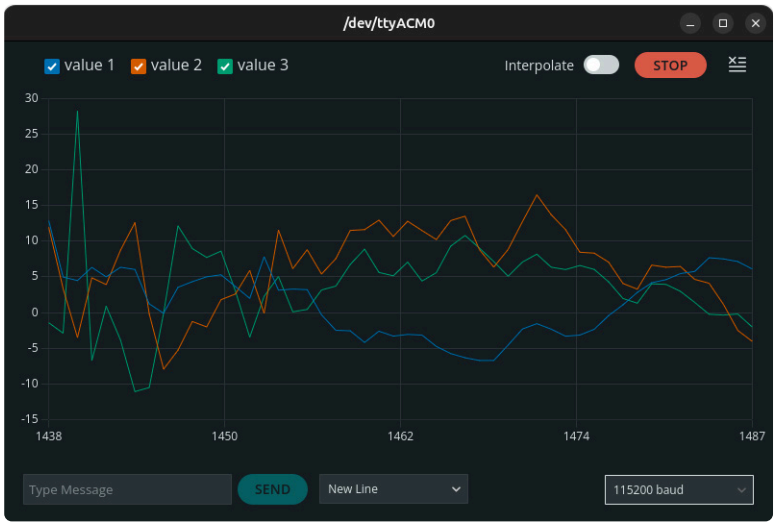

Figure 4. Le traceur série de l'EDI Arduino affiche les données de l'accéléromètre.

mouvements rapides de la main. Créez donc un nouveau croquis avec le code du **listage 1**, également disponible sur [4].
Exécutez ce code sur l'appareil et observez la sortie sur le moniteur série. Cela vous permettra de visualiser les variations de mouvement de façon beaucoup plus rapide, affichant les trois composantes d'accélération en m/s². Si vous cliquez sur l'icône *Serial Plotter* à côté du *Serial Monitor*, vous pouvez même visualiser les trois tracées des composantes de l'accéléromètre (**figure 4**).

Gestes

Pour démarrer avec l'acquisition de données d'accéléromètre en vue de créer un ensemble de données d'entraînement, il est d'abord

Listage 1. Lecture des données de l'accéléromètre LSM6DS3.

```
// XIAO BLE Sense LSM6DS3 Accelerometer Raw Data

#include "LSM6DS3.h"
#include "Wire.h"

//Create a instance of class LSM6DS3
LSM6DS3 myIMU(I2C_MODE, 0x6A);  //I2C device address 0x6A

#define CONVERT_G_TO_MS2 9.80665f
#define FREQUENCY_HZ 50
#define INTERVAL_MS (1000 / (FREQUENCY_HZ + 1))

static unsigned long last_interval_ms = 0;


void setup() {
  Serial.begin(115200);
  while (!Serial)
    ;

  if (myIMU.begin() != 0) {
    Serial.println("Device error");
  } else {
    Serial.println("Device OK!");
  }
}


void loop() {
  if (millis() > last_interval_ms + INTERVAL_MS) {
    last_interval_ms = millis();
    Serial.print(myIMU.readFloatAccelX() * CONVERT_G_TO_MS2, 4);
    Serial.print('\t');
    Serial.print(myIMU.readFloatAccelY() * CONVERT_G_TO_MS2, 4);
    Serial.print('\t');
    Serial.println(myIMU.readFloatAccelZ() * CONVERT_G_TO_MS2, 4);
  }
}
```

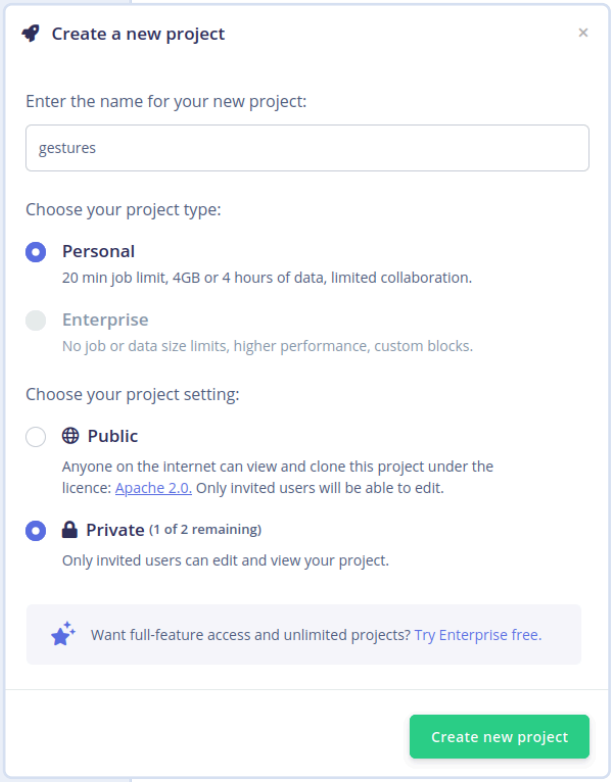

Figure 5. Créer un nouveau projet dans Edge Impulse..

nécessaire de s'inscrire sur Edge Impulse. Commencez par créer un compte, qui offre un compte gratuit dans le cadre de son plan communautaire pour les étudiants, les universités et les développeurs individuels [5]. Dans votre tableau de bord, cliquez sur *Créer un nouveau projet*, donnez un nom au projet, et choisissez de rendre le projet public ou privé (**figure 5**).
Ensuite, il est essentiel de définir quels gestes

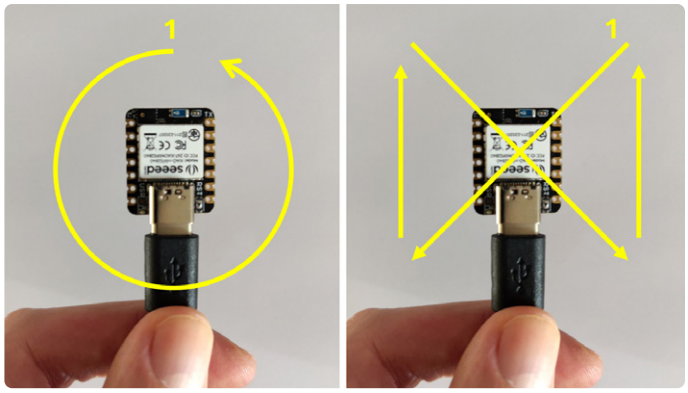

Figure 6. Les deux gestes que nous voulons détecter avec l'accéléromètre intégré de la carte.

```
/bin/edge-impulse-data-forwarder
Edge Impulse data forwarder v1.27.1
? What is your user name or e-mail address (edgeimpulse.com)? koen@vervloesem.eu
? What is your password? [hidden]
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/ttyACM0
[SER] Serial is connected (D1:4A:E5:69:16:66:06:9E)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

? To which project do you want to connect this device? Koen Vervloesem / gestures
[SER] Detecting data frequency...
[SER] Detected data frequency: 50Hz
? 3 sensor axes detected (example values: [2.3115,-6.3171,7.1498]). What do you want to call t
hem? Separate the names with ',': x,y,z
? What name do you want to give this device? xiao
[WS ] Device "xiao" is now connected to project "gestures". To connect to another project, run
 `edge-impulse-data-forwarder --clean`.
[WS ] Go to https://studio.edgeimpulse.com/studio/509218/acquisition/training to build your ma
chine learning model!
```

Figure 7. Le Edge Impulse Data Forwarder envoie les données de l'accéléromètre de votre microcontrôleur vers le cloud pour l'entraînement.

vous souhaitez que votre modèle reconnaisse. Prenez, par exemple, les gestes formant un cercle et une croix. Pour enregistrer geste circulaire, effectuez un mouvement circulaire avec votre dispositif. Pour une croix, déplacez la carte dans un plan imaginaire allant d'en haut à gauche à en bas à droite, puis d'en haut à droite à en bas à gauche, et retournez à la position de départ. (**Figure 6**).

Nous allons maintenant créer un jeu de données avec au moins 50 échantillons pour chaque mouvement, et 50 échantillons pour les mouvements inconnus (aléatoires). Assurez-vous de maintenir la carte de manière constante à travers tous les échantillons, et de réaliser les mouvements dans une direction cohérente, par exemple dans le sens horaire ou antihoraire pour le cercle, et en respectant une durée approximative similaire pour chaque échantillon (par exemple 2,5 s).

Transmission des données à Edge Impulse

Nous disposons déjà du script permettant d'enregistrer les valeurs de l'accéléromètre ; il nous suffit de transmettre ces données à Edge Impulse pendant nos mouvements d'entraînement, puis de diviser les données en durées de 2,5 s. Pour ce faire, nous pouvons utiliser le Data Forwarder d'Edge Impulse, qui fait partie du logiciel Edge Impulse CLI [6]. Commencez par installer les prérequis sur votre ordinateur, à savoir Python 3 et Node.js. Par la suite, installez les outils CLI nécessaires avec la commande :

```
npm install -g edge-impulse-cli
```

Lancez ensuite le Data Forwarder d'Edge Impulse avec :

```
edge-impulse-data-forwarder
```

Lorsque vous exécutez cette commande, vous serez invité à entrer votre nom d'utilisateur ou votre adresse électronique et votre mot de passe pour Edge Impulse. Le programme se connecte ensuite à votre appareil via l'interface série USB et vous demande de spécifier le projet auquel les données doivent être associées.

Le transfert de données détecte la fréquence des données, ainsi que les trois axes, et vous invite à nommer les axes. Nommez-les *x*, *y*, et *z*, séparés par des virgules. Ensuite, nommez l'appareil, après quoi le programme affiche l'URL de la section *Data acquisition* du projet (**figure 7**).

Enregistrement des données de l'entrainement

Dans la section *Collect data* à droite, vous devriez voir votre appareil connecté. Le nombre d'axes et la fréquence devraient déjà être corrects, mais nous devons modifier la durée d'échantillonnage par défaut qui est de 10 s. Nous utiliserons des échantillons de 2,5 s, mais nous répéterons le même geste plusieurs fois dans chaque enregistrement, puis nous diviserons l'enregistrement avec un intervalle de 2,5 secondes entre chaque répétition. Pour atteindre 50 répétitions, nous avons besoin d'un enregistrement de 250 s. Comme il est difficile de se concentrer aussi longtemps sur la réalisation de mouvements précis, nous enregistrerons 5 échantillons de 50 s, chacun composé de 10 répétitions. Définissez donc une durée d'échantillon de 50 000 ms, et n'oubliez pas d'attribuer une étiquette à chaque geste, par exemple *circle*. Maintenez la planche stable et cliquez sur *Start sampling*. Effectuez un geste circulaire ne dépassant pas 2,5 secondes, à répéter toutes les 5 secondes jusqu'à ce que l'échantillonnage s'arrête. Répétez cette procédure cinq fois. Ensuite, changer l'étiquette pour le deuxième nom de classe *cross*, enregistrer cinq échantillons de 50 s en faisant ce geste (croix). Enfin, enregistrez des échantillons avec l'étiquette *unknown* en faisant des mouvements aléatoires ou en posant la carte sur une table. Après cela, vous devriez avoir recueilli plus de 12 minutes de données (**figure 8**).

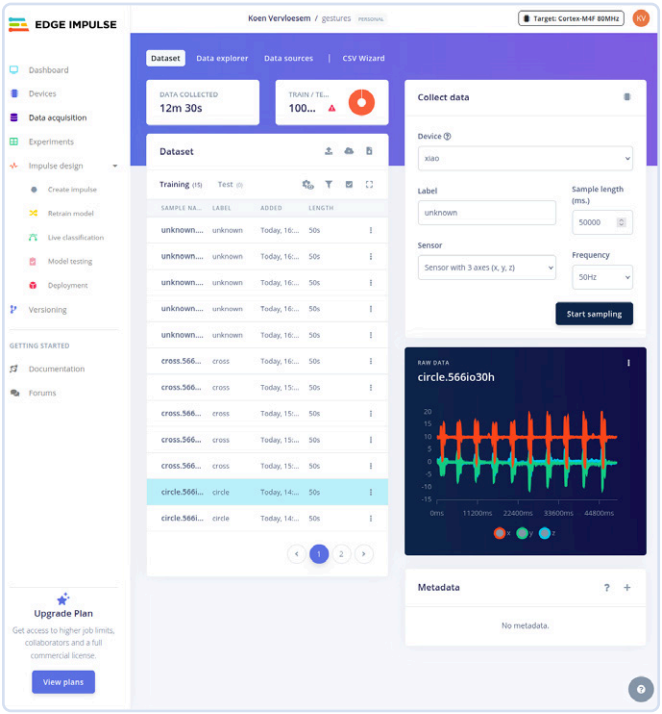

Figure 8. Nous venons d'enregistrer 12 minutes de relevés d'accéléromètre.

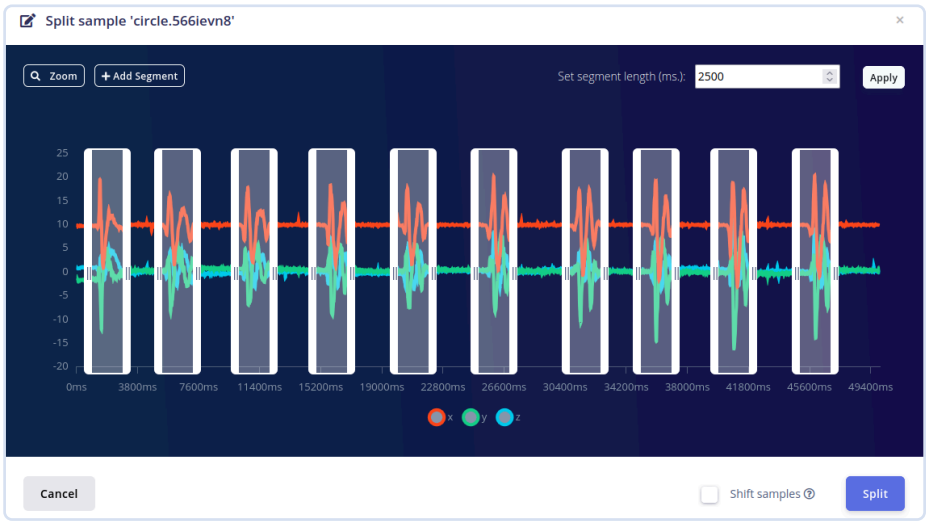

Figure 9. Diviser les enregistrements de l'accéléromètre en 10 échantillons de gestes

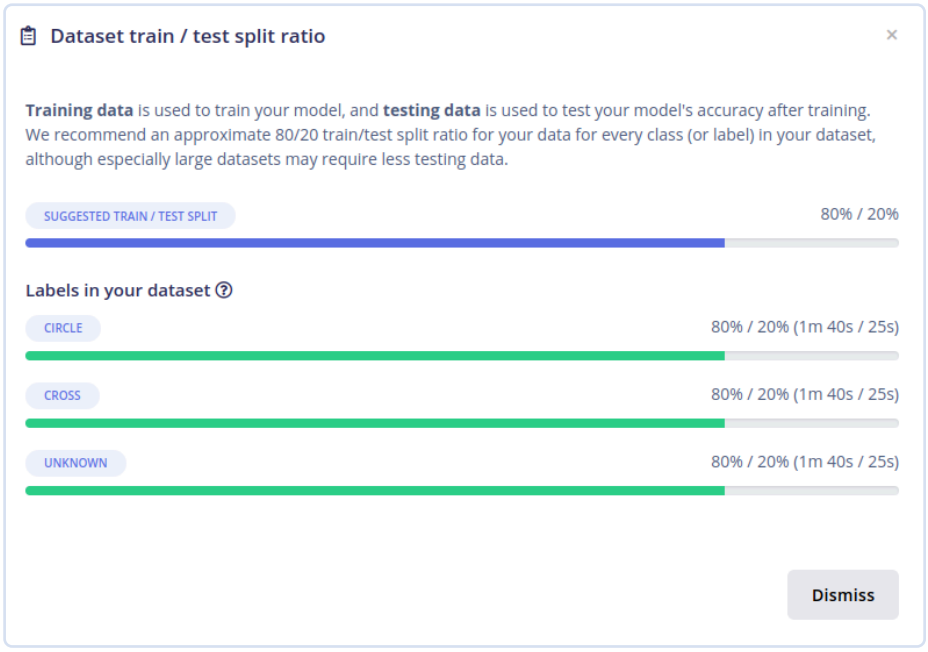

Figure 10. Il s'agit d'une répartition équilibrée entre entraînement et test.

Pour chaque échantillon enregistré, cliquez sur les trois points situés à côté, puis sélectionnez *Split sample*. Pour *Set segment length (ms.),* entrez 2500 et cliquez sur *Appliquer*. Si vos gestes ne dépassent pas 2,5 s, Edge Impulse les segmente automatiquement. Cliquez sur *Diviser* pour extraire les gestes individuels en tant qu'échantillons. Pour chaque échantillon original, vous obtiendrez 10 échantillons de 3 s maintenant (**figure 9**). Le nombre d'échantillons inconnus peut varier en fonction de vos mouvements aléatoires. Vous pouvez ajouter des échantillons supplémentaires en cliquant sur *Add Segment*, par exemple pour inclure des échantillons avec la carte posée à plat. Divisez tous les échantillons enregistrés en conséquence. Si la reconnaissance automatique des segments par Edge Impulse est erronée, vous pouvez toujours ajuster manuellement les segments. À la fin, vous devriez avoir 150 échantillons au total, ce qui donne un peu plus de 6 minutes de données.

Accédez à l'option *Train / Test split* signalée par un point d'exclamation, puis cliquez sur *Perform train / test split*. Chacune de vos étiquettes devrait maintenant être divisée en environ 80 % de données d'entraînement et 20 % de données de test (**figure 10**). Vous pouvez encore déplacer des échantillons des données d'entraînement vers les données de test ou l'inverse en cliquant sur les trois points d'un échantillon et en choisissant *Move to test set* ou *Move to training set* pour remédier à un déséquilibre.

Création du modèle

Cliquez sur *Create impulse* sous *Impulse design.* Vous devez d'abord configurer votre périphérique cible. Bien que le XIAO nRF52840 Sense ne soit pas listé, vous pouvez sélectionner le *Nordic nRF52840 DK (Cortex-M4F 64MHz)*, qui a le même processeur. Donnez un nom à votre appareil et cliquez sur Save.

Une impulse est une série de blocs traitant vos données depuis la série temporelle brute jusqu'aux caractéristiques de sortie. Dans le premier bloc, configurez la taille de la fenêtre à 2 500 ms et l'augmentation de la fenêtre à 400 ms. Cliquez ensuite sur *Add a processing block* et choisissez *Spectral analysis*. Ce bloc identifie les caractéristiques du signal dans le domaine des fréquences. Ensuite, cliquez sur *Add a learning block* et choisissez *Classification*. Ce bloc utilise les caractéristiques spectrales pour détecter si le signal correspond à un cercle, à une croix ou à un geste inconnu. Enfin, cliquez sur *Save impulse* (**figure 11**).

Cliquez ensuite sur *Spectral features*. Sous *Filter*, cliquez sur *Type* et choisissez *low* pour ajouter un filtre passe-bas. Réglez la fréquence de coupure à 10 Hz pour éliminer les signaux de basse fréquence et réglez l'ordre à 2 pour utiliser un filtre de Butterworth du second ordre. Vous pourrez modifier ces valeurs ultérieurement. Sous *Analyse*, réglez la longueur de la FFT sur 128 fréquences de sortie. Cliquez ensuite sur *Save parameters*. Vous accédez alors à l'onglet *Generate features*. Cliquez sur *Generate features*. Les résultats s'affichent dans le *Feature explorer*. Si les caractéristiques sont clairement différentes pour différents gestes, vous devriez voir les couleurs de chaque geste regroupées (**figure 12**).

Cliquez ensuite sur *Classifier*. L'*architecture du réseau neuronal* présente une couche d'entrée, deux couches denses et une couche de sortie avec les trois classes. Cliquez sur *Add an extra layer*, choisissez *Dropout*, et réglez le taux d'abandon à 0,2. Cela améliore la précision et vous pouvez ajuster le taux ultérieurement pour trouver la valeur optimale.

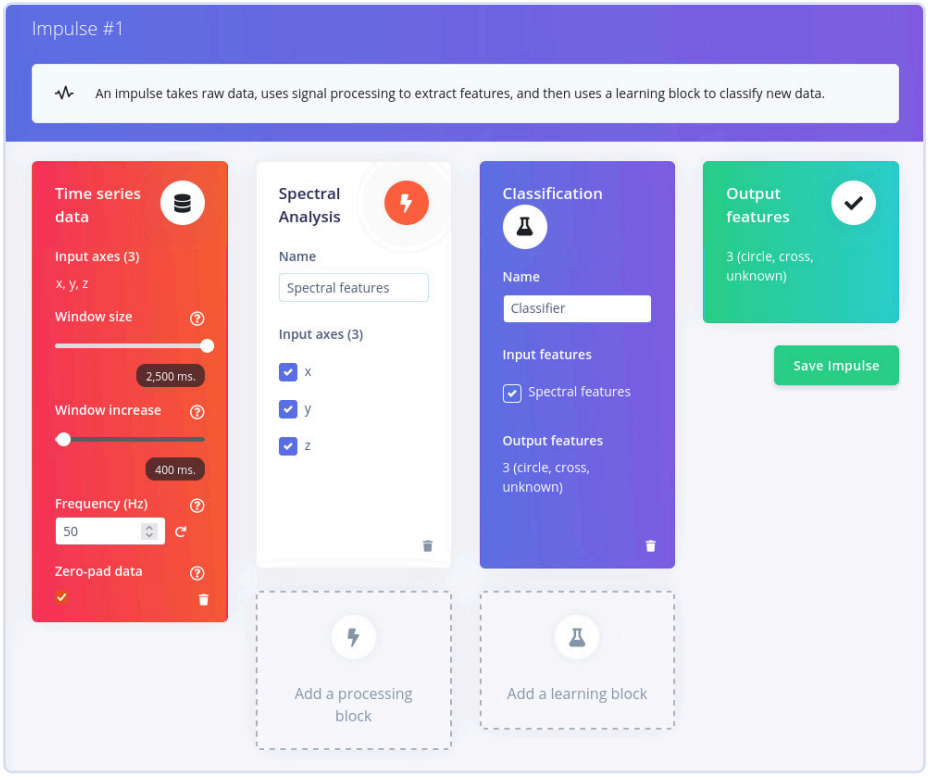

Figure 11. Créez un ***impulse*** *pour classer vos données.*

Figure 12. L'explorateur de caractéristiques montre une distinction prometteuse entre les trois classes.

Déplacez la couche entre les deux couches denses. Augmentez le nombre de cycles d'entraînement à 60, puis cliquez sur *Save & train*.

Dans notre cas, tous les cercles et toutes les croix ont été correctement identifiés, mais 14 % des mouvements inconnus ont été mal classés en tant que cercles (**figure 13**). Expérimentez avec différents taux d'abandon ou d'autres paramètres, entraînez à nouveau le modèle et observez l'impact sur la précision. Notez qu'Edge Impulse affiche par défaut la version quantifiée (optimisée) du modèle. Cliquez dessus pour passer à la version *non optimisée (float32)* et vérifiez si la précision est meilleure. Dans notre cas, la précision était moindre, avec 18 % des cercles incorrectement reconnus comme des croix.

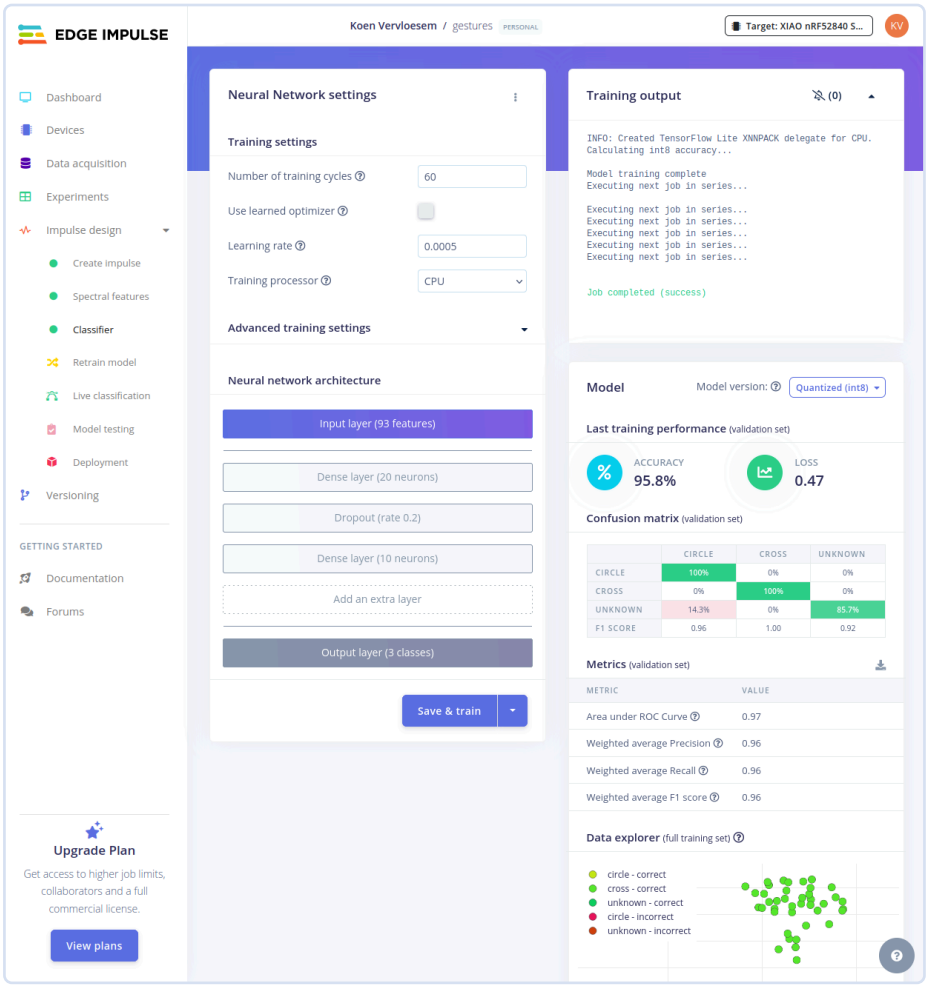

Figure 13. Le modèle entraîné atteint une précision de 95,8 %.

Test du modèle

Naviguez jusqu'à *Live classification*. Réglez la longueur de l'échantillon à 3 000, cliquez sur *Start sampling*, et faites un geste avec votre carte. Le modèle traitera les données en temps réel et affichera ses prédictions, indiquant l'horodatage ainsi que la probabilité associée à chaque geste détecté.

Edge Impulse a formé le modèle en se basant uniquement sur les données d'apprentissage, vous devez donc encore le tester sur l'ensemble de données de test. Allez dans *Model testing* et cliquez sur *Classify all*. Cela exécutera le modèle sur tous les échantillons des données de test et comparera la sortie aux étiquettes attribuées.

Si la précision de ce test de modèle est significativement inférieure à celle obtenue sur les données d'apprentissage, cela indique un surajustement potentiel. Vous pouvez essayer de modifier certains paramètres dans l'étape d'apprentissage du modèle ou rassembler de nouveaux échantillons pour y remédier.

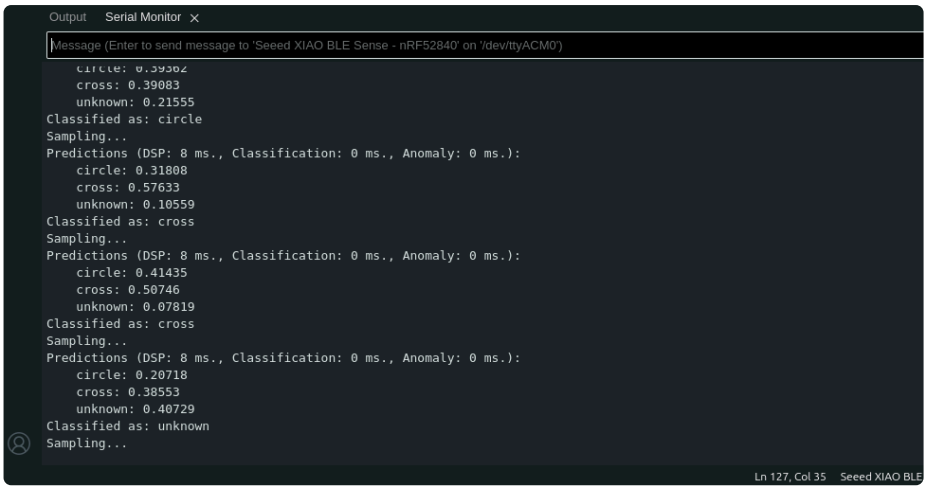

Figure 14. Détection en direct des gestes sur le capteur Seeed XIAO nRF52840.

Déploiement sur votre appareil

La dernière étape consiste à déployer votre modèle d'apprentissage automatique sur votre carte. Dans l'onglet *Deployment*, sélectionnez *Arduino library* en la recherchant dans le champ de texte. Choisissez la quantification, puis cliquez sur *Build* pour générer un fichier ZIP contenant la bibliothèque Arduino que vous pouvez installer sur votre ordinateur. Lancez l'EDI Arduino et allez dans *Sketch / Include Library / Add .ZIP Library*... et sélectionnez le fichier ZIP téléchargé. Dans mon exemple, avec un projet intitulé *gestures*, le fichier s'appelle *ei-gestures-arduino-1.0.1.zip*. Vous trouverez ensuite des exemples de code sous *File / Examples / gestures_inferencing*. Cependant, ces exemples sont conçus pour d'autres cartes que la XIAO nRF52840 Sense. Pour cette dernière, vous pouvez exécuter le croquis Arduino présenté dans le **listage 2**, basé sur les exemples d'Edge Impulse pour d'autres cartes.

Notez que le fichier d'en-tête `gestures_inferencing.h` au début fait référence à la bibliothèque que vous avez installée avec le modèle d'apprentissage automatique intégré. Dans la fonction loop, le code s'exécute en continu et gère l'acquisition des données depuis l'accéléromètre, stocke ces données dans une mémoire tampon, convertit la mémoire tampon brute en un signal traitable, puis exécute le classificateur sur le signal. Enfin, la fonction affiche les prédictions pour chaque classe, ainsi que la classe ayant la probabilité la plus élevée. Compilez et téléchargez ce sketch sur votre carte via l'IDE Arduino, puis réalisez des gestes en tenant la carte pour observer les prédictions de gestes en temps réel dans la sortie du moniteur série (**figure 14**).

Pour aller plus loin

Utiliser Edge Impulse pour développer un modèle d'apprentissage automatique qui détecte des gestes via des données d'accéléromètre est relativement simple. Ce modèle peut être déployé sous forme de bibliothèque Arduino compatible avec plusieurs cartes de développement. Vous pouvez également personnaliser le sketch de base sur un Seeed XIAO nRF52840 Sense pour réaliser différentes actions en fonction des gestes reconnus. Par exemple, vous pouvez configurer la carte pour qu'elle fonctionne comme un clavier USB, envoyant des commandes à votre ordinateur pour gérer des applications telles qu'un lecteur multimédia.

240449-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (koen@vervloesem.eu), ou contactez Elektor (redaction@elektor.fr).

Produits

- **Raspberry Pi AI Kit**
  www.elektor.fr/20879

LIENS

[1] Antonio Aloisio, « télécommande IR universelle basée sur l'IA », édition spéciale IA 2024 d'Elektor : https://www.elektormagazine.fr/240433-04
[2] EDI Arduino : https://www.arduino.cc/en/software
[3] Edge Impulse : https://edgeimpulse.com
[4] Téléchargements : https://www.elektormagazine.fr/240449-04
[5] Edge Impulse plans : https://edgeimpulse.com/pricing
[6] Edge Impulse CLI : https://docs.edgeimpulse.com/docs/tools/edge-impulse-cli

Listage 2. Inférence des gestes à partir des données de l'accéléromètre LSM6DS3.

```
#include <gestures_inferencing.h>
#include <LSM6DS3.h>
#include <Wire.h>

#define CONVERT_G_TO_MS2    9.80665f
```

```
#define MAX_ACCEPTED_RANGE  2.0f

static bool debug_nn = false; // Set this to true to see
                              // e.g. features generated from the raw signal
LSM6DS3 myIMU(I2C_MODE, 0x6A);

void setup()
 {
     Serial.begin(115200);
     Serial.println("Edge Impulse Inferencing Demo");

     if (!myIMU.begin()) {
         ei_printf("Failed to initialize IMU!\r\n");
     }
     else {
         ei_printf("IMU initialized\r\n");
     }

     if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
         ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME
                   should be equal to 3 (the 3 sensor axes)\n");
         return;
     }
 }

 /**
  * @brief Return the sign of the number
  *
  * @param number
  * @return int 1 if positive (or 0) -1 if negative
  */
 float ei_get_sign(float number) {
     return (number >= 0.0) ? 1.0 : -1.0;
 }

 /**
 * @brief      Get data and run inferencing
 *
 * @param[in]  debug  Get debug info if true
 */
 void loop()
 {
     ei_printf("Sampling...\n");

     // Allocate a buffer here for the values we'll read from the IMU
     float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };

     for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3) {
         // Determine the next tick (and then sleep later)
         uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);

         buffer[ix] = myIMU.readFloatAccelX();
         buffer[ix+1] = myIMU.readFloatAccelY();
         buffer[ix+2] = myIMU.readFloatAccelZ();
```

```
        for (int i = 0; i < 3; i++) {
            if (fabs(buffer[ix + i]) > MAX_ACCEPTED_RANGE) {
                buffer[ix + i] = ei_get_sign(buffer[ix + i]) * MAX_ACCEPTED_RANGE;
            }
        }

        buffer[ix + 0] *= CONVERT_G_TO_MS2;
        buffer[ix + 1] *= CONVERT_G_TO_MS2;
        buffer[ix + 2] *= CONVERT_G_TO_MS2;

        delayMicroseconds(next_tick - micros());
    }

    // Turn the raw buffer in a signal which we can the classify
    signal_t signal;
    int err = numpy::signal_from_buffer(buffer,
                            EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
    if (err != 0) {
        ei_printf("Failed to create signal from buffer (%d)\n", err);
        return;
    }

    // Run the classifier
    ei_impulse_result_t result = { 0 };

    err = run_classifier(&signal, &result, debug_nn);
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        return;
    }

    // print the predictions
    ei_printf("Predictions ");
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei_printf(": \n");
    int max_value_index = 0;
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", result.classification[ix].label,
                                    result.classification[ix].value);
        if(result.classification[ix].value >=
          result.classification[max_value_index].value) {
            max_value_index = ix;
        }
    }
#if EI_CLASSIFIER_HAS_ANOMALY == 1
    ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif

    ei_printf("Classified as: %s\n",
              result.classification[max_value_index].label);

}
```