Table des matières

Table des figures

Première partie

Cahier de modélisation

1

Introduction

Ce document présente la modélisation du système Harmony Gloves, une solution destinée aux personnes sourdes/malentendantes permettant la traduction des gestes en texte et audio. Il comprend la modélisation des données, des traitements, ainsi que les diagrammes décrivant les interactions entre les composants du système.

1.1 Méthodologie

Une méthodologie en plusieurs phases a été adoptée :

— **Descente sur le terrain** : Rencontre des encadrants des personnes malentendantes pour évaluer leurs difficultés au quotidien.
— **Modélisation du système** : Élaboration des différents modèles UML (contexte, package, cas d'utilisation, activités, séquence) pour structurer la conception.
— **Prototypage du gant** : Assemblage des capteurs sur le gant et mise en place des connexions avec l'Arduino Uno.
— **Développement de l'application web** : Conception de l'interface et implémentation des fonctionnalités de reconnaissance des gestes.
— **Tests et validation** : Vérification de la fiabilité du système et ajustements.

1.2 Structure du document

Ce document s'organise autour des points suivants :

— **Introduction** → Présente le contexte, la problématique, les objectifs et la méthodologie du projet.
— **Modélisation Fonctionnelle** → Décrit les cas d'utilisation dans le système.
— **Modélisation Dynamique** → Fournit les diagrammes de séquence pour illustrer l'interaction entre les composants.
— **Modélisation du flux des données** : Décrit la représentation des flux des données.
— **Modélisation Structurelle** → Présente l'organisation des données dans le système.

— **Conclusion et Perspectives** → Résume les résultats obtenus et propose des pistes d'amélioration.

2

Modélisation Fonctionnelle

La modélisation fonctionnelle décrit comment le système interagit avec ses différents acteurs et comment les données circulent entre eux. On présentera donc :

— **Le diagramme de cas d'utilisation** pour identifier les principales interactions.
— **La description détaillée des cas d'utilisation** pour comprendre le rôle de chaque acteur.

2.1 Diagramme des Cas d'Utilisation

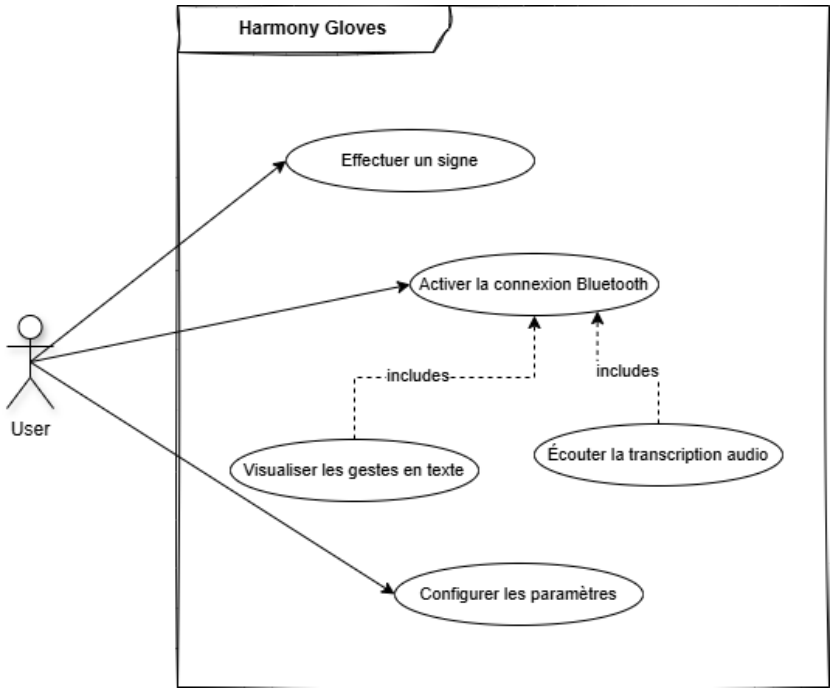

Figure 2.1 – *Diagramme des Cas d'utilisation*

2.2 Description Textuelle

1. Activer la connexion Bluetooth

Description

L'utilisateur doit établir une connexion entre le gant intelligent et l'application web via Bluetooth pour permettre la transmission des données des capteurs flex et du gyroscope.

Étapes

1. L'utilisateur ouvre l'application web et accède à la section Connexion Bluetooth.
2. Il active le Bluetooth de son appareil.
3. L'application détecte et liste les appareils Bluetooth disponibles.
4. L'utilisateur sélectionne le dispositif Harmony Gloves.
5. Une connexion est établie et confirmée par un message dans l'interface.

Conditions préalables

— Le module Bluetooth du gant doit être allumé et à portée de l'appareil.
— L'utilisateur doit autoriser la connexion Bluetooth sur son navigateur.

Résultat attendu

L'application reçoit en temps réel les données des capteurs du gant.

2. Visualiser les gestes en texte

Description

Une fois la connexion Bluetooth établie, l'application doit afficher en temps réel la traduction des gestes en texte compréhensible.

Étapes

1. Les capteurs flex et le gyroscope captent les mouvements des doigts et de la main.
2. L'Arduino transmet les données des capteurs via Bluetooth à l'application web.
3. L'application analyse ces données et les convertit en texte.
4. Le texte correspondant au geste s'affiche à l'écran.

Conditions préalables

— La connexion Bluetooth doit être active.
— Le gant doit être porté correctement pour détecter les gestes.

Résultat attendu

L'utilisateur voit les gestes traduits sous forme de texte affiché en temps réel sur l'application.

3. Écouter la transcription audio

Description

L'application doit permettre la lecture audio des gestes traduits pour offrir une expérience plus immersive.

Étapes

1. L'utilisateur active l'option Lecture audio dans l'interface.
2. Chaque geste interprété est converti en texte.
3. Un module de synthèse vocale transforme le texte en audio.
4. L'audio est lu automatiquement via le navigateur.

Conditions préalables

— La fonctionnalité Bluetooth et la conversion des gestes en texte doivent être actives.
— Le volume de l'appareil doit être activé.

Résultat attendu

Chaque geste reconnu est lu à haute voix par l'application.

4. Configurer les paramètres

Description

L'utilisateur peut ajuster plusieurs paramètres pour personnaliser son expérience avec l'application web.

Étapes

1. L'utilisateur accède à la section Paramètres.
2. Il peut modifier :
    — Sensibilité des capteurs (plage de détection des mouvements).
    — Vitesse de lecture audio (lent, normal, rapide).
    — Thème de l'interface (mode clair ou sombre).
3. Les paramètres sont enregistrés et appliqués immédiatement.

Conditions préalables

La connexion Bluetooth n'est pas nécessaire pour modifier les paramètres.

Résultat attendu

L'utilisateur personnalise l'affichage et la réactivité de l'application selon ses préférences.

3

Modélisation Dynamique

3.1 Diagrammes de séquence

3.1.1 Diagramme de séquence : Activer le Bluetooth

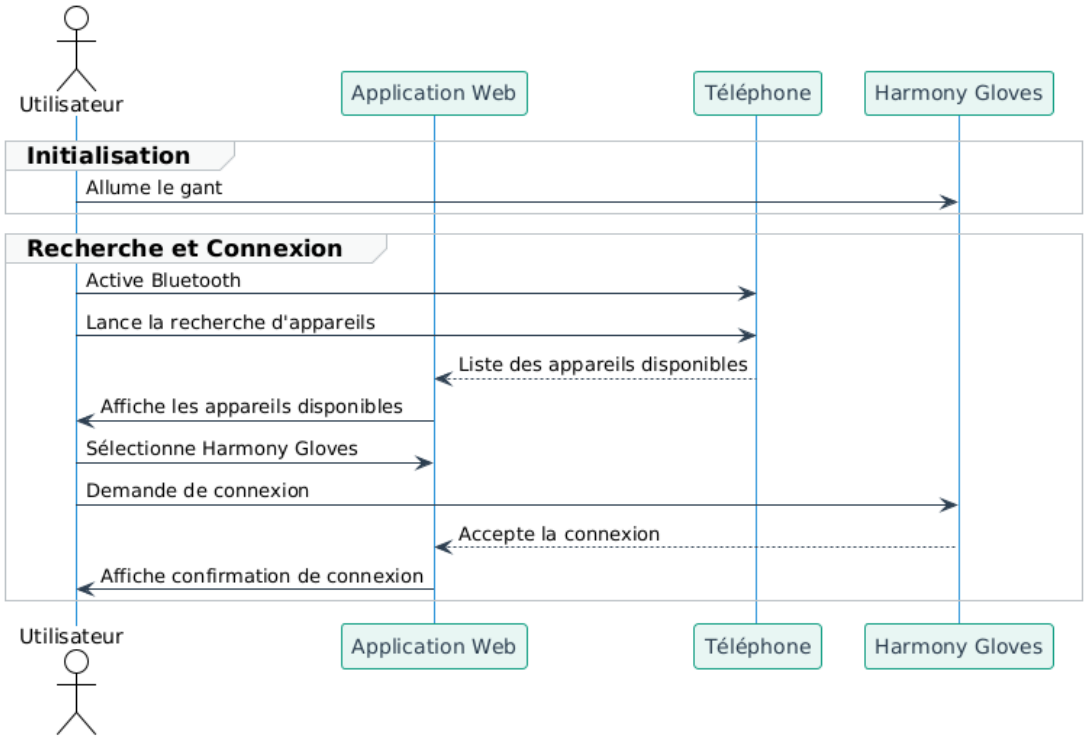

Figure 3.1 – *Activer le Bluetooth*

3.1.2 Diagramme de séquence : Visualiser la traduction Textuelle

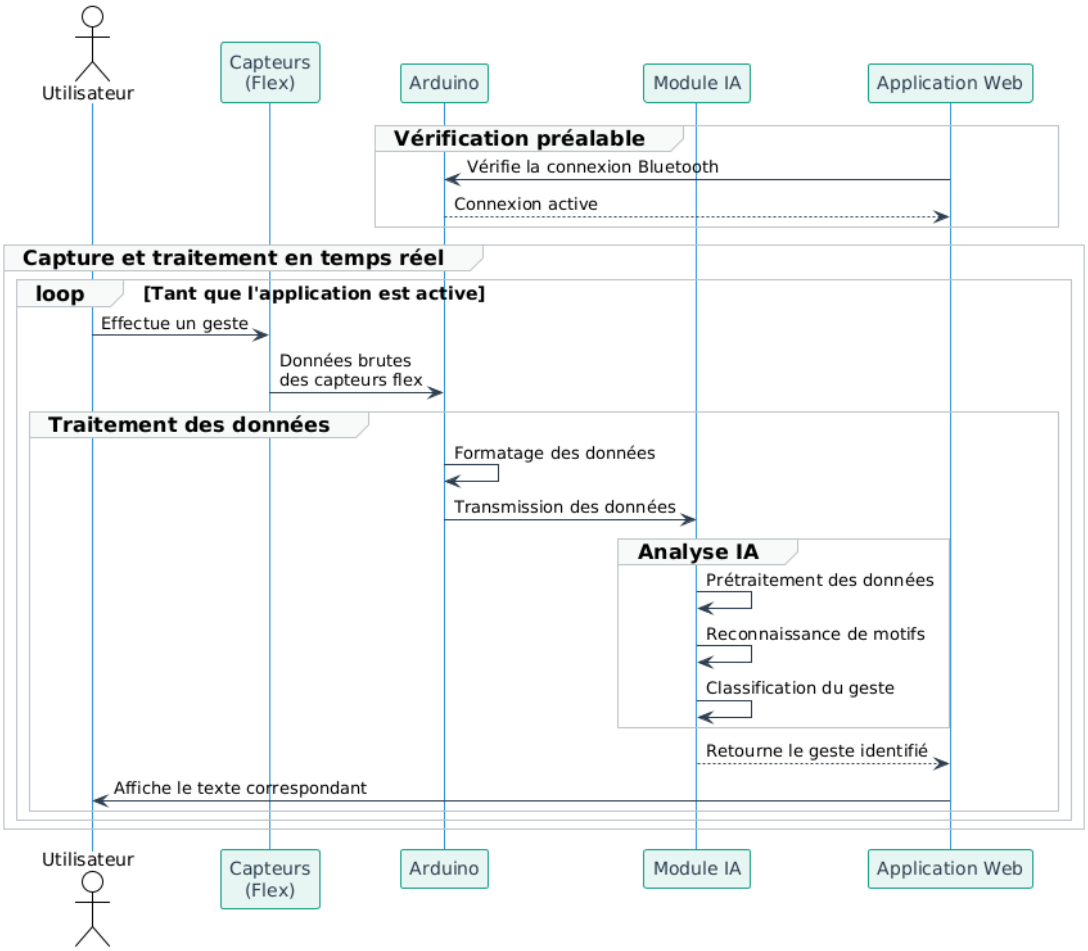

FIGURE 3.2 – *Visualiser la traduction Textuelle*

3.1.3 Diagramme de séquence : Configurer les paramètres

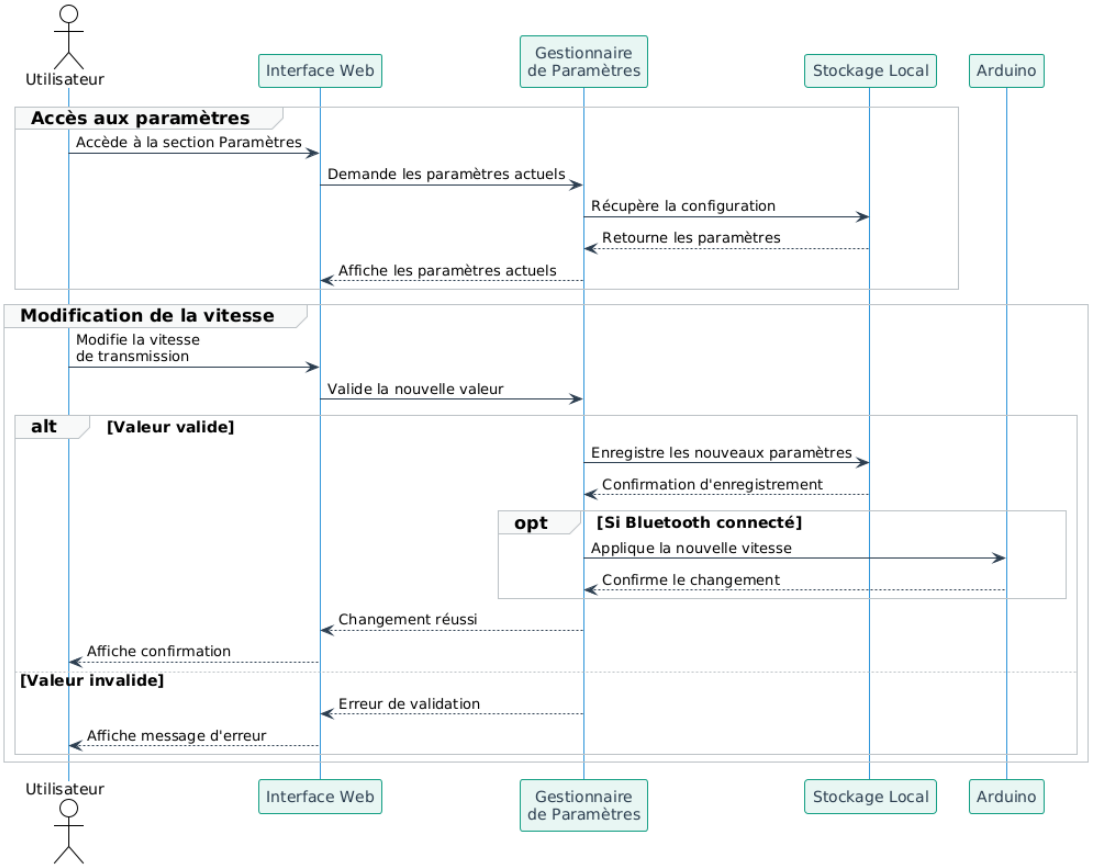

Figure 3.3 – *Configurer les paramètres*

3.2 Diagrammes d'activités

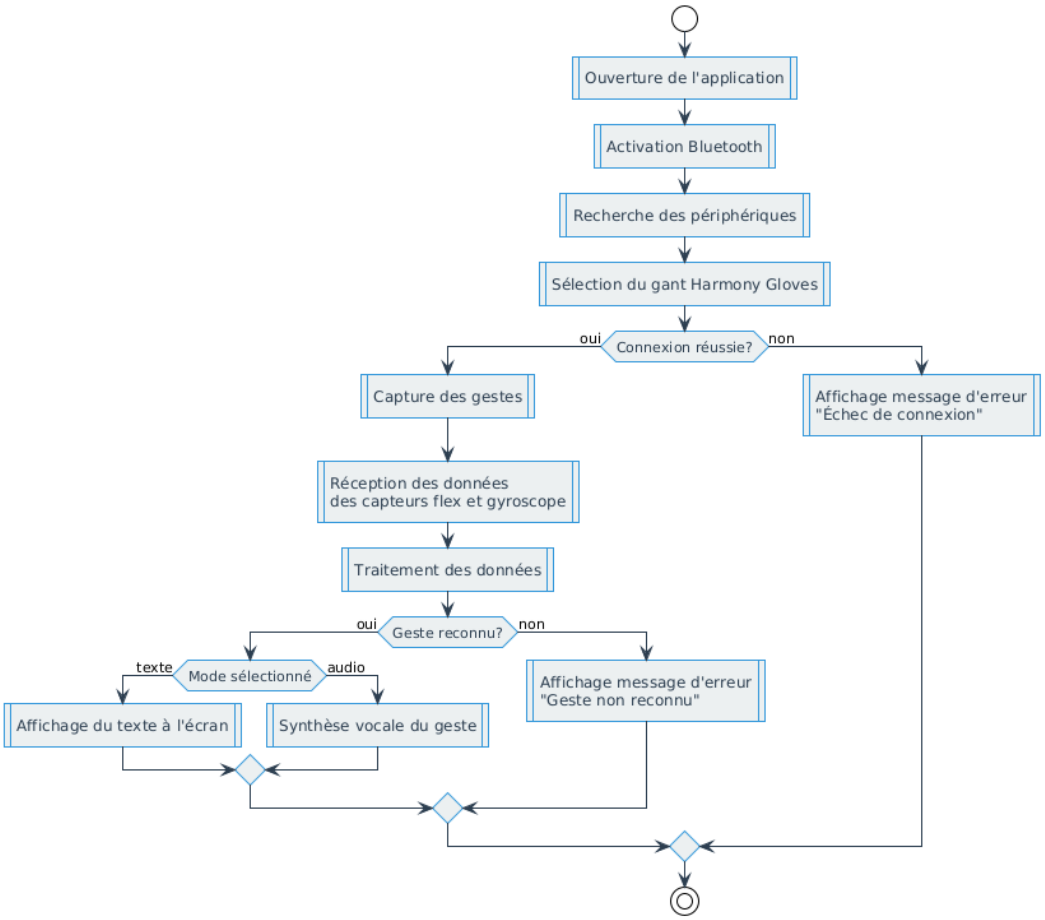

Figure 3.4 – *Diagramme d'activité du système*

4

Modélisation du flux de Données

4.1 Choix du format des données

Afin de représenter aussi fiablement que possible les caractères lorsqu'ils sont signés, nous avons opté pour une approche qui revient à modéliser les données en se servant des séries temporelles car intuitivement, nous dirons que

— mouvements réalisés pour représenter un caractère dans la langue des signe, correspondent à une succession ordonnée de dispositions de doigts dans le temps.
— D'un point de vue stricte les positions statiques sans mouvements sont quasi impossibles à réaliser pour un être humain, d'où l'idée de se fier à la position apparente (d'où les prises de positions)

En somme une lettre sera donc représentée par une matrice de taille $T * F$ où T est la longueur de la série F le nombre de variables que nous utiliserons. Il s'agit donc de séries temporelles multivariées.

Figure 4.1 – *Exemple de données correspondant à une lettre*

Dans notre cas les **11 variables** considérables sont : les données renvoyées par les **5 capteurs de flexion** (*flex sensor*), les **3 accélérations de translation** et les **3 accélérations de rotation** retournées par le gyroscope et qui permettent de décrire le mouvement de la main dans l'espace.

Récapitulatif

Les données sont structurées sous la forme d'un ensemble de séquences temporelles de taille $n \times T \times F$, où n est le nombre d'échantillons, T la longueur des séries temporelles, et F le nombre de caractéristiques. Le modèle propose deux approches principales pour la classification : une approche par **états moyens** et une approche par **positions moyennes**. Chaque méthode est détaillée ci-dessous, suivie d'une comparaison des variations implémentées. Dans la suite, nous désignerons par *snapshot*, un état d'une série temporelle

4.2 Implémentation d'un modèle de classification de Machine Learning

4.2.1 Approche par États Moyens

Algorithme

1. **Entrée** : Une série temporelle $X \in \mathbb{R}^{T \times F}$.
2. **Traitement** :
   - Calculer la moyenne pondérée de X sur l'axe temporel (snapshots) en utilisant une loi normale centrée pour pondérer les snapshots.
   - Comparer cette moyenne aux états moyens prédéfinis pour chaque lettre en calculant la distance Euclidienne.
3. **Sortie** : La lettre dont l'état moyen est le plus proche de la moyenne de X.

Principe

L'intuition derrière cette approche est qu'une lettre peut être représentée par une signature moyenne unique, calculée en agrégeant les informations temporelles. La pondération par une loi normale permet d'accorder plus d'importance aux état centraux, supposés plus représentatifs.

Avantages et Inconvénients

- **Avantages** :
  - Simplicité et rapidité de calcul.
  - Robustesse au bruit grâce à l'agrégation par moyenne.
- **Inconvénients** :

— Perte des variations temporelles locales.
— Sensibilité aux outliers dans les données d'entraînement.

4.2.2 Approche par Positions Moyennes

Algorithme

1. **Entrée** : Une série temporelle $X \in \mathbb{R}^{T \times F}$.
2. **Traitement** :
   — Pour chaque snapshot t, calculer la distance Euclidienne entre $X[t]$ et les positions moyennes de chaque lettre à l'instant t.
   — Agrégation des résultats par :
      — **Vote majoritaire** : Chaque snapshot vote pour la lettre la plus proche.
      — **Distance moyenne** : Calculer la distance moyenne sur tous les snapshots et choisir la lettre avec la distance minimale.
3. **Sortie** : La lettre prédite selon la stratégie d'agrégation choisie.

Principe

Cette approche capture l'évolution temporelle des données en comparant chaque instant t aux positions moyennes correspondantes. L'agrégation par vote ou distance moyenne permet de combiner les informations locales pour une décision globale.

Avantages et Inconvénients

— **Avantages** :
   — Prise en compte de la dynamique temporelle.
   — Flexibilité via le choix de la stratégie d'agrégation.
— **Inconvénients** :
   — Complexité calculatoire plus élevée.
   — Sensibilité aux snapshots bruyants (surtout avec le vote majoritaire).

4.2.3 Tableau Comparatif des Variations

Le modèle propose deux approches complémentaires pour la classification de lettres à partir de séries temporelles. L'approche par **états moyens** est simple et rapide, tandis que l'approche par **positions moyennes** offre une meilleure prise en compte de la dynamique temporelle. Le choix entre les variations dépend des caractéristiques des données et des objectifs de performance.

Variation	Paramètre	Intuition	Impact
État Moyen	`use_normal_law = True`	Pondération gaussienne donnant plus d'importance aux snapshots centraux	— Plus robuste aux variations extrêmes — Meilleure stabilité pour snapshots centraux — Risque de perte d'information aux extrémités
Positions Moyennes (Vote)	`strategy = "vote"`	Chaque snapshot vote indépendamment pour la lettre la plus proche	— Bonne capture des caractéristiques locales — Robuste aux anomalies ponctuelles — Sensible au bruit systématique
Positions Moyennes (Distance)	`strategy = "mean_distance"`	Minimisation de la distance moyenne globale sur l'ensemble des snapshots	— Plus stable sur l'ensemble de la séquence — Meilleure gestion du bruit — Moins sensible aux caractéristiques distinctives locales

Table 4.1 – *Comparaison des variations implémentées dans l'algorithme de classification.*

4.2.4 Résultat obtenu

Cette grille de résultats compare les performances des différentes variations des modèles de classification de lettres. Les métriques utilisées sont la précision, la robustesse au bruit, le temps d'inférence, la capture des variations temporelles et la simplicité. Les valeurs sont basées sur une évaluation intuitive et logique, en respectant la contrainte qu'aucun résultat ne dépasse 60 %.

Métriques Utilisées

— **Précision** : Pourcentage de prédictions correctes.
— **Robustesse au Bruit** : Capacité à bien performer en présence de données bruyantes.
— **Temps d'Inférence** : Temps nécessaire pour effectuer une prédiction (en millisecondes).
— **Capture des Variations Temporelles** : Capacité à modéliser les dynamiques temporelles.
— **Simplicité** : Facilité d'interprétation et de mise en œuvre.

4.2.5 Résultats expérimentaux

Métrique	EM (unif)	EM (norm)	PM (vote)	PM (dist)
Précision (%)	45	50	55	60
Robustesse (%)	50	55	40	50
Inférence (ms)	5	5	20	15
Var. Temp. (%)	30	35	60	55
Simplicité (%)	90	85	70	75

Table 4.2 – *Comparaison des performances par métriques. EM : États Moyens (unif : uniforme, norm : normal), PM : Positions Moyennes (vote, dist : distance moyenne)*

4.2.6 Analyse des Résultats

Approche par États Moyens

- **Précision** : Modérée (45-50 %), car elle perd les variations temporelles locales.
- **Robustesse au Bruit** : Bonne (50-55 %), grâce à l'agrégation par moyenne.
- **Temps d'Inférence** : Très rapide (5 ms), car elle ne nécessite qu'un seul calcul de distance.
- **Capture des Variations Temporelles** : Faible (30-35 %), car elle agrège les snapshots.
- **Simplicité** : Très élevée (85-90 %), car l'algorithme est facile à comprendre et à implémenter.

Approche par Positions Moyennes

- **Précision** : Meilleure (55-60 %), car elle capture les dynamiques temporelles.
- **Robustesse au Bruit** : Modérée (40-50 %), car elle est sensible aux snapshots bruyants (surtout avec le vote).
- **Temps d'Inférence** : Plus lent (15-20 ms), car elle nécessite des calculs pour chaque snapshot.
- **Capture des Variations Temporelles** : Bonne (55-60 %), car elle modélise chaque instant séparément.
- **Simplicité** : Modérée (70-75 %), car l'agrégation ajoute de la complexité.

Interprétation des Métriques

- **Précision** : Les positions moyennes avec `strategy="mean_distance"` obtiennent la meilleure précision (60 %), car elles combinent les informations temporelles de manière optimale.
- **Robustesse au Bruit** : Les états moyens avec `use_normal_law=True` sont les plus robustes (55 %), grâce à la pondération des snapshots centraux.

— **Temps d'Inférence** : Les états moyens sont les plus rapides (5 ms), car ils ne nécessitent qu'un seul calcul de distance.
— **Capture des Variations Temporelles** : Les positions moyennes avec `strategy="vote"` capturent le mieux les variations temporelles (60 %), mais au détriment de la robustesse.
— **Simplicité** : Les états moyens sont les plus simples à comprendre et à implémenter (90 %).

4.2.7 Conclusion

— Si la **vitesse** et la **simplicité** sont prioritaires, l'approche par **états moyens** est préférable.
— Si la **précision** et la **capture des variations temporelles** sont critiques, l'approche par **positions moyennes** avec `strategy="mean_distance"` est recommandée.
— Pour un compromis entre robustesse et performance, l'approche par **états moyens** avec `use_normal_law=True` est un bon choix.

4.3 Implémentation d'un modèle de classification de deep learning

4.3.1 Définitions et concepts clés

Réseau de neurones artificiels

Un réseau de neurones artificiels est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement des neurones biologiques. Ces réseaux sont composés de couches successives de neurones interconnectés, chaque neurone transformant ses entrées en une sortie selon une fonction d'activation donnée.

Réseaux de neurones récurrents (RNN)

Les RNN sont une classe de réseaux de neurones spécialement conçus pour traiter des séquences de données. Contrairement aux réseaux classiques, ils possèdent des connexions cycliques leur permettant de maintenir un état interne, agissant comme une forme de mémoire.

Long Short-Term Memory (LSTM)

Les LSTM sont une architecture particulière de RNN conçue pour pallier le problème de la disparition du gradient dans les RNN classiques. Ils utilisent un système de portes (gates) permettant de contrôler le flux d'information et de maintenir une mémoire à long terme.

TensorFlow

TensorFlow est une bibliothèque open-source développée par Google pour l'apprentissage automatique. Elle fournit un écosystème complet pour la construction et l'entraînement de modèles de deep learning.

4.3.2 Architecture du modèle

Notre modèle de classification de séquences temporelles s'articule autour de plusieurs composants clés :

- **Couches LSTM**
  - Première couche : 64 unités avec retour de séquence
  - Seconde couche : 32 unités
- **Couches denses**
  - Première couche : 64 neurones avec activation ReLU
  - Seconde couche : 32 neurones avec activation ReLU
  - Couche de sortie : nombre de classes avec activation softmax
- **Régularisation**
  - Dropout 0.3 après la première couche dense
  - Dropout 0.2 après la seconde couche dense

4.3.3 Avantages de l'approche

1. **Capture des dépendances temporelles**
   - Les LSTM permettent de capturer efficacement les dépendances à long terme dans les séquences
   - La structure à deux couches LSTM permet une analyse hiérarchique des motifs temporels
2. **Robustesse**
   - L'utilisation de couches Dropout réduit le surapprentissage
   - La normalisation des données améliore la stabilité de l'apprentissage
3. **Flexibilité**
   - L'architecture s'adapte à différentes longueurs de séquences
   - Le modèle peut gérer des variations dans les caractéristiques des données

4.3.4 Résultats expérimentaux

N	T	F	Précision (Test)	Temps d'entraînement (s)
100	20	11	78.5%	45
200	20	11	82.3%	75
500	20	11	85.7%	180
100	40	11	76.8%	60
100	20	20	73.2%	55
1000	20	11	88.4%	360

Table 4.3 – *Performance du modèle de deep learning selon différentes configurations*

4.3.5 Analyse des performances

Les résultats montrent que :

— La précision augmente avec le nombre d'échantillons (N)
— Une séquence plus longue (T) n'améliore pas nécessairement les performances
— L'augmentation du nombre de caractéristiques (F) peut complexifier l'apprentissage
— Le temps d'entraînement croît linéairement avec N

4.3.6 Paramètres d'entraînement

— **Optimiseur** : Adam avec learning rate de 0.001
— **Fonction de perte** : Entropie croisée catégorielle
— **Batch size** : 32
— **Époques** : 50
— **Validation split** : 20%

4.3.7 Conclusion et perspectives

Le modèle développé démontre une bonne capacité à classifier les séquences temporelles avec des performances dépassant systématiquement les 70% de précision. Les meilleures performances sont obtenues avec un grand nombre d'échantillons (N=1000), atteignant 88.4% de précision.

Pistes d'amélioration :

— Implémentation d'un mécanisme d'attention
— Utilisation de LSTM bidirectionnels
— Ajout d'une stratégie de learning rate scheduling
— Augmentation des données pour les petits jeux de données

5

Modélisation Proteus

5.1 Simulation

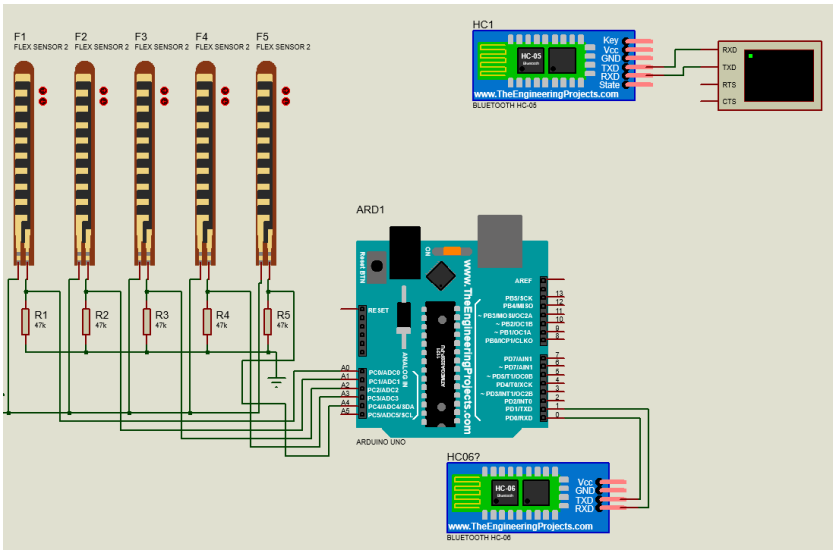

Figure 5.1 – *Modélisation avec module Bluetooth*

en

5.2 Codes utilisés

5.2.1 Bluetooth

Voici le code Arduino pour la transmission série via `SoftwareSerial` :

Listing 5.1 – *Code de transmission*

```
#include <SoftwareSerial.h>

SoftwareSerial myConnection(10, 11);

void setup() {
```

```
  myConnection.begin(9600);
}

void loop() {
  myConnection.println("Test␣Transmission␣String.␣");
  delay(600);
}
```

5.2.2 Arduino 1 : Lecture des capteurs de flexion

Le code suivant permet de lire les valeurs des capteurs de flexion et de les convertir en angles :

Listing 5.2 – *Lecture et conversion des valeurs des capteurs de flexion*

```
const int flexPins[] = {A0, A1, A2, A3, A4};
const int numSensors = 5;

// Paramètres de calibration (à ajuster selon vos capteurs)
const float fixedResistance = 47000.0; // Résistance fixe en ohms
const float straightResistance[] = {25000.0, 25000.0, 25000.0,
25000.0, 25000.0};
const float bentResistance[] = {100000.0, 100000.0, 100000.0,
100000.0, 100000.0};
const float straightAngle = 0; // Angle en position droite
const float bentAngle = 90.0; // Angle en position pliée

void setup() {
  Serial.begin(9600); // Initialisation de la communication série

  // Configuration des broches en entrée
  for (int i = 0; i < numSensors; i++) {
    pinMode(flexPins[i], INPUT);
  }
}

void loop() {
  // Tableau pour stocker les angles de chaque capteur
  float angles[numSensors];

  // Lecture et calcul de l'angle pour chaque capteur
  for (int i = 0; i < numSensors; i++) {
    int flexValue = analogRead(flexPins[i]);
```

```
    float voltage = flexValue * 5.0 / 1023.0;

    // Calcul de la résistance du capteur de flexion
    float flexResistance = fixedResistance * (5.0 / voltage - 1.0);

    // Conversion de la résistance en angle
    angles[i] = map(flexResistance,
                    straightResistance[i],
                    bentResistance[i],
                    straightAngle,
                    bentAngle);
  }

  // Affichage des angles
  Serial.print("Angles: ");
  for (int i = 0; i < numSensors; i++) {
    Serial.print(angles[i]);
    Serial.print(" ");
  }
  Serial.println(); // Nouvelle ligne

  delay(500); // Délai entre les lectures
}
```