RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ AHMED BEN BELLA (ORAN 1)

Institut des sciences appliquées et des techniques ISTA

Département d'optique et de mécanique de précision

Projet présenté à l'Institut des sciences appliquées et de technologie en vue de l'obtention du diplôme de master en instrumentation et métrologie

Présenté par **MEGUENINE Manar**

Thème

Étalonnage des capteurs pour les Smart Rooms : surveillance des enfants et des adultes grâce à l'IA

Jury:

Président	M. CHAIB Mohammed	University of Oran 1
Examinateur	M. BEKKAR Amine Azzedine	University of Oran 1
Encadrant	Mme. HAMDANI Nesrine	University of Oran 1

Juin 2026

Remerciements

Avant tout, je rends grâce à Allah, le Tout-Puissant, qui m'a accordé la santé, la patience, la force et la persévérance nécessaires pour mener ce travail à son terme. Malgré les épreuves et les difficultés rencontrées au cours de ce parcours, Sa bénédiction et Sa miséricorde m'ont permis de poursuivre mon chemin avec détermination.

J'adresse mes plus profonds remerciements à mes chers parents, pour leur amour inconditionnel, leurs sacrifices innombrables, leur soutien constant et leurs encouragements qui ont toujours été une source de motivation et de confiance.

Je tiens également à exprimer ma sincère gratitude à mes frères et ma sœur, Younes, Ilyas et Israa, ainsi qu'à Jamila, pour leur présence, leur soutien moral et leur affection tout au long de cette aventure académique.

Mes remerciements les plus chaleureux vont également à mes amis de cœur, Sid Ali, Souheila et Ismaïl, dont l'amitié sincère, l'écoute et les encouragements ont été d'un grand réconfort durant les moments les plus exigeants de ce parcours.

J'adresse un remerciement particulier à Kedjar Sara pour son aide précieuse, sa disponibilité et son soutien qui ont contribué à l'aboutissement de ce travail.

Je tiens à exprimer ma profonde reconnaissance à mon encadreuse, Mme Hamdani Nesrine, pour son accompagnement, ses conseils avisés, sa patience et son engagement tout au long de la réalisation de ce mémoire.

Je tiens à adresser mes sincères remerciements aux honorables membres du jury M. Chaib Mohamed et M. Bekkar Amine Azzedine pour avoir accepté d'évaluer ce travail. Je leur suis reconnaissante pour le temps consacré à son examen ainsi que pour leurs remarques et suggestions constructives. je tiens aussi à exprimer ma profonde gratitude à l'ensemble des professeurs de l'institut pour leur dévouement, leur disponibilité et la richesse des connaissances qu'ils m'ont transmises tout au long de mon parcours universitaire.

Enfin, je remercie toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire et à la réussite de mon parcours universitaire.

À toutes et à tous, j'adresse l'expression de ma profonde gratitude et de mon respect le plus sincère.

Résumé

Ce projet vise à intégrer divers capteurs et des technologies d'intelligence artificielle dans un système de chambre intelligente, en se concentrant sur la surveillance des enfants et des adultes.

L'objectif principal est d'assurer des mesures précises des capteurs grâce à des techniques d'étalonnage automatique, tout en exploitant l'IA pour surveiller et détecter des anomalies, telles que les chutes, les activités irrégulières ou les changements environnementaux.

De plus, le système permettra d'optimiser le confort et la sécurité en temps réel, améliorant ainsi l'environnement global de la pièce.
Mots clés: Intelligence Artificielle, chambre intelligente, étalonnage automatique, détection des anomalies, environnement global.

Table des matières

Remerciements **1**

Résumé **3**

Introduction générale **1**

1 État de l'art des Smart Rooms et des technologies de surveillance **7**

1.1 Introduction 7
1.2 Définition et évolution des Smart Rooms 8
1.3 Internet des Objets (IoT) et environnements intelligents 9
1.4 Types de capteurs utilisés dans les Smart Rooms 10
1.4.1 Capteurs environnementaux 11
1.4.2 Capteurs de mouvement et de présence 12
1.4.3 Capteurs biométriques et caméras 13
1.5 Applications de surveillance des enfants et des adultes 13
1.6 Enjeux éthiques et protection de la vie privée 16
1.7 Conclusion 17

2 Étalonnage et intelligence artificielle **19**

2.1 Introduction 19
2.2 Notion d'étalonnage des capteurs 20
2.3 Méthodes d'étalonnage 22
2.3.1 Méthodes statistiques 22
2.3.2 Méthodes expérimentales 23
2.3.3 Méthodes automatiques 24
2.4 Sources d'erreurs et incertitudes de mesure 25
2.5 Définition de l'intelligence artificielle 26
2.5.1 Définition du Machine Learning 27
2.5.2 Définition du Deep Learning 27
2.6 Algorithmes appliqués à l'étalonnage et à la surveillance 28
2.7 Conclusion 31

3 Conception et mise en œuvre du système proposé **32**
3.1 Introduction . . . 32
3.2 Architecture globale de la Smart Room . . . 33
3.2.1 Architecture fonctionnelle du système proposé . . . 34
3.3 Choix des capteurs et justification . . . 35
3.3.1 Arduino Nano . . . 35
3.3.2 DHT22 . . . 36
3.3.3 HC-SR501 . . . 37
3.3.4 Flame IR LM339 . . . 37
3.3.5 Prototype matériel . . . 38
3.4 Collecte et prétraitement des données . . . 40
3.5 Méthodologie d'étalonnage des capteurs . . . 41
3.6 Modèles d'intelligence artificielle utilisés . . . 43
3.6.1 RandomForestRegressor pour l'étalonnage . . . 43
3.6.2 RandomForestClassifier pour la classification . . . 43
3.6.3 Intégration API Flask et application Flutter . . . 43
3.6.4 Pseudo-algorithmes du système proposé . . . 45
3.7 Conclusion . . . 48

4 Expérimentation, résultats et discussion **49**
4.1 Introduction . . . 49
4.2 Description de l'environnement expérimental . . . 50
4.3 Scénarios de test enfants / adultes . . . 52
4.4 Évaluation des performances des capteurs étalonnés . . . 53
4.5 Évaluation des modèles d'intelligence artificielle . . . 54
4.6 Analyse et discussion des résultats . . . 56
4.6.1 Analyse de la calibration . . . 56
4.6.2 Analyse de la classification . . . 56
4.6.3 Comparaison avec les datasets Kaggle . . . 56
4.6.4 Intégration API Flask et Flutter . . . 57
4.6.5 Notifications et alerte critique . . . 59
4.7 Limites du système proposé . . . 62
4.8 Conclusion et perspectives . . . 62

Conclusion générale **64**

Références bibliographiques **69**

A Annexes techniques du système proposé **72**
Présentation générale des annexes . . . 72

A.1 Annexe A : Matériel utilisé et câblage du prototype 72
A.1.1 Liste du matériel 72
A.1.2 Récapitulatif des connexions 73
A.2 Annexe B : Structure du dataset expérimental 73
A.2.1 Variables principales 74
A.2.2 États considérés 74
A.2.3 Exemple de lignes du dataset 74
A.3 Annexe C : Pseudo-algorithmes du système proposé 75
A.3.1 Algorithme 1 : Acquisition et structuration des données 75
A.3.2 Algorithme 2 : Calibration hybride intelligente 75
A.3.3 Algorithme 3 : Classification de l'état de la chambre 76
A.3.4 Algorithme 4 : Génération de l'alerte et affichage Flutter 76
A.4 Annexe D : Calibration hybride intelligente 77
A.4.1 Correction par biais moyen pour la température 77
A.4.2 Principe de calibration par RandomForestRegressor 77
A.4.3 Résumé des résultats de calibration 78
A.5 Annexe E : Classification Random Forest 78
A.5.1 Résumé des performances de classification 79
A.6 Annexe F : API Flask 79
A.6.1 Structure simplifiée de l'API 79
A.6.2 Exemple de réponse JSON 80
A.7 Annexe G : Application Flutter et notifications 81
A.7.1 Structure recommandée du projet Flutter 81
A.7.2 Principe d'appel de l'API 81
A.7.3 Gestion des notifications 82
A.8 Annexe H : Comparaison avec les datasets Kaggle 82
A.9 Annexe I : Commandes principales d'exécution 82
A.10 Annexe J : Fichiers importants du projet 83
Conclusion des annexes 83

Liste des figures

1.1 Catégories de capteurs utilisés dans les Smart Rooms. 11

2.1 Principe général de l'étalonnage d'un capteur. 22
2.2 Différence entre calibration des capteurs et classification de l'état de la chambre. 29
2.3 Pipeline de calibration hybride et de classification de l'état de la chambre. 30

3.1 Architecture globale du système de surveillance proposé. 33
3.2 Architecture fonctionnelle du système proposé : entrées, traitement et sorties. 35
3.3 Carte Arduino Nano utilisée comme unité principale d'acquisition. 36
3.4 Capteur DHT22 utilisé pour mesurer la température et l'humidité de la chambre. 36
3.5 Capteur PIR HC-SR501 utilisé pour détecter la présence et le mouvement. 37
3.6 Capteur Flame IR LM339 utilisé pour détecter un risque de flamme. 37
3.7 Prototype matériel du système proposé basé sur Arduino Nano et capteurs. 38
3.8 Schéma de câblage du prototype Arduino Nano avec les capteurs utilisés. . 39
3.9 Pipeline de calibration hybride et de classification de l'état de la chambre. 42
3.10 Exemple de réponse JSON renvoyée par l'API Flask. 44
3.11 Dashboard Flutter affichant les valeurs capteurs, l'état prédit et les probabilités. 45

4.1 Distribution des classes du dataset expérimental. 51
4.2 Comparaison des erreurs avant et après calibration hybride. 53
4.3 Matrice de confusion du modèle Random Forest. 55
4.4 Comparaison des températures entre le dataset expérimental et les datasets Kaggle. 57
4.5 Exemple de réponse JSON renvoyée par l'API Flask. 58
4.6 Dashboard Flutter affichant les valeurs capteurs, l'état prédit et les probabilités. 59
4.7 Notification critique affichée dans l'application Flutter. 60
4.8 Écran de test manuel de prédiction dans l'application Flutter. 61

Liste des tableaux

1.1 Travaux existants sur les Smart Rooms et la surveillance intelligente 15

2.1 Sources d'erreurs possibles des capteurs utilisés 26
2.2 Algorithmes appliqués à l'étalonnage des capteurs et à la surveillance intelligente 30

3.1 Matériel utilisé dans le système proposé 39
3.2 Structure du dataset expérimental 40
3.3 États considérés pour la classification de la chambre 41
3.4 Méthodologie de calibration hybride proposée 42
3.5 Modèles et composants logiciels utilisés 48

4.1 Environnement expérimental 51
4.2 Scénarios de test enfants / adultes 52
4.3 Résultats de calibration hybride 53
4.4 Résultats du RandomForestClassifier 54
4.5 Comparaison avec datasets Kaggle 57
4.6 Fonctionnalités API Flask et Flutter 61

Introduction générale

L'évolution récente des systèmes embarqués, de l'Internet des objets et de l'intelligence artificielle a profondément transformé la manière dont les environnements domestiques peuvent être observés, analysés et assistés. Les chambres connectées, ou *Smart Rooms*, s'inscrivent dans cette dynamique en intégrant des capteurs, des unités de traitement et des interfaces logicielles capables de suivre l'état d'un espace intérieur. Dans le domaine de la santé, de l'assistance et de la sécurité domestique, ces technologies offrent des perspectives importantes, notamment pour les personnes vulnérables telles que les enfants, les personnes âgées ou les personnes nécessitant une surveillance régulière.

La surveillance des enfants et des personnes âgées dans une chambre constitue un cas d'usage sensible. Il ne s'agit pas seulement d'observer un environnement, mais de détecter des situations pouvant représenter un inconfort, un danger ou une anomalie. Une température trop élevée, une humidité importante, une absence prolongée de mouvement ou la détection d'un risque de flamme peuvent être des indicateurs utiles pour déclencher une alerte ou attirer l'attention d'un proche. Dans ce contexte, la chambre connectée devient un espace d'aide à la décision, capable de fournir des informations simples, lisibles et exploitables.

Les travaux existants sur les smart homes et la surveillance intelligente montrent que plusieurs technologies peuvent être utilisées pour suivre les activités humaines ou l'état d'un environnement intérieur. Certaines solutions reposent sur des caméras, des microphones, des dispositifs biométriques ou des capteurs portés par l'utilisateur. Ces approches peuvent être efficaces dans certains contextes, mais elles soulèvent également des questions importantes de confidentialité, d'acceptabilité et d'éthique, en particulier lorsqu'elles sont installées dans des espaces privés comme une chambre [2, 7]. Pour cette raison, les solutions non intrusives, basées sur des capteurs simples et ambiants, constituent une orientation pertinente pour les environnements domestiques sensibles.

Le présent mémoire s'inscrit dans cette logique. Le système proposé vise à concevoir une Smart Room non intrusive destinée à la surveillance de l'état de la chambre, sans utiliser de caméra, de biométrie ou d'identification de la personne. L'objectif n'est pas de

reconnaître un enfant, un adulte ou une personne âgée, mais d'observer des variables environnementales et événementielles utiles : température, humidité, présence ou mouvement, et risque de flamme. Cette orientation permet de réduire l'intrusion dans la vie privée tout en maintenant une capacité de surveillance adaptée à des cas d'usage domestiques.

D'un point de vue matériel, le système s'appuie sur une carte Arduino Nano utilisée comme unité principale d'acquisition et de contrôle. Trois capteurs sont retenus : le DHT22 pour mesurer la température et l'humidité, le HC-SR501 pour détecter la présence ou le mouvement, et le Flame IR LM339 pour détecter un risque de flamme. Ces composants sont peu coûteux, faciles à intégrer et compatibles avec une architecture embarquée légère. Cependant, comme beaucoup de capteurs low-cost, ils peuvent produire des mesures imparfaites, bruitées ou sensibles aux conditions d'utilisation. La question de l'étalonnage devient alors essentielle.

L'étalonnage des capteurs consiste à corriger les mesures brutes afin d'obtenir des valeurs plus fiables et plus cohérentes. Dans les systèmes classiques, cette correction peut reposer sur des méthodes statistiques simples, comme l'application d'un biais moyen ou d'une régression linéaire. Les travaux récents montrent toutefois que des modèles d'apprentissage automatique, en particulier les forêts aléatoires, peuvent améliorer la calibration de capteurs à faible coût dans des contextes environnementaux [1, 10, 15]. Cette approche est particulièrement intéressante pour un système de surveillance domestique, car elle permet de combiner simplicité matérielle et intelligence logicielle.

La motivation principale de ce mémoire est donc de proposer une solution complète, allant de l'acquisition des données jusqu'à l'affichage mobile des alertes, en passant par l'étalonnage intelligent et la classification de l'état de la chambre. La solution ne se limite pas à un montage électronique : elle intègre également un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés, des modèles d'apprentissage automatique, une API Flask et une application Flutter. Cette chaîne permet de démontrer la faisabilité d'une Smart Room non intrusive, intelligente et accessible.

La mise en place d'un système de surveillance dans une chambre connectée soulève plusieurs défis. Le premier concerne la nature des données collectées. Dans un espace privé, il est nécessaire de limiter au maximum les données sensibles afin de préserver la vie privée des utilisateurs. Les technologies telles que les caméras ou certains dispositifs biométriques peuvent fournir des informations riches, mais elles sont souvent perçues comme intrusives et peuvent être difficiles à accepter dans une chambre [7]. Un système adapté à la surveillance domestique doit donc privilégier des capteurs ambiants, simples et non identifiants.

Le deuxième défi concerne la fiabilité des mesures. Les capteurs à faible coût, bien qu'intéressants pour leur accessibilité, ne fournissent pas toujours des valeurs parfaitement stables ou exactes. Les mesures de température, d'humidité ou de flamme peuvent varier selon l'environnement, la position du capteur, les conditions d'alimentation ou les caractéristiques propres au composant. Une décision prise directement à partir de données brutes peut donc être moins fiable, notamment dans les situations critiques où une alerte doit être déclenchée avec prudence.

Le troisième défi concerne l'interprétation des données. Une chambre connectée produit plusieurs variables hétérogènes : mesures continues, états binaires, valeurs analogiques et informations de présence. Ces données doivent être combinées pour classer l'état global de la chambre. Il ne suffit donc pas de lire une température ou une valeur de flamme de manière isolée ; il faut construire une logique capable de reconnaître différents scénarios de surveillance, comme une présence normale, une absence, une chambre chaude, une humidité élevée, un risque de flamme ou une situation critique. Comment concevoir une Smart Room non intrusive, fondée sur des capteurs low-cost, capable d'étalonner intelligemment ses mesures et de classifier l'état d'une chambre afin de surveiller des situations liées aux enfants et aux personnes âgées, tout en préservant la simplicité matérielle, la confidentialité et l'exploitabilité des alertes ?

Cette problématique conduit à une réflexion globale sur l'équilibre entre coût, fiabilité, intelligence logicielle et respect de la vie privée. Le système recherché doit être suffisamment simple pour être déployé avec des composants accessibles, mais suffisamment intelligent pour corriger les mesures et interpréter l'état de la chambre. Il doit également être compréhensible pour l'utilisateur final, qui reçoit des informations sous forme de valeurs capteurs, d'état prédit, de probabilités et de notifications.

L'objectif général de ce mémoire est de concevoir et d'évaluer une solution de Smart Room non intrusive intégrant un étalonnage intelligent des capteurs et une classification automatique de l'état de la chambre. Le système vise à soutenir la surveillance d'environnements destinés aux enfants et aux personnes âgées, sans recourir à des technologies intrusives.

Les objectifs spécifiques sont les suivants :

- concevoir une architecture matérielle basée sur Arduino Nano, DHT22, HC-SR501 et Flame IR LM339 ;
- structurer un dataset expérimental selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés ;
- appliquer une calibration hybride intelligente afin d'améliorer la qualité des mesures issues des capteurs ;
- utiliser une correction par biais moyen pour l'étalonnage de la température ;

— utiliser un modèle RandomForestRegressor pour l'étalonnage de l'humidité et du signal analogique du capteur de flamme ;
— entraîner un modèle RandomForestClassifier pour classifier l'état de la chambre à partir des données calibrées ;
— comparer certaines caractéristiques du dataset expérimental avec des datasets Kaggle afin de situer les valeurs obtenues par rapport à d'autres contextes ;
— développer une API Flask permettant de fournir les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités et le niveau d'alerte ;
— développer une application Flutter capable d'afficher les informations de surveillance et de déclencher des notifications visuelles et locales.

Ces objectifs traduisent une volonté de produire une solution complète, allant du capteur à l'interface utilisateur. Le mémoire ne se limite donc pas à l'étude théorique de l'étalonnage ou de la classification, mais propose une chaîne fonctionnelle intégrant matériel, données, intelligence artificielle, API et application mobile.

Les contributions principales de ce travail sont la conception d'une Smart Room non intrusive, la mise en place d'une calibration hybride combinant correction statistique et apprentissage automatique, la classification de l'état de la chambre par Random Forest, ainsi que l'intégration des résultats dans une API Flask et une application Flutter fonctionnelle avec notifications.

La méthodologie adoptée dans ce mémoire repose sur une démarche progressive, structurée autour de quatre axes complémentaires : la conception matérielle, la préparation des données, l'étalonnage intelligent et l'intégration logicielle. Cette démarche permet de relier les aspects électroniques, algorithmiques et applicatifs du système proposé.

La première étape consiste à définir l'architecture matérielle de la chambre connectée. La carte Arduino Nano est choisie pour son format compact, sa simplicité d'utilisation et sa compatibilité avec les capteurs retenus. Le DHT22 fournit les mesures de température et d'humidité, le HC-SR501 donne une information de présence ou de mouvement, et le Flame IR LM339 fournit une information liée au risque de flamme. L'ensemble de ces capteurs permet de surveiller l'état de la chambre sans collecter d'image, de voix ou de donnée biométrique.

La deuxième étape concerne la préparation des données. Le travail s'appuie sur un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Cette formulation est importante, car elle garantit une cohérence entre les données utilisées pour l'entraînement des modèles et la structure attendue lors de l'exploitation du prototype. Les variables brutes sont organisées de manière à représenter

les entrées du système : température, humidité, mouvement, état numérique du capteur de flamme et valeur analogique associée.

La troisième étape consiste à appliquer une calibration hybride intelligente. La température est corrigée par une méthode de biais moyen, adaptée lorsque l'écart observé peut être représenté par une correction globale simple. L'humidité et la valeur analogique du capteur Flame IR LM339 sont calibrées à l'aide d'un RandomForestRegressor. Ce choix est justifié par la capacité des forêts aléatoires à modéliser des relations non linéaires entre les variables et par leur utilisation dans des travaux de calibration de capteurs low-cost [1, 15]. Les résultats obtenus indiquent une amélioration de la calibration de +1.24 % pour la température, +12.39 % pour l'humidité et +48.35 % pour la variable *flame_analog*.

La quatrième étape porte sur la classification de l'état de la chambre. Après calibration, les données sont utilisées pour entraîner un RandomForestClassifier. Ce modèle permet de classer l'état de la chambre selon les scénarios étudiés, comme la présence, l'absence, une chambre chaude, une humidité élevée, un risque de flamme ou une situation critique. L'utilisation d'un modèle Random Forest présente plusieurs avantages dans ce contexte : robustesse face aux variables hétérogènes, interprétation relativement accessible, bon comportement sur des jeux de données structurés et compatibilité avec une intégration légère dans une chaîne applicative. Le modèle a atteint une accuracy de 0.99 sur les données d'évaluation considérées.

La cinquième étape consiste à situer les valeurs du dataset expérimental par rapport à des datasets Kaggle. Cette comparaison n'a pas pour objectif de remplacer les données du système proposé, mais de fournir un point de repère externe pour certaines variables environnementales, notamment la température. Elle contribue à renforcer l'analyse expérimentale en comparant les ordres de grandeur observés avec ceux d'autres jeux de données.

Enfin, la dernière étape concerne l'intégration logicielle. Une API Flask est développée afin de transmettre les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités associées aux classes et le message d'alerte. Cette API a été validée à travers une réponse JSON structurée. Une application Flutter est ensuite utilisée pour afficher le tableau de bord, les valeurs des capteurs, l'état prédit et les probabilités. Les notifications sont intégrées à l'aide de SnackBar, AlertDialog et *flutter_local_notifications*, ce qui permet de signaler les situations critiques de manière visible et immédiate.

Cette méthodologie présente l'intérêt de construire une chaîne complète et cohérente. Les données brutes ne sont pas seulement collectées ; elles sont corrigées, interprétées, exposées par une API puis affichées dans une interface mobile. Le système proposé illustre ainsi l'apport de l'intelligence artificielle dans une architecture IoT simple, tout en respectant

le principe de non-intrusion.

Le mémoire est organisé en plusieurs chapitres complémentaires afin de présenter progressivement le contexte scientifique, les choix techniques, la méthodologie proposée et les résultats obtenus.

Le premier chapitre présente l'état de l'art des Smart Rooms, des smart homes et des technologies de surveillance intelligente. Il aborde les systèmes de surveillance domestique, les capteurs non intrusifs, les applications liées aux enfants et aux personnes âgées, ainsi que les enjeux de confidentialité et d'acceptabilité. Ce chapitre permet de positionner le travail par rapport aux solutions existantes et de justifier le choix d'une approche sans caméra ni biométrie.

Le deuxième chapitre est consacré à l'étalonnage des capteurs et à l'intelligence artificielle. Il présente les limites des capteurs low-cost, les méthodes statistiques de correction, les algorithmes d'apprentissage automatique utilisés pour la calibration et les modèles de classification adaptés aux données issues de capteurs. Les références sur Random Forest, la calibration environnementale et la reconnaissance d'activité dans les smart homes y sont mobilisées afin de justifier les choix algorithmiques du mémoire.

Le troisième chapitre décrit la conception du système proposé. Il détaille l'architecture matérielle, le rôle de chaque capteur, le schéma de câblage, le format des données et le pipeline global allant de l'acquisition à la classification. Ce chapitre met également en évidence le caractère non intrusif du système et la cohérence entre les capteurs choisis et les scénarios de surveillance étudiés.

Le quatrième chapitre présente l'expérimentation, les résultats obtenus et la discussion. Il expose les résultats de calibration, l'évaluation du modèle de classification, la comparaison avec les datasets Kaggle, ainsi que l'intégration logicielle à travers l'API Flask et l'application Flutter. Enfin, la conclusion générale synthétise les apports du travail, les limites rencontrées et les perspectives d'amélioration.

Chapitre 1

État de l'art des Smart Rooms et des technologies de surveillance

1.1 Introduction

La surveillance intelligente des environnements domestiques constitue aujourd'hui un domaine de recherche important, à la croisée de l'Internet des Objets, des systèmes embarqués, de l'intelligence artificielle et des interfaces mobiles. Les espaces intérieurs, longtemps considérés comme de simples lieux d'habitation, deviennent progressivement des environnements capables de mesurer leur propre état, d'interpréter certains événements et d'assister l'utilisateur dans la prise de décision. Cette évolution concerne particulièrement les chambres connectées, ou *Smart Rooms*, qui peuvent contribuer au confort, à la sécurité et au suivi de personnes vulnérables.

Dans le cadre de ce mémoire, l'intérêt porte sur une Smart Room non intrusive destinée à la surveillance des enfants et des personnes âgées. Le système proposé ne cherche pas à identifier les personnes ni à analyser leur apparence, mais à surveiller l'état de la chambre à travers des variables simples et utiles : température, humidité, présence ou mouvement, risque de flamme et génération d'alertes. Cette orientation est importante, car une chambre est un espace privé dans lequel les solutions trop intrusives, telles que les caméras ou certains dispositifs biométriques, peuvent poser des difficultés d'acceptation et de confidentialité [2, 7].

L'objectif de ce chapitre est de présenter l'état de l'art relatif aux Smart Rooms et aux technologies de surveillance. Il commence par définir les Smart Rooms et leur évolution depuis la domotique classique vers les environnements intelligents. Il introduit ensuite le rôle de l'Internet des Objets dans l'acquisition, la transmission et le traitement des données. Le chapitre présente également les principales familles de capteurs utilisées dans

les chambres connectées, en distinguant les capteurs environnementaux, les capteurs de mouvement et de présence, ainsi que les capteurs biométriques et les caméras comme technologies existantes. Enfin, il analyse les applications de surveillance destinées aux enfants et aux adultes, puis discute les enjeux éthiques et la protection de la vie privée.

Ce chapitre permet ainsi de situer le système proposé par rapport aux travaux existants. Il montre que les capteurs IoT constituent une base pertinente pour la surveillance intelligente, mais que le choix des capteurs doit être cohérent avec le contexte d'utilisation. Pour une chambre destinée à des enfants ou à des personnes âgées, une solution non intrusive fondée sur des capteurs simples, sans caméra et sans biométrie, représente un compromis adapté entre sécurité, acceptabilité et respect de la vie privée.

1.2 Définition et évolution des Smart Rooms

Une Smart Room peut être définie comme un espace intérieur équipé de capteurs, d'actionneurs, d'unités de calcul et de moyens de communication permettant de collecter des données, d'analyser l'état de l'environnement et de fournir des services adaptés au contexte. Contrairement à une chambre ordinaire, une Smart Room ne se limite pas à accueillir un utilisateur ; elle devient capable de produire des informations sur son propre fonctionnement. Elle peut par exemple mesurer la température, l'humidité, la luminosité, la présence, la qualité de l'air ou certains événements inhabituels.

L'évolution des Smart Rooms s'inscrit dans la continuité de la domotique classique. Les premiers systèmes domotiques visaient principalement l'automatisation de tâches simples : commande d'éclairage, contrôle de chauffage, ouverture ou fermeture d'équipements, ou programmation d'appareils électriques. Ces systèmes reposaient souvent sur des règles fixes et sur une intervention humaine importante. Avec le développement de l'Internet des Objets, des microcontrôleurs et des plateformes logicielles, les environnements domestiques sont devenus plus adaptatifs. Les objets connectés peuvent désormais communiquer entre eux, transmettre des données à une API, être consultés à distance et alimenter des modèles d'analyse.

La différence entre une Smart Home et une Smart Room réside surtout dans l'échelle du système. Une Smart Home désigne généralement l'ensemble d'un logement connecté, comprenant plusieurs pièces, différents sous-systèmes et une infrastructure plus large. Une Smart Room, en revanche, se concentre sur une pièce particulière. Cette restriction d'échelle peut être un avantage lorsqu'il s'agit de surveiller un espace sensible, comme une chambre d'enfant ou une chambre destinée à une personne âgée. Elle permet de limiter les données collectées, de simplifier le déploiement matériel et de concentrer l'analyse sur un ensemble restreint de scénarios.

Dans la littérature, les environnements intelligents sont souvent associés à des plateformes capables de collecter des données sur les activités quotidiennes, d'analyser des séquences d'événements et de fournir une assistance adaptée. La plateforme CASAS, par exemple, illustre l'intérêt des environnements intelligents pour l'observation des activités domestiques à l'aide de capteurs ambiants [8]. Les travaux sur la reconnaissance d'activité humaine dans les smart homes montrent également que les capteurs IoT peuvent être exploités pour comprendre certains comportements ou états de l'environnement [4].

Dans un contexte de santé, de confort et de sécurité, la Smart Room peut jouer plusieurs rôles. Elle peut fournir une information continue sur l'environnement, détecter des situations potentiellement dangereuses, aider à prévenir certains risques et transmettre des alertes à l'utilisateur ou à un proche. Pour les enfants, l'intérêt réside dans la surveillance des conditions de la chambre et la détection de situations inhabituelles. Pour les personnes âgées, l'intérêt concerne l'assistance au maintien à domicile, la détection de changements simples dans les routines et l'amélioration de la sécurité domestique. Ces usages doivent toutefois être pensés avec prudence afin d'éviter une surveillance excessive ou intrusive.

1.3 Internet des Objets (IoT) et environnements intelligents

L'Internet des Objets, ou IoT, désigne l'ensemble des objets physiques capables de collecter des données, de communiquer à travers un réseau et d'interagir avec des services numériques. Dans une Smart Room, les objets connectés peuvent être des capteurs, des microcontrôleurs, des modules de communication, des passerelles, des serveurs ou des applications mobiles. Leur rôle est de transformer un espace ordinaire en environnement mesurable et exploitable.

Une architecture IoT se compose généralement de plusieurs couches. La première est la couche de perception, qui regroupe les capteurs chargés d'acquérir les données. Dans le cas d'une chambre connectée, il peut s'agir de capteurs de température, d'humidité, de présence, de gaz, de fumée, de lumière ou d'autres paramètres environnementaux. La deuxième couche concerne la transmission des données, qui peut être réalisée par liaison série, Wi-Fi, Bluetooth, Zigbee ou d'autres protocoles selon les contraintes du système. La troisième couche correspond au traitement, local ou distant, permettant de nettoyer les données, de les calibrer, de les classifier ou de les stocker. Enfin, la couche applicative rend les résultats accessibles à l'utilisateur à travers une interface.

Dans le domaine de la santé connectée, les architectures IoT sont largement étudiées pour le suivi des patients, l'amélioration de la qualité de vie et la surveillance à distance [3]. Ces systèmes peuvent intégrer des capteurs physiologiques, des dispositifs portables,

des applications mobiles et des plateformes de traitement. Cependant, dans une chambre privée, le choix des données collectées doit rester proportionné à l'objectif poursuivi. Pour un système de surveillance environnementale, il n'est pas nécessaire de collecter des données visuelles ou biométriques lorsque des mesures simples suffisent à caractériser l'état de la pièce.

L'intégration de l'intelligence artificielle dans les environnements IoT permet d'aller au-delà de la simple lecture de valeurs. Les données collectées peuvent être utilisées pour entraîner des modèles capables de reconnaître un état, de détecter une anomalie ou de produire une alerte. Dans les smart homes, les algorithmes de reconnaissance d'activité humaine exploitent souvent des séquences issues de capteurs ambiants, de capteurs de contact ou de capteurs de mouvement [4, 9]. Dans le présent mémoire, l'intelligence artificielle intervient notamment dans la classification de l'état de la chambre à l'aide d'un RandomForestClassifier.

Le rôle d'une application mobile est également central. Une Smart Room ne doit pas seulement collecter des données ; elle doit les présenter de façon claire. L'application mobile permet d'afficher les valeurs capteurs, l'état prédit, les probabilités associées et les alertes. Elle joue donc un rôle d'interface entre le système technique et l'utilisateur final. Dans le système proposé, cette fonction est assurée par une application Flutter reliée à une API Flask.

L'IoT favorise ainsi la surveillance non intrusive en permettant de combiner des capteurs simples avec des traitements logiciels plus avancés. Le système peut rester matériellement léger tout en gagnant en intelligence grâce à la calibration, à la classification et à la transmission structurée des informations. Cette logique est particulièrement adaptée aux chambres connectées, où la discrétion du dispositif et la lisibilité des alertes sont des critères importants.

1.4 Types de capteurs utilisés dans les Smart Rooms

Les capteurs constituent la base matérielle de toute Smart Room. Ils permettent de transformer des phénomènes physiques ou environnementaux en données exploitables par un système numérique. Leur choix influence directement la qualité de la surveillance, le coût du prototype, la consommation énergétique, la facilité de déploiement et le niveau d'intrusion perçu par les utilisateurs.

Dans les travaux sur les smart homes, les capteurs peuvent être classés en plusieurs familles : capteurs environnementaux, capteurs de mouvement et de présence, capteurs de contact, capteurs physiologiques, microphones, caméras et dispositifs portables [2, 4]. Chaque famille possède ses avantages et ses limites. Les capteurs environnementaux sont

généralement peu intrusifs, mais ils ne donnent qu'une information indirecte sur l'état de la pièce. Les capteurs de présence fournissent une indication utile sur l'occupation ou le mouvement, mais ne décrivent pas l'identité de la personne. Les caméras et capteurs biométriques apportent des informations plus riches, mais soulèvent des questions éthiques et techniques plus fortes.

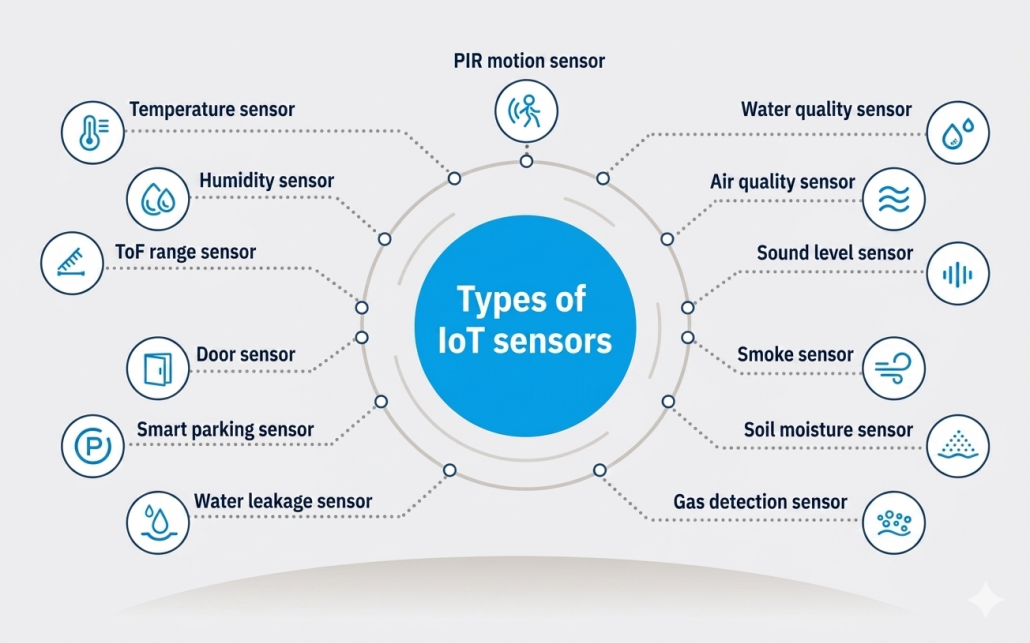


FIGURE 1.1 – Catégories de capteurs utilisés dans les Smart Rooms.

1.4.1 Capteurs environnementaux

Les capteurs environnementaux mesurent les conditions physiques de l'espace intérieur. Ils peuvent suivre la température, l'humidité, la qualité de l'air, la concentration de certains gaz, la présence de fumée, la luminosité, la pression atmosphérique ou encore certains paramètres acoustiques. Dans une chambre connectée, ces capteurs sont particulièrement utiles car ils renseignent directement sur le confort et la sécurité de l'environnement.

La température est un indicateur essentiel. Une chambre trop chaude ou trop froide peut être inconfortable, voire problématique pour des personnes sensibles. L'humidité joue également un rôle important, car une humidité élevée peut favoriser un inconfort respiratoire, la condensation ou une dégradation de la qualité de l'environnement intérieur. Les capteurs de gaz, de fumée ou de flamme sont quant à eux liés à la prévention des situations dangereuses. Les capteurs de lumière peuvent permettre d'évaluer l'ambiance lumineuse ou de détecter certains changements dans la pièce.

Dans notre système, le capteur DHT22 est utilisé pour mesurer la température et l'humidité de la chambre. Ce capteur présente l'avantage d'être accessible, simple à connecter à

un microcontrôleur et adapté à un prototype embarqué. Il illustre bien l'intérêt des capteurs low-cost dans les projets IoT. Toutefois, comme beaucoup de capteurs de ce type, ses mesures peuvent nécessiter une étape de correction. Cette limite justifie l'intérêt de l'étalonnage, qui sera étudié plus en détail dans le chapitre suivant.

Le système proposé utilise également un capteur Flame IR LM339 afin de détecter un risque de flamme. Ce capteur apporte une information liée à la sécurité de la chambre. Il ne permet pas de remplacer un dispositif industriel de sécurité incendie, mais il constitue un indicateur utile dans un prototype de surveillance intelligente. Son signal analogique peut être exploité pour distinguer différents niveaux de risque et alimenter une logique d'alerte.

1.4.2 Capteurs de mouvement et de présence

Les capteurs de mouvement et de présence permettent de savoir si une personne est présente dans une pièce ou si un mouvement est détecté. Ils sont largement utilisés dans les environnements intelligents, car ils fournissent une information utile sans nécessiter l'enregistrement d'images. Parmi les technologies courantes, on trouve les capteurs PIR, les capteurs infrarouges, les ultrasons, certains radars, les capteurs de pression et les capteurs de contact.

Les capteurs PIR, ou capteurs infrarouges passifs, détectent les variations de rayonnement infrarouge produites par le mouvement d'un corps chaud dans leur champ de détection. Ils sont très utilisés pour l'éclairage automatique, la sécurité domestique et la détection de présence. Leur principal avantage est leur caractère non visuel : ils indiquent la présence d'un mouvement sans produire d'image et sans identifier la personne. Ils sont également simples à intégrer dans un prototype Arduino.

Dans notre système, le capteur HC-SR501 permet de détecter la présence ou le mouvement dans la chambre sans recourir à une caméra. Cette caractéristique est cohérente avec l'objectif de non-intrusion. Le capteur ne fournit pas une description détaillée de l'activité humaine, mais il donne une information suffisante pour certains scénarios de surveillance, notamment la distinction entre présence et absence ou la prise en compte d'un mouvement dans l'état global de la chambre.

D'autres technologies peuvent être utilisées dans les Smart Rooms. Les capteurs ultrasoniques peuvent détecter une distance ou une présence en émettant des ondes sonores. Les radars peuvent suivre des mouvements plus fins dans certains contextes. Les capteurs de pression peuvent être intégrés dans un lit, un fauteuil ou un tapis pour détecter une occupation. Ces solutions peuvent être utiles, mais elles introduisent parfois des contraintes supplémentaires de coût, de positionnement ou d'interprétation.

1.4.3 Capteurs biométriques et caméras

Les capteurs biométriques et les caméras font partie des technologies existantes dans l'état de l'art des environnements intelligents. Les caméras peuvent être utilisées avec des méthodes de vision par ordinateur pour analyser une scène. Les microphones peuvent permettre l'analyse de sons ou d'événements acoustiques. Les capteurs cardiaques, respiratoires ou physiologiques peuvent fournir des informations précises sur l'état d'une personne. Ces technologies sont souvent étudiées dans des systèmes avancés de santé connectée ou de surveillance domestique [3, 2].

Cependant, ces dispositifs présentent des limites importantes dans une chambre. Les caméras collectent des informations visuelles sensibles et peuvent être perçues comme intrusives. Les microphones peuvent capter des conversations ou des sons privés. Les capteurs biométriques manipulent des données personnelles particulièrement sensibles, dont le stockage, la transmission et l'exploitation doivent être strictement encadrés. De plus, certaines technologies dépendent fortement des conditions d'utilisation, comme l'éclairage, la position de la personne ou la qualité du signal.

L'acceptabilité de ces systèmes est donc un enjeu majeur. Les travaux portant sur la perception de la surveillance dans les smart homes montrent que les utilisateurs, notamment les adultes âgés, peuvent exprimer des préoccupations liées à la confidentialité, à la sécurité des données, à l'autonomie et à la finalité de la collecte [7]. Une technologie techniquement performante n'est pas nécessairement acceptable si elle donne le sentiment d'une surveillance permanente ou excessive.

Pour cette raison, notre projet privilégie une solution non intrusive sans caméra ni biométrie. Le système collecte uniquement des mesures environnementales et événementielles : température, humidité, présence ou mouvement, et risque de flamme. Cette stratégie limite les données sensibles tout en permettant une surveillance pertinente de l'état de la chambre.

1.5 Applications de surveillance des enfants et des adultes

Les Smart Rooms peuvent être utilisées dans plusieurs contextes liés à la surveillance domestique. Pour les enfants, l'objectif principal est souvent de garantir un environnement sûr et confortable. La surveillance peut porter sur la température de la chambre, l'humidité, la présence d'un mouvement ou l'apparition d'un risque. Pour les personnes âgées, la surveillance peut contribuer à l'assistance au maintien à domicile, au suivi de routines simples et à la détection de changements dans l'environnement. Dans les deux cas, il est

important que le système reste compréhensible, discret et fiable.

La surveillance environnementale d'une chambre repose sur l'idée que certains paramètres peuvent signaler une situation nécessitant une attention. Une température élevée peut indiquer un inconfort ou un risque thermique. Une humidité anormale peut traduire une mauvaise qualité de l'environnement. Une détection de présence ou d'absence peut contribuer à l'interprétation de l'état de la pièce. Un risque de flamme, même détecté de manière simple, doit pouvoir déclencher une alerte rapide. Ces scénarios correspondent directement au positionnement du système proposé.

Les travaux existants montrent une grande diversité de solutions. Certaines approches se concentrent sur les capteurs ambiants et les données non intrusives. D'autres exploitent des dispositifs plus riches, comme des capteurs physiologiques ou des technologies visuelles, mais avec des limites plus importantes en termes de confidentialité et d'acceptabilité. Le tableau 1.1 présente une synthèse de travaux réels liés aux Smart Rooms, aux smart homes et à la surveillance intelligente.

Tableau 1.1 – Travaux existants sur les Smart Rooms et la surveillance intelligente

Auteur / Année	**Capteurs utilisés**	**Algorithme**	**Solution proposée**	**Résultat / précision**	**Limites**
Wang et al. (2021) [2]	PIR, capteurs de contact, pression, courant électrique, capteurs environnementaux, microphones et caméras comme technologies existantes	Revue et taxonomie des solutions de surveillance non intrusive	Analyse des technologies de santé connectée dans les espaces privés et les smart homes	Non précisée ; revue de 55 travaux	Peu de preuves cliniques à grande échelle ; les caméras et certaines données physiologiques posent des enjeux de confidentialité
Abdulmalek et al. (2022) [3]	Capteurs de température, fréquence cardiaque, SpO2, pression artérielle, ECG, dispositifs portables et passerelles IoT	Revue systématique des architectures IoT de santé	Synthèse des systèmes de surveillance médicale connectée pour améliorer la qualité de vie	Non précisée ; résultats variables selon les systèmes étudiés	Interopérabilité, sécurité, confidentialité et fiabilité des réseaux IoT
Bouchabou et al. (2021) [4]	Capteurs IoT ambiants, capteurs de mouvement, capteurs d'objets, RFID et capteurs portables	Méthodes data-driven, knowledge-driven, apprentissage classique et deep learning	Revue des méthodes de reconnaissance d'activité humaine dans les smart homes	Variable selon les datasets et scénarios	Activités concurrentes, multi-occupants, dérive temporelle et manque de généralisation
Cook et al. (2013) [8]	Capteurs de mouvement, contact de portes ou objets, température, lumière et infrastructure domotique CASAS	Apprentissage automatique et architecture adaptative pour smart home	Plateforme "Smart Home in a Box" pour collecter et analyser les activités à domicile	Non précisée dans l'article de synthèse de la plateforme	Nécessite une infrastructure de capteurs ; dépend de la qualité du déploiement et de l'acceptation par l'utilisateur
Chen et al. (2012) [9]	Flux multi-capteurs dans un environnement smart home	Raisonnement à base d'ontologies et approche knowledge-driven	Reconnaissance continue des activités de la vie quotidienne à partir de connaissances métier et de données capteurs	Non précisée ici ; performance dépendante des activités et du modèle de connaissance	Construction de l'ontologie et adaptation aux habitudes réelles peuvent être coûteuses
Susnea et al. (2019) [5]	Capteurs binaires low-cost pour routine quotidienne de personnes âgées vivant seules	Analyse de routine à partir d'événements binaires	Surveillance non intrusive des habitudes quotidiennes pour détecter des changements de comportement	Non précisée	Approche limitée aux signaux binaires ; interprétation dépendante du contexte de logement
Yu et al. (2019) [6]	Capteurs non intrusifs pour comportements quotidiens et environnement bâti : sommeil, cuisine, eau, température, fenêtres, portes	Analyse descriptive de données de capteurs	Étude pilote d'une smart home pour personnes âgées avec suivi continu à domicile	Faisabilité démontrée ; précision non précisée	Étude pilote avec une seule participante ; généralisation limitée
Campbell et al. (2024) [7]	Technologies smart home, IoT, capteurs domestiques, assistants et dispositifs de surveillance	Scoping review centrée sur perception, vie privée, risques et bénéfices	Analyse de l'acceptabilité des technologies de surveillance chez les adultes de 50 ans et plus	Non précisée ; revue qualitative des perceptions	Préoccupations fortes sur confidentialité, sécurité, autonomie et finalité de la collecte des données

L'analyse de ces travaux met en évidence plusieurs tendances. D'abord, les capteurs IoT sont largement utilisés pour surveiller les environnements domestiques et fournir des informations sur les conditions de vie. Les capteurs de mouvement, les capteurs de contact et les capteurs environnementaux sont particulièrement fréquents dans les solutions non intrusives. Ensuite, les approches algorithmiques varient fortement selon les objectifs : certaines solutions se limitent à une analyse descriptive, tandis que d'autres utilisent des méthodes de reconnaissance d'activité, des ontologies, de l'apprentissage automatique ou des approches plus avancées.

Les travaux existants montrent également que les résultats dépendent beaucoup du contexte. Les performances ne sont pas toujours exprimées par une précision unique, car elles varient selon les datasets, les logements, les capteurs et les scénarios étudiés. C'est pourquoi il est préférable de rester prudent dans la comparaison des résultats. Lorsqu'une précision n'est pas clairement indiquée, elle ne doit pas être supposée.

Notre travail se positionne dans la catégorie des solutions non intrusives. Il se distingue par l'association de capteurs simples et accessibles avec une étape d'étalonnage intelligent avant la classification. Cette étape est importante, car elle vise à améliorer la qualité des mesures avant leur utilisation par le modèle de décision. L'originalité du système ne réside donc pas seulement dans la collecte des données, mais dans la chaîne complète : acquisition, calibration hybride, classification, API Flask et application Flutter avec notifications.

1.6 Enjeux éthiques et protection de la vie privée

La surveillance d'une chambre connectée soulève des enjeux éthiques importants. Une chambre est un espace intime, associé au repos, à la vie privée et à la sécurité personnelle. Toute technologie installée dans cet espace doit donc être pensée avec prudence. Le fait qu'un système soit techniquement possible ne signifie pas qu'il soit automatiquement acceptable. L'acceptabilité dépend de la nature des données collectées, de la finalité du traitement, de la transparence du système et du contrôle laissé à l'utilisateur.

Le premier principe éthique concerne la limitation des données sensibles. Un système de surveillance domestique doit collecter uniquement les informations nécessaires à son objectif. Dans le cas d'une surveillance environnementale, il est possible de se limiter à des mesures de température, d'humidité, de présence ou de risque. Cette approche réduit les risques liés à l'identification des personnes, à l'enregistrement de scènes privées ou à la conservation de données personnelles sensibles.

Le deuxième principe concerne le consentement et l'information. Les utilisateurs doivent comprendre ce que le système mesure, pourquoi il le mesure et comment les alertes sont

générées. Dans le cas des personnes âgées, les travaux sur la perception des technologies de surveillance montrent que l'acceptation dépend fortement de la confiance, du sentiment d'autonomie et de la perception des bénéfices [7]. Une solution qui améliore la sécurité mais réduit fortement le sentiment de liberté peut être rejetée ou mal vécue.

Le troisième principe concerne la sécurité des données. Les mesures collectées par une Smart Room doivent être protégées contre les accès non autorisés. Même lorsqu'elles ne contiennent pas d'image ou de donnée biométrique, elles peuvent révéler des habitudes de présence, des routines ou des informations sur l'environnement domestique. Il est donc nécessaire de limiter l'accès aux données, de structurer les échanges avec prudence et de privilégier des mécanismes de transmission adaptés.

Les capteurs non visuels offrent un avantage éthique important. Ils permettent de surveiller un état ou un événement sans produire d'image de la personne. Cette caractéristique ne supprime pas tous les risques, mais elle les réduit considérablement par rapport à des dispositifs visuels ou biométriques. Dans le système proposé, la collecte se limite à des mesures environnementales et événementielles. Le système ne collecte pas d'image, de voix ou de donnée biométrique. Cette limitation est cohérente avec l'objectif de concevoir une Smart Room respectueuse de la vie privée.

Enfin, la transparence des alertes constitue un élément essentiel. Une alerte doit être compréhensible pour l'utilisateur : elle doit indiquer l'état prédit, le niveau d'alerte et les valeurs associées lorsque cela est possible. L'application mobile joue ici un rôle important, car elle transforme les résultats techniques du modèle en information lisible. Dans notre système, l'API Flask et l'application Flutter permettent d'afficher les valeurs brutes, les valeurs calibrées, les probabilités et les notifications, ce qui contribue à rendre le système plus explicable.

1.7 Conclusion

Ce chapitre a présenté l'état de l'art des Smart Rooms et des technologies de surveillance utilisées dans les environnements intelligents. Les Smart Rooms représentent une évolution de la domotique classique vers des espaces capables de collecter, transmettre et analyser des données contextuelles. L'Internet des Objets joue un rôle central dans cette évolution, car il permet d'associer des capteurs, des microcontrôleurs, des API et des applications mobiles dans une architecture cohérente.

L'analyse des capteurs montre que plusieurs familles de technologies peuvent être utilisées dans une chambre connectée. Les capteurs environnementaux permettent de suivre la température, l'humidité, la qualité de l'air ou certains risques. Les capteurs de mouvement et de présence fournissent une information utile sans recourir à l'image. Les caméras,

microphones et capteurs biométriques existent dans l'état de l'art, mais ils soulèvent des questions importantes de confidentialité, d'acceptabilité et de sécurité des données.

Les travaux étudiés montrent l'intérêt des capteurs IoT pour la surveillance des personnes vulnérables, mais aussi la diversité des approches et des limites. Certains systèmes privilégient des dispositifs riches mais intrusifs, tandis que d'autres s'appuient sur des capteurs ambiants plus simples. Le présent mémoire se positionne dans cette seconde catégorie, en proposant une Smart Room non intrusive basée sur Arduino Nano, DHT22, HC-SR501 et Flame IR LM339. L'originalité du travail réside dans l'ajout d'une calibration hybride intelligente avant la classification de l'état de la chambre.

Cette analyse justifie la transition vers le chapitre suivant, consacré à l'étalonnage des capteurs et à l'intelligence artificielle. En effet, si les capteurs sont essentiels dans une Smart Room, leurs mesures peuvent être imprécises ou influencées par le contexte. L'étalonnage devient alors une étape fondamentale pour améliorer la qualité des données avant leur exploitation par un modèle de classification.

Chapitre 2

Étalonnage et intelligence artificielle

2.1 Introduction

Le chapitre précédent a montré que les Smart Rooms reposent principalement sur la collecte de données issues de capteurs. Ces capteurs permettent de transformer l'état d'un environnement intérieur en valeurs numériques exploitables par un système informatique. Cependant, la disponibilité d'une mesure ne garantit pas automatiquement sa fiabilité. Dans un système de surveillance intelligente, la qualité des données constitue une condition essentielle, car une valeur incorrecte ou instable peut conduire à une mauvaise interprétation de l'état de la chambre.

Dans le contexte de ce mémoire, le système proposé repose sur une carte Arduino Nano associée à plusieurs capteurs à faible coût : le DHT22 pour la température et l'humidité, le HC-SR501 pour la présence ou le mouvement, et le Flame IR LM339 pour le risque de flamme. Ces composants présentent l'avantage d'être accessibles, simples à intégrer et adaptés à un prototype de chambre connectée. Néanmoins, comme tout dispositif de mesure, ils peuvent être influencés par le bruit, les conditions environnementales, le positionnement, l'alimentation ou les limites propres au composant. L'étalonnage apparaît alors comme une étape nécessaire pour améliorer la cohérence des mesures.

Ce chapitre a pour objectif d'expliquer les bases théoriques nécessaires à la compréhension de l'étalonnage et de l'intelligence artificielle appliqués aux capteurs. Il présente d'abord la notion d'étalonnage, en distinguant la mesure brute, la valeur de référence et la valeur calibrée. Il expose ensuite les principales méthodes d'étalonnage : méthodes statistiques, méthodes expérimentales et méthodes automatiques. Une attention particulière est portée aux sources d'erreurs et aux incertitudes de mesure, car elles permettent de comprendre pourquoi les capteurs à faible coût peuvent nécessiter une correction.

Le chapitre introduit ensuite l'intelligence artificielle, le Machine Learning et le Deep Learning. Ces notions sont présentées de manière théorique, tout en restant liées au projet. L'objectif n'est pas encore de détailler l'implémentation expérimentale, qui sera présentée dans les chapitres suivants, mais de justifier les choix méthodologiques. En particulier, le chapitre explique pourquoi le Machine Learning est adapté aux données tabulaires issues de capteurs et pourquoi Random Forest représente un choix pertinent pour combiner calibration et classification dans une Smart Room non intrusive.

Enfin, ce chapitre distingue clairement deux opérations complémentaires : la calibration et la classification. La calibration vise à corriger les mesures des capteurs afin d'obtenir des valeurs plus fiables. La classification vise ensuite à interpréter ces valeurs pour prédire l'état de la chambre. Dans ce mémoire, cette distinction est centrale : la température est corrigée par une méthode statistique de biais moyen, l'humidité et le signal analogique du capteur Flame IR LM339 sont calibrés par RandomForestRegressor, puis l'état de la chambre est prédit par RandomForestClassifier.

2.2 Notion d'étalonnage des capteurs

L'étalonnage d'un capteur désigne l'ensemble des opérations permettant d'établir une relation entre la valeur mesurée par ce capteur et une valeur considérée comme référence. Dans un système de mesure, la valeur brute correspond à la donnée directement fournie par le capteur. Cette valeur peut être correcte, mais elle peut aussi contenir un biais, du bruit ou une erreur liée aux conditions d'acquisition. La valeur de référence correspond à une mesure plus fiable, obtenue par un instrument de référence, une procédure contrôlée ou une valeur attendue dans un scénario expérimental. La valeur calibrée est la valeur obtenue après correction de la mesure brute.

L'erreur de mesure peut être exprimée de manière simple comme la différence entre la valeur brute et la valeur de référence :

$$e = x_{\text{brut}} - x_{\text{reference}}$$

Cette équation permet de représenter l'écart observé entre ce que le capteur indique et ce que l'on considère comme plus proche de la réalité. L'objectif de la calibration est de réduire cet écart ou, au minimum, de le rendre plus prévisible. Une fonction de calibration peut alors être définie de manière générale comme une transformation appliquée à la valeur brute :

$$x_{\text{calibre}} = f(x_{\text{brut}})$$

La fonction f peut être très simple, par exemple une correction par soustraction d'un biais moyen, ou plus complexe, par exemple un modèle d'apprentissage automatique. Dans tous les cas, la finalité reste la même : produire une valeur plus fiable avant de l'utiliser dans une décision ou une classification.

Dans les systèmes IoT, l'étalonnage joue un rôle important. Les capteurs utilisés dans les prototypes connectés sont souvent choisis pour leur coût réduit, leur disponibilité et leur facilité d'intégration. Ces qualités sont importantes pour concevoir un système accessible, mais elles peuvent s'accompagner d'une précision limitée. Les travaux sur les capteurs à faible coût montrent que des méthodes de calibration, notamment basées sur l'apprentissage automatique, peuvent améliorer les performances de mesure dans certains contextes environnementaux [1, 10, 15].

Un exemple simple peut être donné avec la température. Si un capteur indique régulièrement une température légèrement supérieure à une valeur de référence, il est possible de calculer un biais moyen puis de le soustraire aux mesures futures. La valeur brute devient alors une valeur corrigée. Ce type de correction ne suppose pas nécessairement une relation complexe entre les variables ; il est adapté lorsque l'erreur est relativement stable.

Dans le présent projet, la calibration corrige les valeurs issues des capteurs avant leur utilisation par le modèle de classification. Elle constitue donc une étape intermédiaire entre l'acquisition des données et la décision. Le système s'appuie sur un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Cette structuration permet de conserver une cohérence entre les variables produites par le prototype, les valeurs calibrées et les classes utilisées pour décrire l'état de la chambre.

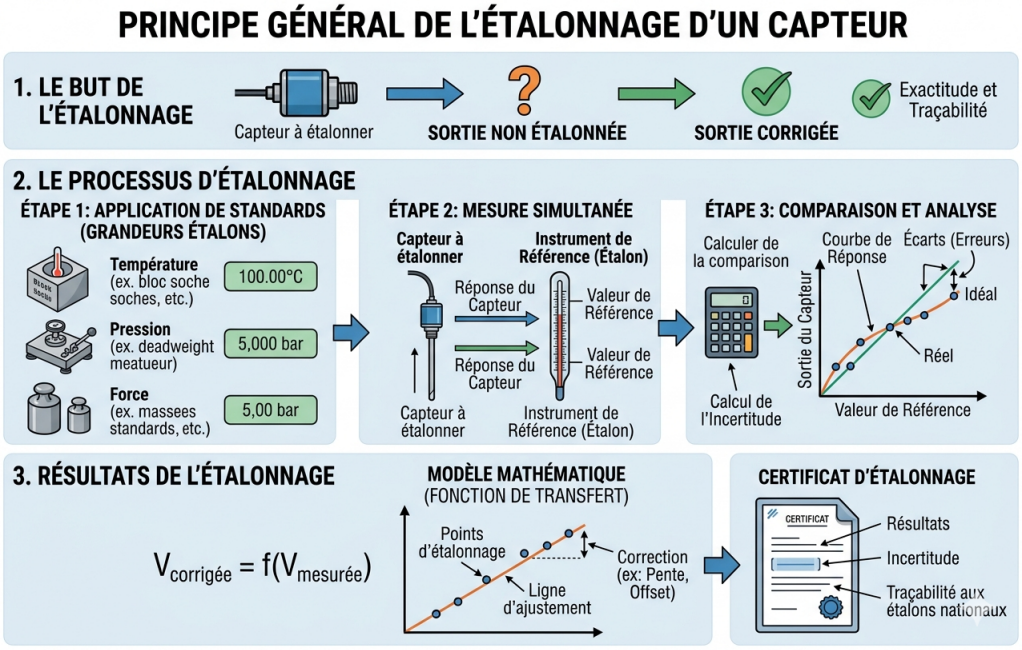


FIGURE 2.1 – Principe général de l'étalonnage d'un capteur.

2.3 Méthodes d'étalonnage

Les méthodes d'étalonnage peuvent être classées selon leur niveau de complexité et leur manière de modéliser l'erreur. Certaines méthodes reposent sur des statistiques simples, d'autres sur des procédures expérimentales contrôlées, et d'autres encore sur des modèles automatiques capables d'apprendre une relation à partir des données. Le choix d'une méthode dépend du type de capteur, de la nature de l'erreur, de la disponibilité des données de référence et de l'objectif final du système.

2.3.1 Méthodes statistiques

Les méthodes statistiques constituent souvent le premier niveau d'étalonnage. Elles sont simples à mettre en oeuvre, compréhensibles et adaptées lorsque l'erreur présente une structure régulière. Parmi ces méthodes, on trouve la moyenne, l'écart-type, la moyenne mobile, les filtres simples, la correction linéaire et la correction par biais moyen.

La correction par biais moyen consiste à calculer l'écart moyen entre les mesures brutes et les valeurs de référence, puis à appliquer cette correction aux nouvelles mesures. Dans le cas de la température, le biais peut être défini comme suit :

$$biais_{\text{temperature}} = \frac{1}{n} \sum_{i=1}^{n} (T_{\text{brute},i} - T_{\text{reference},i})$$

La valeur calibrée est ensuite obtenue par :

$$T_{\text{calibree}} = T_{\text{brute}} - biais_{\text{temperature}}$$

Cette méthode est particulièrement adaptée lorsque le capteur présente un décalage relativement constant. Elle a l'avantage d'être légère, rapide et facile à interpréter. En revanche, elle devient limitée lorsque l'erreur dépend de plusieurs variables ou varie selon les conditions environnementales.

Dans notre projet, la température est corrigée par une correction par biais moyen. Ce choix est cohérent lorsque l'objectif est de compenser un écart global entre la mesure brute du DHT22 et une valeur de référence. La correction statistique permet ainsi de traiter une variable de manière simple, sans recourir inutilement à un modèle plus complexe.

Les moyennes mobiles et filtres simples peuvent également être utilisés pour réduire les fluctuations ponctuelles. Ils ne corrigent pas nécessairement un biais permanent, mais permettent de lisser des valeurs instables. Dans un système de surveillance, ce type de filtrage peut éviter qu'une variation très courte ne provoque une interprétation excessive. Toutefois, un lissage trop fort peut aussi retarder la détection d'un événement important. Le choix du filtre doit donc rester adapté au scénario.

2.3.2 Méthodes expérimentales

Les méthodes expérimentales reposent sur la comparaison des mesures du capteur avec des valeurs de référence obtenues dans des conditions contrôlées. Elles peuvent impliquer un instrument de référence, la répétition des mesures, la variation progressive d'une grandeur physique ou la construction d'une courbe de calibration. Ces méthodes sont fondamentales, car elles fournissent la base à partir de laquelle une correction peut être définie.

Dans une démarche expérimentale, il est important de répéter les mesures afin d'évaluer la stabilité du capteur. Une seule mesure ne suffit pas à caractériser un comportement, car elle peut être influencée par une fluctuation ponctuelle. La répétition permet d'estimer la dispersion des valeurs, de repérer les erreurs systématiques et de vérifier si la correction reste valable dans plusieurs conditions.

Les mesures dans des conditions contrôlées permettent également de mieux comprendre la relation entre la valeur brute et la valeur de référence. Par exemple, un capteur d'humidité peut être observé dans plusieurs niveaux d'humidité afin de vérifier si l'erreur reste constante ou si elle varie selon la plage de mesure. De même, un capteur de flamme peut être sensible à la distance, à l'angle ou à la lumière infrarouge ambiante. Une calibration expérimentale permet donc de mieux identifier les facteurs qui influencent le signal.

Les courbes de calibration représentent graphiquement la relation entre la mesure brute et la valeur de référence. Si la relation est approximativement linéaire, une correction linéaire peut suffire. Si la relation est plus complexe, une méthode automatique peut devenir préférable. La reproductibilité constitue un critère essentiel : une méthode de calibration doit produire des corrections cohérentes lorsque l'expérience est répétée dans des conditions comparables.

Dans le cadre de ce mémoire, les méthodes expérimentales servent de base à la définition de valeurs de référence et de scénarios contrôlés. Elles ne sont pas présentées ici comme une implémentation complète, mais comme un principe méthodologique nécessaire à la construction d'un dataset cohérent et à l'évaluation des corrections appliquées.

2.3.3 Méthodes automatiques

Les méthodes automatiques d'étalonnage utilisent des modèles capables d'apprendre une relation entre des variables d'entrée et une valeur cible. Elles sont particulièrement utiles lorsque l'erreur du capteur ne peut pas être décrite par un simple biais ou par une relation linéaire. Dans ce cas, les algorithmes d'apprentissage supervisé peuvent modéliser des dépendances plus complexes.

La régression linéaire est l'une des méthodes automatiques les plus simples. Elle cherche à approximer une relation sous la forme d'une droite ou d'une combinaison linéaire de variables. Elle est facile à interpréter, mais peut être insuffisante lorsque la relation entre la mesure brute et la valeur de référence est non linéaire. Les régressions non linéaires, les arbres de décision, les forêts aléatoires et certains réseaux de neurones peuvent alors être utilisés.

RandomForestRegressor est un modèle d'ensemble basé sur plusieurs arbres de décision. Chaque arbre apprend une relation à partir d'un sous-ensemble des données et des variables. Les prédictions des arbres sont ensuite agrégées afin de produire une estimation plus robuste. Cette méthode est souvent adaptée aux données tabulaires, supporte des relations non linéaires et peut être relativement robuste au bruit. Les travaux de Zimmerman et al. montrent l'intérêt de Random Forest pour améliorer la calibration de capteurs à faible coût dans un contexte de qualité de l'air [1]. D'autres travaux comparent également plusieurs algorithmes de régression pour la calibration de capteurs environnementaux [10, 15].

Dans notre projet, l'humidité et la valeur analogique du capteur Flame IR LM339 sont calibrées avec RandomForestRegressor. Ce choix est motivé par le fait que ces signaux peuvent présenter des relations plus complexes que la température. L'humidité peut dépendre de variations locales et de conditions environnementales, tandis que le signal de

flamme peut être influencé par la distance, l'angle, le seuil et certaines perturbations lumineuses. Une méthode automatique permet donc de mieux représenter ces relations qu'une simple correction moyenne.

2.4 Sources d'erreurs et incertitudes de mesure

Les erreurs de mesure peuvent provenir de plusieurs sources. Dans un système embarqué, elles ne sont pas uniquement liées au capteur lui-même ; elles peuvent aussi dépendre du circuit, de l'alimentation, de l'environnement et du mode d'acquisition. Comprendre ces sources d'erreurs est indispensable pour concevoir une calibration pertinente.

Le bruit électronique peut provoquer des variations dans les lectures, en particulier sur les signaux analogiques. Une alimentation instable peut également influencer les valeurs mesurées. L'imprécision propre au composant, indiquée dans sa documentation technique, constitue une autre source d'incertitude. À cela s'ajoutent les conditions environnementales : ventilation, température locale, humidité, lumière ambiante ou présence d'obstacles.

Le positionnement du capteur joue également un rôle important. Un capteur de température placé trop près d'une source de chaleur ou dans une zone mal ventilée peut fournir une mesure qui ne représente pas correctement l'ensemble de la chambre. Un capteur PIR peut ne pas détecter correctement un mouvement si l'angle ou la distance ne sont pas adaptés. Un capteur de flamme peut être influencé par une lumière infrarouge externe, par la distance à la source ou par le seuil de sensibilité.

Enfin, les capteurs peuvent présenter un délai de réponse ou un vieillissement progressif. Le délai de réponse signifie que le capteur ne réagit pas instantanément à une variation de l'environnement. Le vieillissement peut modifier son comportement au fil du temps. Ces éléments renforcent l'importance du prétraitement, de la calibration et de l'analyse prudente des mesures.

Tableau 2.1 – Sources d'erreurs possibles des capteurs utilisés

Capteur	**Grandeur mesurée**	**Sources d'erreurs possibles**	**Effet sur la surveillance**	**Correction proposée**
DHT22	Température	Bruit, délai de réponse, positionnement, ventilation	Mauvaise estimation de température	Correction par biais moyen
DHT22	Humidité	Instabilité hygrométrique, délai de réponse, variations locales	Fausse alerte humidité	RandomForestRegressor
HC-SR501	Présence / mouvement	Sensibilité, angle, mouvement parasite, zone de détection	Fausse présence ou absence	Prise en compte dans la classification
Flame IR LM339	Risque de flamme / signal analogique	Lumière infrarouge externe, distance, angle, seuil	Faux risque de flamme	RandomForestRegressor + classification
Arduino Nano / alimentation	Lecture des signaux	Bruit électrique, alimentation instable	Variations de lecture	Filtrage et prétraitement

Le Tableau 2.1 synthétise les sources d'erreurs possibles des composants utilisés dans le système proposé. Il montre que chaque capteur peut être affecté par des facteurs spécifiques. La correction ne doit donc pas être uniforme pour toutes les variables. Cette observation justifie l'approche hybride adoptée dans le mémoire : une méthode statistique simple pour la température, des méthodes d'apprentissage automatique pour l'humidité et le signal analogique de flamme, puis une classification globale de l'état de la chambre.

2.5 Définition de l'intelligence artificielle

L'intelligence artificielle peut être définie comme un ensemble de méthodes permettant à une machine d'accomplir des tâches qui nécessitent habituellement certaines formes de raisonnement, d'apprentissage, d'adaptation ou de prise de décision. Dans un système technique, l'IA ne signifie pas nécessairement une autonomie complète ; elle peut simplement désigner la capacité à transformer des données en informations utiles.

Dans les Smart Rooms, l'intelligence artificielle peut intervenir à plusieurs niveaux. Elle peut aider à filtrer les données, à corriger des mesures, à reconnaître un état, à détecter une anomalie ou à produire une alerte. L'intérêt de l'IA est donc de donner du sens aux mesures capteurs. Une température, une humidité ou une valeur de flamme ne sont que des nombres ; un modèle d'analyse peut les transformer en état interprétable, comme une situation normale, une chambre humide, une chambre chaude ou un risque critique.

Dans le cadre du présent mémoire, l'intelligence artificielle est utilisée de manière ciblée. Elle n'a pas pour objectif de remplacer l'ensemble de la logique du système, mais d'améliorer deux étapes importantes : l'étalonnage de certaines mesures et la classification de

l'état de la chambre. Cette approche permet de conserver une architecture simple tout en bénéficiant de modèles capables de traiter des relations plus complexes que des règles fixes.

2.5.1 Définition du Machine Learning

Le Machine Learning, ou apprentissage automatique, est une branche de l'intelligence artificielle qui permet à un modèle d'apprendre à partir de données. Au lieu de programmer explicitement toutes les règles de décision, on fournit au modèle des exemples à partir desquels il ajuste ses paramètres. Dans un apprentissage supervisé, les données d'entrée sont associées à une valeur cible. La cible peut être une valeur numérique dans le cas d'une régression, ou une classe dans le cas d'une classification.

Dans un problème de régression, le modèle cherche à prédire une valeur continue. Pour l'étalonnage des capteurs, la régression peut être utilisée pour prédire une valeur calibrée à partir d'une ou plusieurs mesures brutes. Dans un problème de classification, le modèle cherche à attribuer une classe à une observation. Pour une Smart Room, la classification peut permettre de prédire l'état de la chambre à partir des valeurs capteurs.

Les données issues de capteurs sont souvent tabulaires : chaque ligne représente une observation, et chaque colonne représente une variable. Ce format est particulièrement compatible avec de nombreux algorithmes de Machine Learning, comme la régression linéaire, les arbres de décision, Random Forest, KNN ou SVM. Dans les smart homes, ces méthodes peuvent être utilisées pour interpréter des événements, reconnaître des activités ou classifier des états [4, 9].

Dans notre projet, deux usages du Machine Learning sont distingués. Le RandomForestRegressor est utilisé pour l'étalonnage de l'humidité et du signal analogique de flamme. Le RandomForestClassifier est utilisé pour classifier l'état de la chambre. Cette séparation permet de clarifier le rôle de chaque modèle : le premier corrige des valeurs, le second prend une décision sur l'état global de l'environnement.

2.5.2 Définition du Deep Learning

Le Deep Learning, ou apprentissage profond, est un sous-domaine du Machine Learning fondé sur des réseaux de neurones profonds. Ces modèles sont composés de plusieurs couches capables d'apprendre des représentations complexes. Ils sont particulièrement utilisés pour traiter des images, des signaux audio, du texte ou de très grands volumes de données.

Le Deep Learning présente plusieurs avantages. Il peut apprendre automatiquement des caractéristiques pertinentes, gérer des données complexes et atteindre de très bonnes per-

formances dans certains domaines. Dans les smart homes, il est étudié pour la reconnaissance d'activité humaine et l'analyse de séquences de capteurs [4]. Cependant, il présente aussi des limites : il nécessite souvent davantage de données, plus de puissance de calcul et une interprétation plus difficile.

Dans le contexte de ce mémoire, le Deep Learning n'est pas le choix principal. Le système repose sur un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Les données sont tabulaires, le nombre de variables est limité et l'objectif est de conserver une approche robuste, explicable et adaptée à un prototype. Random Forest constitue donc un compromis plus approprié entre performance, simplicité et interprétabilité.

2.6 Algorithmes appliqués à l'étalonnage et à la surveillance

Plusieurs familles d'algorithmes peuvent être appliquées à l'étalonnage des capteurs et à la surveillance intelligente. Les méthodes statistiques et linéaires sont simples, rapides et interprétables. Elles sont adaptées lorsque la relation entre la mesure brute et la valeur corrigée est relativement stable. Les arbres de décision et les forêts aléatoires permettent de modéliser des relations non linéaires. Les méthodes comme SVM ou KNN peuvent être efficaces pour certains jeux de données, mais leur comportement dépend fortement du choix des paramètres, de l'échelle des variables et de la structure des données. Les réseaux de neurones et le Deep Learning sont puissants, mais nécessitent souvent des volumes de données plus importants.

Les approches knowledge-driven, comme celles basées sur des ontologies, utilisent des connaissances métier pour interpréter les activités ou les états d'un environnement. Elles peuvent être pertinentes lorsque les règles de contexte sont bien connues, mais elles demandent une modélisation préalable importante [9]. Les approches data-driven, quant à elles, apprennent directement à partir des données. Les travaux sur les smart homes montrent que ces approches sont nombreuses et que leurs performances varient selon les datasets, les capteurs et les scénarios [4].

Random Forest occupe une place particulière dans ce mémoire. Il s'agit d'un modèle d'ensemble composé de plusieurs arbres de décision. Chaque arbre apprend sur une partie des données et produit une prédiction. En régression, les prédictions des arbres sont généralement moyennées. En classification, les arbres votent pour une classe. Cette agrégation réduit le risque de surapprentissage par rapport à un arbre isolé et améliore la robustesse du modèle.

Random Forest est bien adapté aux données tabulaires issues de capteurs. Il peut gérer des relations non linéaires, supporter un certain niveau de bruit et fonctionner sans nécessiter une transformation très complexe des variables. Ces caractéristiques expliquent son intérêt pour la calibration de capteurs à faible coût [1] et pour des tâches de classification dans des systèmes de surveillance.

Dans ce mémoire, RandomForestRegressor est utilisé pour calibrer l'humidité et *flame_analog*. RandomForestClassifier est utilisé pour prédire l'état de la chambre. La température, quant à elle, est corrigée avec une méthode statistique de biais moyen. Cette combinaison constitue une calibration hybride intelligente, car elle associe une correction simple lorsque cela est suffisant et un modèle d'apprentissage lorsque la relation est plus complexe.

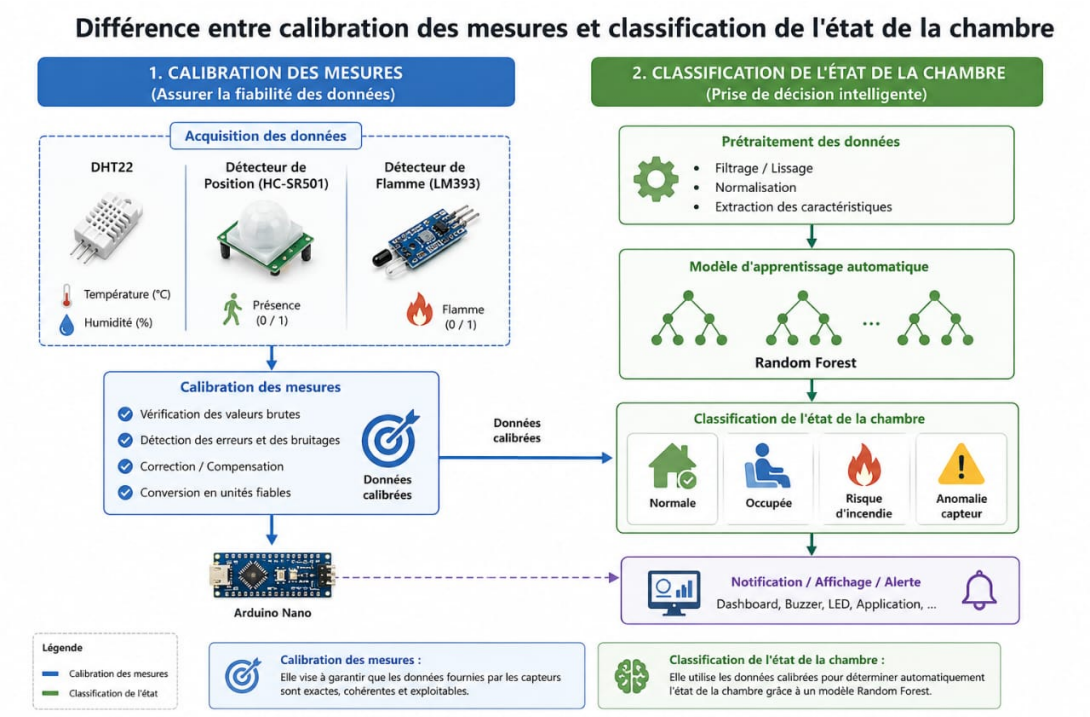


FIGURE 2.2 – Différence entre calibration des capteurs et classification de l'état de la chambre.

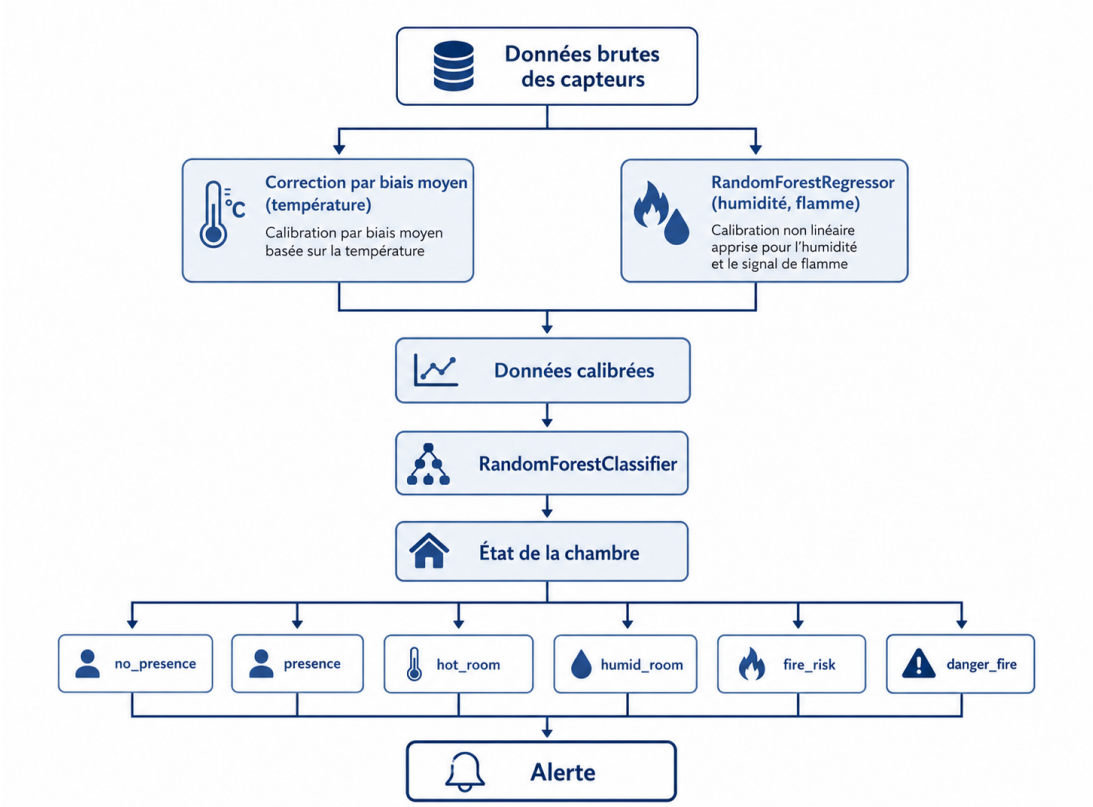


FIGURE 2.3 – Pipeline de calibration hybride et de classification de l'état de la chambre.

TABLEAU 2.2 – Algorithmes appliqués à l'étalonnage des capteurs et à la surveillance intelligente

Auteur / Année	Domaine	Algorithme	Solution	Résultat / précision	Limites
Zimmerman et al. (2018) [1]	Calibration de capteurs à faible coût de qualité de l'air	Random Forest Regression	Modèle de calibration supervisé pour améliorer les mesures de capteurs à faible coût	Erreurs relatives rapportées : environ $< 5\%$ pour CO2, 10–15% pour CO et O3, et 30% pour NO2	Résultats liés au contexte de colocalisation et aux polluants étudiés ; transfert à d'autres capteurs à valider
Han et al. (2021) [10]	Calibration de capteurs de gaz à faible coût	Régression linéaire simple, régression linéaire multiple, Random Forest, LSTM	Comparaison de méthodes statistiques et ML pour CO, NO2, O3 et SO2	Variable selon le gaz et le modèle ; plusieurs calibrations améliorent les mesures brutes	Sensibilité aux facteurs environnementaux ; besoin de données de référence fiables
Kumar et Sahu (2021) [15]	Calibration de capteur PM2.5 à faible coût	MLR, Lasso, Ridge, SVR, kNN, MLP, arbre de régression, Random Forest, Gradient Boosting	Évaluation comparative de neuf algorithmes de régression pour calibrer un Plantower PMS5003	Scores de test rapportés : kNN 0,97, RF 0,96, GB 0,95	Risque de surapprentissage ; performances dépendantes des conditions de température et d'humidité
Bouchabou et al. (2021) [4]	Reconnaissance d'activité dans les smart homes	Algorithmes classiques, approches data-driven, CNN, LSTM et deep learning	Taxonomie des méthodes de reconnaissance d'activité à partir de capteurs IoT ambiants	Variable selon les datasets et protocoles d'évaluation	Deep Learning prometteur mais sensible au manque de données, à la dérive et à la variabilité des logements
Chen et al. (2012) [9]	Surveillance smart home et reconnaissance d'activité	Ontologies, raisonnement sémantique, approche knowledge-driven	Modélisation explicite du contexte et des activités pour la reconnaissance en temps réel	Non précisée dans le tableau ; dépend du scénario expérimental	Nécessite une expertise métier et une modélisation préalable détaillée
Cook et al. (2013) [8]	Smart environments	Apprentissage automatique et infrastructure CASAS	Collecte et analyse d'activités domestiques à partir de capteurs ambiants	Non précisée dans l'article de synthèse de la plateforme	Dépend de l'installation des capteurs, du contexte du logement et de l'acceptation par l'utilisateur
Abdulmalek et al. (2022) [3]	IoT healthcare monitoring	Traitement de données capteurs, architectures IoT, parfois IA/ML selon les systèmes étudiés	Revue des architectures de surveillance santé connectée et de leurs composants	Variable selon les applications ; précision non uniformisée	Hétérogénéité des capteurs, sécurité, consommation énergétique et standardisation
Wang et al. (2021) [2]	Surveillance non intrusive en espace privé	Interprétation de données comportementales, physiologiques et environnementales	Cadre d'analyse des capteurs et fonctions de monitoring : fonctionnel, urgence, sécurité, social, physiologique	Non précisée ; revue de technologies et usages	Preuves médicales encore limitées ; enjeux de confidentialité et d'acceptabilité

Le Tableau 2.2 montre la diversité des approches utilisées dans la littérature. Les mé-

thodes statistiques et linéaires sont simples, mais elles peuvent être limitées lorsque les relations entre variables deviennent complexes. Les méthodes automatiques, comme Random Forest, permettent de mieux modéliser ces relations et d'améliorer la robustesse des prédictions dans certains contextes. Le Deep Learning est puissant, mais il nécessite souvent plus de données et offre une interprétation moins directe.

Ainsi, Random Forest représente un compromis adapté à notre projet. Il permet d'appliquer une régression pour la calibration et une classification pour la décision, tout en restant compatible avec des données tabulaires issues de capteurs. Cette flexibilité justifie son utilisation dans une Smart Room non intrusive où l'objectif est de produire des mesures corrigées puis un état de chambre interprétable.

2.7 Conclusion

Ce chapitre a présenté les fondements théoriques de l'étalonnage des capteurs et de l'intelligence artificielle appliqués à une Smart Room. L'étalonnage apparaît comme une étape essentielle pour améliorer la fiabilité des mesures, en particulier lorsque le système repose sur des capteurs à faible coût. Les erreurs peuvent provenir du capteur, du circuit, de l'environnement, de l'alimentation ou du positionnement. Les corriger permet d'obtenir des données plus cohérentes avant leur utilisation par un modèle de décision.

Les méthodes d'étalonnage peuvent être statistiques, expérimentales ou automatiques. Les méthodes statistiques, comme la correction par biais moyen, sont simples et adaptées lorsque l'erreur est relativement stable. Les méthodes expérimentales permettent d'établir des valeurs de référence et d'analyser la reproductibilité. Les méthodes automatiques, notamment celles basées sur le Machine Learning, sont pertinentes lorsque les relations entre variables sont non linéaires ou difficiles à modéliser manuellement.

L'intelligence artificielle permet de transformer des mesures capteurs en informations interprétables. Le Machine Learning est particulièrement adapté aux données tabulaires issues du système proposé. Le Deep Learning constitue une approche avancée, mais il n'est pas le choix principal dans ce mémoire en raison du format des données, de la taille du dataset et du besoin d'interprétabilité. Random Forest est retenu comme un compromis entre performance, robustesse et simplicité.

Dans la continuité de ce chapitre, le chapitre suivant présentera la conception et la mise en oeuvre du système proposé. Il montrera comment les capteurs, la carte Arduino Nano, le dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés, la calibration hybride intelligente et la classification sont organisés dans une architecture complète de chambre connectée.

Chapitre 3

Conception et mise en œuvre du système proposé

3.1 Introduction

Les chapitres précédents ont présenté le contexte des Smart Rooms, les technologies de surveillance non intrusive, ainsi que les principes théoriques de l'étalonnage et de l'intelligence artificielle. Le présent chapitre s'inscrit dans la continuité de cette analyse en décrivant la conception du système proposé. Il ne s'agit plus seulement d'étudier les concepts, mais d'organiser les différents éléments matériels et logiciels dans une architecture cohérente destinée à surveiller l'état d'une chambre connectée.

Le système proposé est une Smart Room non intrusive destinée à la surveillance des enfants et des personnes âgées comme cas d'usage. Il ne cherche pas à identifier les personnes, mais à suivre l'état de la chambre à travers des mesures environnementales et événementielles : température, humidité, présence ou mouvement, risque de flamme et alertes. Cette orientation permet de limiter la nature des données collectées et de privilégier une solution simple, accessible et respectueuse de l'espace privé.

La conception du système repose sur une chaîne complète allant de l'acquisition des signaux jusqu'à l'affichage mobile. Les capteurs DHT22, HC-SR501 et Flame IR LM339 sont reliés à une carte Arduino Nano, utilisée comme unité principale d'acquisition. Les données sont ensuite organisées sous forme d'un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Une calibration hybride intelligente est appliquée avant la classification de l'état de la chambre par un modèle RandomForestClassifier. Les résultats sont ensuite exposés par une API Flask et affichés dans une application Flutter avec notifications.

L'objectif de ce chapitre est donc de présenter l'organisation générale du système, les choix matériels, le format des données, la méthodologie d'étalonnage et les modèles d'intelligence artificielle utilisés. Les résultats numériques détaillés, les performances et les discussions expérimentales seront réservés au chapitre suivant.

3.2 Architecture globale de la Smart Room

L'architecture globale du système proposé suit une logique en plusieurs étapes. Les capteurs mesurent l'environnement de la chambre, l'Arduino Nano acquiert les signaux, les données sont structurées, puis une calibration hybride intelligente corrige certaines valeurs brutes. Les valeurs calibrées sont ensuite utilisées par un modèle de classification afin de prédire l'état global de la chambre. Enfin, une API Flask fournit les résultats sous forme JSON à une application Flutter chargée de l'affichage et des notifications.

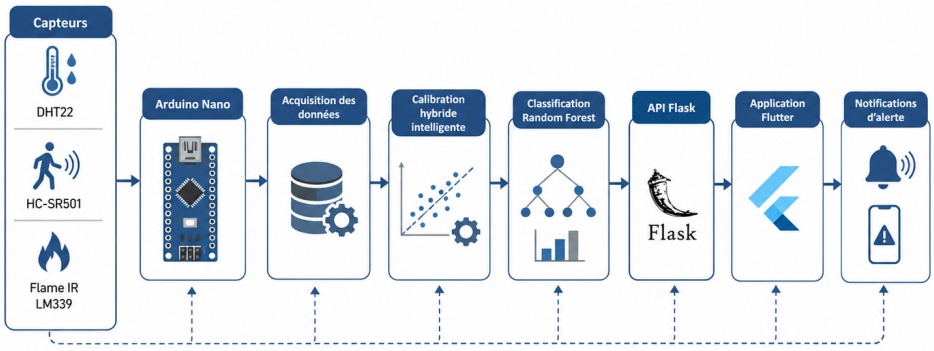


FIGURE 3.1 – Architecture globale du système de surveillance proposé.

La figure 3.1 présente les principaux blocs du système. Le premier bloc regroupe les capteurs. Le DHT22 fournit la température et l'humidité, le HC-SR501 fournit une information de présence ou de mouvement, et le Flame IR LM339 fournit une information numérique et analogique liée au risque de flamme. Ces capteurs ont été choisis pour leur simplicité d'intégration et pour leur caractère non intrusif.

Le deuxième bloc correspond à l'Arduino Nano. Cette carte assure l'acquisition des signaux et l'organisation des lectures selon un format exploitable. Elle joue le rôle d'interface

entre le monde physique et le traitement informatique. Les valeurs obtenues forment les variables d'entrée du système, telles que la température, l'humidité, le mouvement, la détection numérique de flamme et la valeur analogique du capteur de flamme.

Le troisième bloc concerne la calibration hybride intelligente. Cette étape corrige les valeurs brutes avant leur utilisation par le modèle de classification. La température est corrigée par biais moyen, tandis que l'humidité et la variable *flame_analog* sont calibrées par RandomForestRegressor. Cette stratégie permet d'adapter la méthode de calibration à la nature de chaque variable.

Le quatrième bloc correspond à la classification. Le RandomForestClassifier reçoit les variables capteurs et prédit l'état de la chambre. Les classes considérées décrivent des situations telles que l'absence, la présence, une chambre chaude, une humidité élevée, un risque de flamme ou un danger lié au feu. La sortie du modèle est ensuite transmise à la partie logicielle.

Le cinquième bloc concerne l'intégration. L'API Flask permet d'exposer les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités et les alertes sous forme JSON. L'application Flutter exploite cette réponse pour afficher un tableau de bord et déclencher des notifications à l'aide de SnackBar, AlertDialog et *flutter_local_notifications*. Le système ne collecte ni image, ni voix, ni donnée biométrique ; il se limite aux mesures nécessaires à l'interprétation de l'état de la chambre.

3.2.1 Architecture fonctionnelle du système proposé

Afin de clarifier le passage des mesures vers les décisions applicatives, le système proposé peut être représenté comme une chaîne fonctionnelle composée d'entrées, d'un jeu de données structuré, de traitements intelligents et de sorties. Les entrées proviennent des capteurs utilisés dans le prototype : le DHT22 fournit *temperature* et *humidity*, le HC-SR501 fournit *motion*, tandis que le Flame IR LM339 fournit *flame_digital* et *flame_analog*.

Le jeu de données expérimental est structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Il regroupe les colonnes *time_ms*, *temperature*, *humidity*, *motion*, *flame_digital*, *flame_analog* et *state*. Ces données constituent la base utilisée pour préparer les variables d'entrée, appliquer la calibration et entraîner le modèle de classification.

Les traitements se déroulent en plusieurs étapes. Le prétraitement vérifie la cohérence des données, organise les variables explicatives et sépare les données d'apprentissage et de test. La calibration hybride intelligente combine une correction par biais moyen pour *temperature* et des modèles RandomForestRegressor pour *humidity* et *flame_analog*. Ensuite, un modèle RandomForestClassifier prédit la classe *state* correspondant à l'état de

la chambre.

Les sorties du système regroupent les valeurs calibrées, l'état prédit, les probabilités associées à chaque classe, le niveau d'alerte et le message d'alerte. Ces informations sont transmises à l'application Flutter afin d'assurer l'affichage des mesures, la visualisation de l'état prédit et le déclenchement d'une notification lorsque le niveau d'alerte le nécessite.

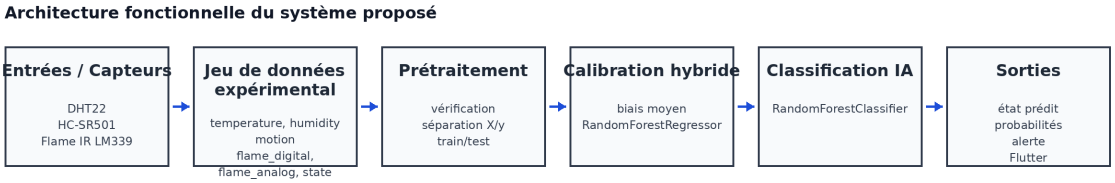


FIGURE 3.2 – Architecture fonctionnelle du système proposé : entrées, traitement et sorties.

3.3 Choix des capteurs et justification

Le choix des composants matériels constitue une étape importante dans la conception d'une Smart Room. Les capteurs doivent être compatibles avec un prototype embarqué, fournir des données pertinentes pour les scénarios étudiés et rester cohérents avec l'objectif de non-intrusion. Les composants retenus permettent de surveiller les dimensions principales de l'état de la chambre : environnement thermique, humidité, présence ou mouvement, et risque de flamme.

3.3.1 Arduino Nano

L'Arduino Nano est utilisé comme unité principale d'acquisition. Son rôle consiste à recevoir les signaux des capteurs, à les lire périodiquement et à les organiser dans un format exploitable. Cette carte est compacte, économique et adaptée aux prototypes IoT. Elle est également compatible avec l'environnement Arduino IDE, ce qui facilite le développement, le câblage et la validation des lectures.

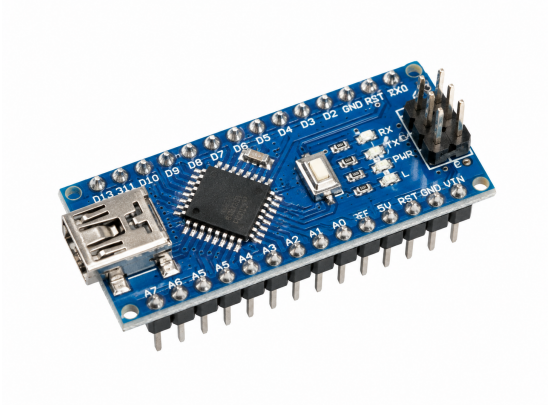


FIGURE 3.3 – Carte Arduino Nano utilisée comme unité principale d'acquisition.

Dans le système proposé, l'Arduino Nano sert d'interface entre les capteurs et la chaîne de traitement. Il ne réalise pas l'ensemble de l'analyse intelligente, mais il fournit la base des mesures. Cette séparation permet de garder la partie embarquée simple et de confier la calibration et la classification à une couche logicielle plus flexible.

3.3.2 DHT22

Le DHT22 est utilisé pour mesurer la température et l'humidité de la chambre. Ces deux grandeurs sont essentielles dans une surveillance environnementale, car elles renseignent sur le confort et sur certaines situations anormales. Une température élevée peut indiquer une chambre chaude, tandis qu'une humidité excessive peut signaler une condition défavorable.

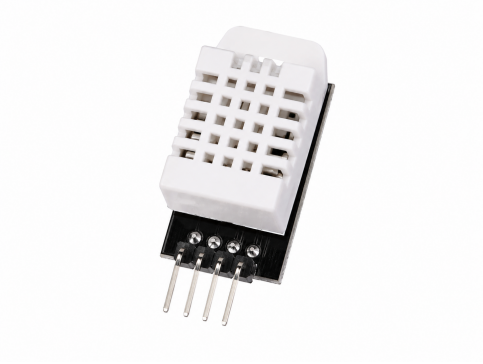


FIGURE 3.4 – Capteur DHT22 utilisé pour mesurer la température et l'humidité de la chambre.

Le DHT22 est simple à intégrer et fournit directement des valeurs numériques. Cependant, il peut présenter du bruit, un délai de réponse ou une sensibilité au positionnement. Ces limites justifient l'utilisation d'une étape de calibration, en particulier lorsque les mesures doivent être utilisées par un modèle de décision.

3.3.3 HC-SR501

Le capteur PIR HC-SR501 est utilisé pour détecter la présence ou le mouvement. Il repose sur la détection de variations infrarouges et fournit une information binaire exploitable dans les scénarios de surveillance. Son principal intérêt est de permettre une détection non visuelle, sans enregistrer d'image de la chambre.

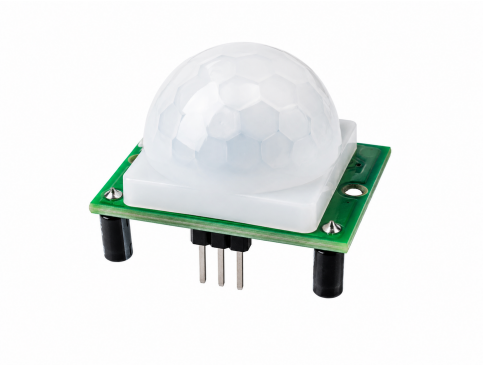


FIGURE 3.5 – Capteur PIR HC-SR501 utilisé pour détecter la présence et le mouvement.

Le HC-SR501 est adapté à une solution non intrusive. Il peut néanmoins être influencé par l'angle de détection, la distance, la sensibilité et certains mouvements parasites. Pour cette raison, son information est prise en compte dans la classification globale plutôt qu'interprétée isolément.

3.3.4 Flame IR LM339

Le capteur Flame IR LM339 est utilisé pour détecter un risque de flamme. Il fournit une sortie numérique, indiquant une détection selon un seuil, et une sortie analogique, représentant une intensité du signal. Cette double information est utile pour différencier une simple lecture ponctuelle d'un signal plus significatif.

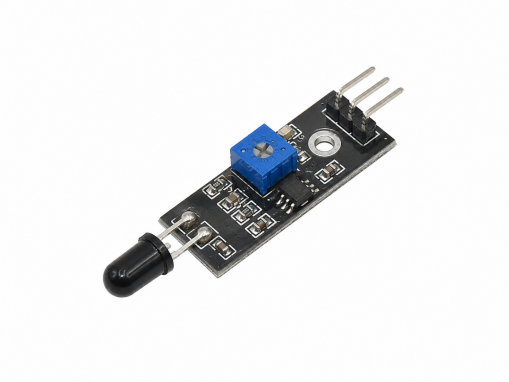


FIGURE 3.6 – Capteur Flame IR LM339 utilisé pour détecter un risque de flamme.

Ce capteur est pertinent pour un prototype de surveillance car il ajoute une dimension de

sécurité. Ses limites concernent notamment la sensibilité à la lumière infrarouge externe, à la distance, à l'angle et au réglage du seuil. La valeur analogique *flame_analog* est donc calibrée par RandomForestRegressor avant d'être utilisée dans la classification.

3.3.5 Prototype matériel

Le prototype matériel regroupe l'Arduino Nano, le DHT22, le HC-SR501 et le Flame IR LM339. L'ensemble est câblé de manière à fournir les signaux nécessaires à la surveillance de la chambre. Le DHT22 transmet ses données vers la broche D2. Le HC-SR501 transmet sa sortie vers D3. Le capteur Flame IR LM339 utilise une sortie numérique connectée à D4 et une sortie analogique connectée à A0. Les modules partagent les connexions d'alimentation VCC vers 5V et GND vers GND.

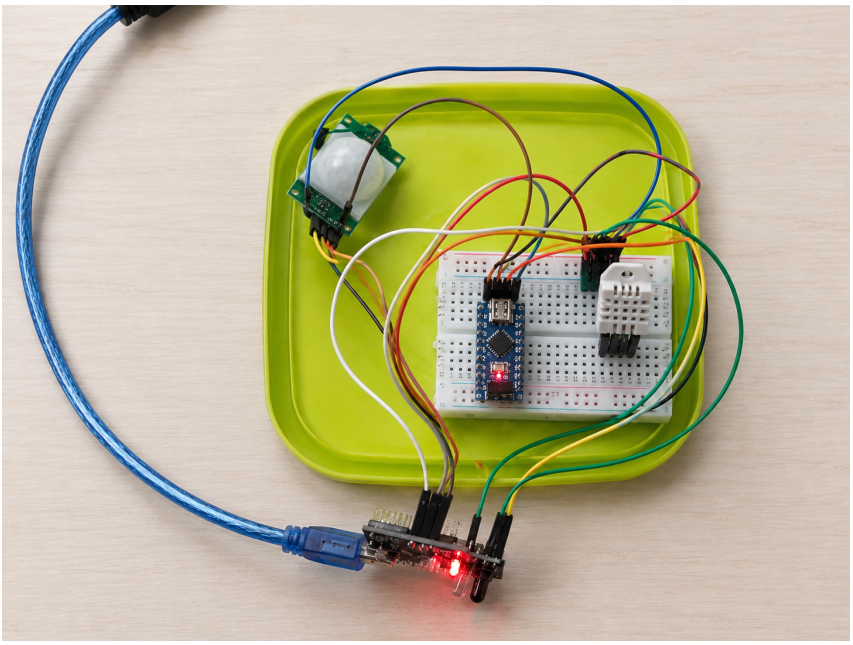


FIGURE 3.7 – Prototype matériel du système proposé basé sur Arduino Nano et capteurs.

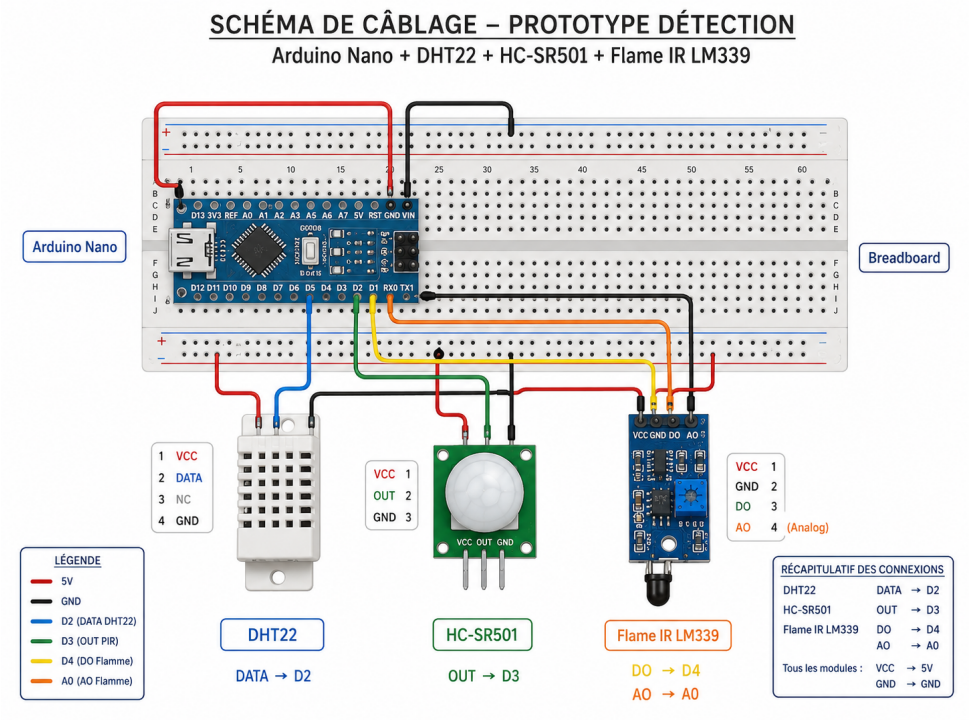


FIGURE 3.8 – Schéma de câblage du prototype Arduino Nano avec les capteurs utilisés.

Le schéma de câblage permet de visualiser la relation entre les capteurs et la carte Arduino Nano. Il facilite également la reproductibilité du montage et clarifie le rôle de chaque broche dans l'acquisition des données.

TABLEAU 3.1 – Matériel utilisé dans le système proposé

Composant	**Rôle dans le système**	**Données fournies**	**Justification**
Arduino Nano	Unité d'acquisition	Signaux capteurs	Compact et adapté aux prototypes
DHT22	Température et humidité	temperature, humidity	Surveillance environnementale
HC-SR501	Présence ou mouvement	motion	Surveillance non intrusive
Flame IR LM339	Détection de flamme	flame_digital, flame_analog	Prévention du risque de feu
PC / Python	Traitement IA	Calibration et classification	Entraînement et exécution des modèles
API Flask	Communication	JSON	Liaison entre IA et Flutter
Flutter	Interface mobile	Affichage et notifications	Interaction utilisateur

3.4 Collecte et prétraitement des données

Les données utilisées dans cette étude sont organisées sous forme d'un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Cette organisation permet de conserver une cohérence entre les lectures capteurs, les variables utilisées par les modèles d'intelligence artificielle et les états de la chambre à prédire.

Chaque observation du dataset correspond à une lecture structurée des capteurs. Les variables principales sont le temps de lecture, la température, l'humidité, la présence ou le mouvement, la sortie numérique du capteur de flamme, la valeur analogique du capteur de flamme et l'état associé à la chambre. Cette structuration permet de passer d'une suite de mesures brutes à un tableau exploitable par des algorithmes de calibration et de classification.

Tableau 3.2 – Structure du dataset expérimental

Variable	**Type**	**Description**	**Utilisation**
time_ms	Numérique	Temps de lecture en millisecondes	Suivi temporel des acquisitions
temperature	Numérique	Température mesurée dans la chambre	Calibration et classification
humidity	Numérique	Humidité mesurée dans la chambre	Calibration et classification
motion	Binaire	Présence ou mouvement, 0 ou 1	Classification de l'état de la chambre
flame_digital	Binaire	Détection numérique de flamme, 0 ou 1	Indicateur de risque
flame_analog	Numérique	Intensité analogique du capteur de flamme	Calibration et classification
state	Catégoriel	État de la chambre	Classe cible du modèle

Les états considérés correspondent aux scénarios de surveillance étudiés. Ils ne décrivent pas l'identité de la personne, mais l'état global de la chambre. Les classes utilisées sont *no_presence*, *presence*, *hot_room*, *humid_room*, *fire_risk* et *danger_fire*. Elles permettent de représenter des situations normales ou anormales à partir des variables capteurs.

Tableau 3.3 – États considérés pour la classification de la chambre

État	**Signification**	**Exemple de situation**
no_presence	Aucune présence détectée	Chambre sans mouvement
presence	Présence détectée	Mouvement détecté sans risque
hot_room	Température élevée	Température au-dessus du seuil
humid_room	Humidité élevée	Humidité anormale
fire_risk	Risque de flamme	Flamme détectée ou signal suspect
danger_fire	Danger feu	Flamme et conditions critiques

Le prétraitement consiste d'abord à organiser les colonnes, vérifier la cohérence des valeurs et distinguer les variables d'entrée de la classe cible. Les variables d'entrée correspondent aux mesures capteurs, tandis que la variable *state* représente la sortie attendue pour la classification. Les valeurs brutes et les valeurs calibrées sont préparées séparément afin de clarifier le rôle de l'étalonnage dans la chaîne de traitement.

Une séparation entre données d'entraînement et données de test est ensuite prévue pour les modèles d'intelligence artificielle. Cette séparation permet d'évaluer les modèles sur des observations non utilisées directement pendant l'apprentissage. La normalisation n'est pas obligatoire pour Random Forest, car ce type de modèle est moins sensible à l'échelle des variables que certains algorithmes comme SVM ou KNN. En revanche, une structure tabulaire claire et cohérente reste indispensable.

3.5 Méthodologie d'étalonnage des capteurs

La méthodologie d'étalonnage proposée repose sur une calibration hybride intelligente. L'objectif est de corriger les valeurs brutes avant leur utilisation par le modèle de classification. Cette étape permet de réduire l'effet des erreurs de mesure et d'améliorer la cohérence des données exploitées par le système.

La température est corrigée par une méthode de biais moyen. Cette correction est adaptée lorsque l'erreur observée reste relativement stable. Le biais est calculé à partir de l'écart moyen entre la température brute et une valeur de référence :

$$biais_{\text{temperature}} = \frac{1}{n} \sum_{i=1}^{n} (T_{\text{brute},i} - T_{\text{reference},i})$$

La température calibrée est ensuite obtenue par :

$$T_{\text{calibree}} = T_{\text{brute}} - biais_{\text{temperature}}$$

Pour l'humidité et la valeur analogique du capteur Flame IR LM339, la calibration repose sur RandomForestRegressor. Ce choix est justifié par la possibilité de relations plus variables ou non linéaires. L'humidité peut être influencée par des variations locales, tandis que le signal analogique de flamme peut dépendre de la distance, de l'angle, de la lumière ambiante et du seuil. L'utilisation d'un modèle de régression permet donc d'apprendre une correction plus flexible.

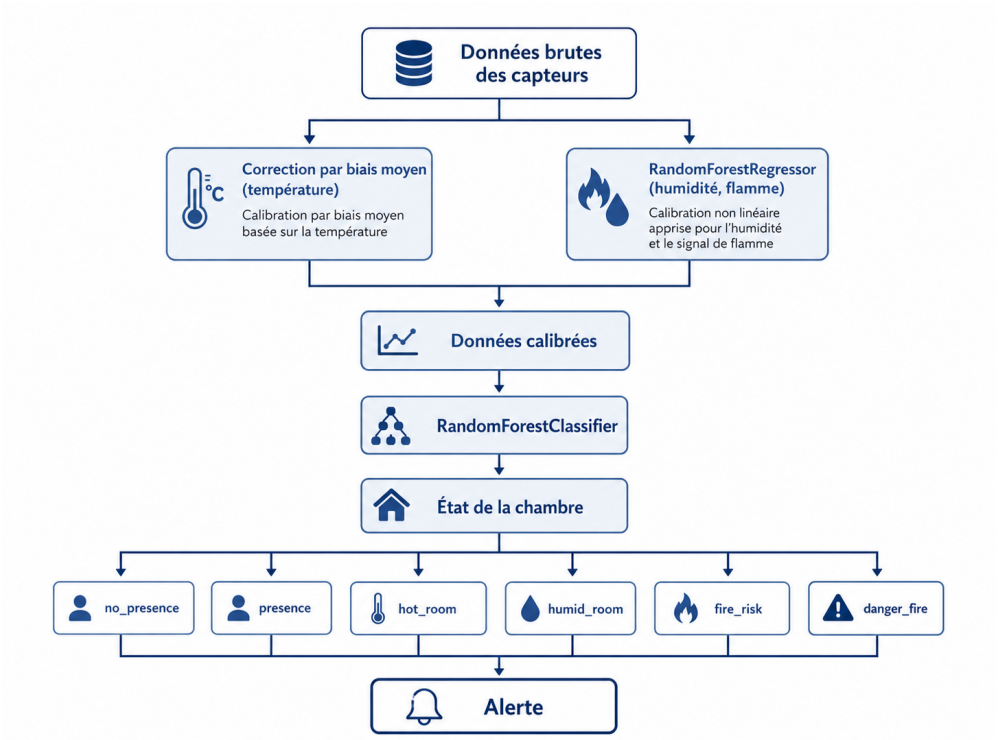


FIGURE 3.9 – Pipeline de calibration hybride et de classification de l'état de la chambre.

La figure 3.9 synthétise la démarche adoptée. Les données brutes des capteurs sont d'abord corrigées, puis les données calibrées alimentent le modèle de classification. L'approche hybride évite d'appliquer un modèle complexe à toutes les variables. Elle associe une méthode statistique simple lorsque cela est suffisant et un modèle d'apprentissage automatique lorsque la relation est plus difficile à représenter.

TABLEAU 3.4 – Méthodologie de calibration hybride proposée

Variable	**Méthode de calibration**	**Justification**	**Sortie obtenue**
temperature	Correction par biais moyen	Erreur relativement stable	temperature_calibrated
humidity	RandomForestRegressor	Relation plus variable	humidity_calibrated
flame_analog	RandomForestRegressor	Signal influencé par distance, angle et seuil	flame_analog_calibrated

3.6 Modèles d'intelligence artificielle utilisés

Les modèles d'intelligence artificielle utilisés dans le système ont deux fonctions principales : corriger certaines mesures et classifier l'état de la chambre. Cette séparation permet de distinguer clairement la calibration, qui agit sur la qualité des valeurs, et la classification, qui interprète ces valeurs pour produire un état global.

3.6.1 RandomForestRegressor pour l'étalonnage

RandomForestRegressor est utilisé pour calibrer l'humidité et la variable *flame_analog*. Les entrées du modèle peuvent inclure les valeurs brutes *temperature_raw*, *humidity_raw*, *motion*, *flame_digital* et *flame_analog_raw*. Les sorties attendues sont *humidity_calibrated* et *flame_analog_calibrated*.

L'objectif de ce modèle est de produire des valeurs corrigées avant la classification. Il ne prédit pas directement l'état de la chambre ; il améliore la qualité de certaines variables utilisées par la suite. Cette distinction est importante, car elle permet de construire une chaîne de traitement progressive : acquisition, calibration, classification, puis affichage.

3.6.2 RandomForestClassifier pour la classification

RandomForestClassifier est utilisé pour prédire l'état global de la chambre. Ses entrées correspondent aux valeurs capteurs, notamment la température, l'humidité, le mouvement, la détection numérique de flamme et la valeur analogique de flamme. Sa sortie est la variable *state*, qui prend l'une des classes définies dans le dataset.

Les classes considérées sont *no_presence*, *presence*, *hot_room*, *humid_room*, *fire_risk* et *danger_fire*. Le rôle du modèle est de combiner les informations disponibles afin de prédire l'état le plus cohérent de la chambre. Cette approche est plus souple qu'une règle unique, car elle prend en compte plusieurs variables simultanément.

3.6.3 Intégration API Flask et application Flutter

L'intégration logicielle permet de rendre les résultats exploitables par l'utilisateur. L'API Flask expose les valeurs et les prédictions sous forme d'une réponse JSON structurée. Cette réponse peut contenir les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités associées aux classes et le niveau d'alerte.

L'application Flutter récupère ces informations et les affiche dans un tableau de bord. Elle présente les mesures capteurs, les valeurs calibrées, l'état prédit et les alertes. Les notifications sont gérées par SnackBar, AlertDialog et *flutter_local_notifications*. Cette intégration permet de relier les modèles d'intelligence artificielle à une interface utilisable.

Réponse JSON de l'API Flask (/sample)


```
{
   "alert_level": "critical",
   "alert_message": "Danger : risque de feu detecte dans la chambre.",
   "calibrated_values": {
      "flame_analog_calibrated": 277,
      "humidity_calibrated": 39.0,
      "temperature_calibrated": 32.3
   },
   "predicted_state": "danger_fire",
   "probabilities": {
      "danger_fire": 0.7,
      "fire_risk": 0.3,
      "hot_room": 0.0,
      "humid_room": 0.0,
      "no_presence": 0.0,
      "presence": 0.0
   },
   "raw_values": {
      "flame_analog_raw": 260,
      "flame_digital": 0,
      "humidity_raw": 45.8,
      "motion": 1,
      "temperature_raw": 32.4
   }
}
```


Figure 1 – Exemple de réponse JSON renvoyée par l'API Flask (/sample).

FIGURE 3.10 – Exemple de réponse JSON renvoyée par l'API Flask.

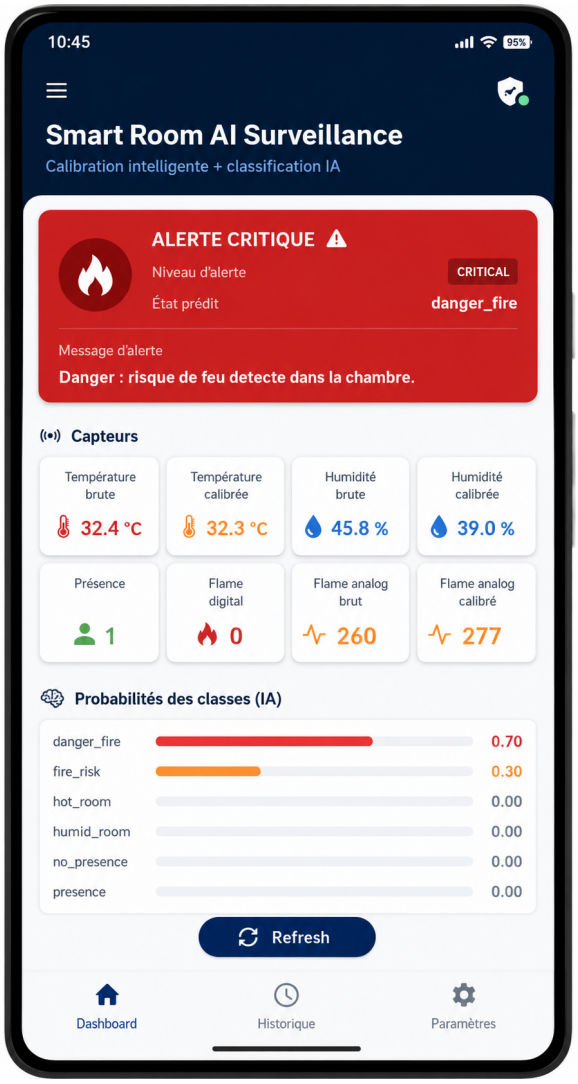


FIGURE 3.11 – Dashboard Flutter affichant les valeurs capteurs, l'état prédit et les probabilités.

Ces deux figures sont présentées comme un aperçu d'intégration. Les résultats expérimentaux détaillés et leur discussion seront traités dans le chapitre suivant.

3.6.4 Pseudo-algorithmes du système proposé

Les étapes du système proposé peuvent être représentées sous forme de pseudo-code afin de clarifier le passage des entrées capteurs vers les sorties applicatives. Cette représentation met en évidence le rôle de l'acquisition, de la structuration du jeu de données, de la calibration hybride, de la classification et de la génération des alertes.

Algorithme 1 : Acquisition et structuration des données

Début

Initialiser les capteurs DHT22, HC-SR501 et Flame IR LM339

Lire *temperature* et *humidity* depuis DHT22

Lire *motion* depuis HC-SR501

Lire *flame_digital* et *flame_analog* depuis Flame IR LM339

Construire une observation :

[*time_ms*, *temperature*, *humidity*, *motion*, *flame_digital*, *flame_analog*]

Ajouter l'observation au dataset expérimental

Fin

Algorithme 2 : Calibration hybride intelligente

Début

Charger les valeurs brutes des capteurs

Calculer *temperature_calibrated* par correction du biais moyen

Appliquer RandomForestRegressor pour prédire *humidity_calibrated*

Appliquer RandomForestRegressor pour prédire *flame_analog_calibrated*

Construire les valeurs calibrées :

[*temperature_calibrated*, *humidity_calibrated*, *flame_analog_calibrated*]

Retourner les valeurs calibrées

Fin

Algorithme 3 : Classification de l'état de la chambre

Début

Charger les variables d'entrée :

temperature, *humidity*, *motion*, *flame_digital*, *flame_analog*

Appliquer le modèle RandomForestClassifier

Prédire l'état de la chambre :

no_presence, *presence*, *hot_room*, *humid_room*, *fire_risk* ou *danger_fire*

Calculer les probabilités de chaque classe

Retourner *predicted_state* et *probabilities*

Fin

Algorithme 4 : Génération de l'alerte et affichage Flutter

Début

Recevoir *predicted_state* et *probabilities*

Si *predicted_state* = *danger_fire* **alors**

alert_level ← *critical*

alert_message ← "Danger : risque de feu détecté dans la chambre."

Sinon si *predicted_state* = *fire_risk* **alors**

alert_level ← *warning*

alert_message ← "Attention : risque de flamme détecté."

Sinon

alert_level ← *normal*

alert_message ← "État normal de la chambre."

Fin si

Construire la réponse JSON :

raw_values, *calibrated_values*, *predicted_state*, *probabilities*, *alert_level*, *alert_message*

Envoyer la réponse à l'application Flutter

Afficher les valeurs, l'état prédit et les probabilités

Déclencher SnackBar, AlertDialog ou notification locale selon *alert_level*

Fin

Ces pseudo-algorithmes montrent que le système proposé suit une chaîne logique complète allant des entrées capteurs jusqu'aux sorties applicatives. Ils complètent l'architecture globale en détaillant le rôle des données, des modèles d'étalonnage, du classificateur et du module d'alerte.

Tableau 3.5 – Modèles et composants logiciels utilisés

Composant	**Rôle**	**Entrées**	**Sorties**
RandomForestRegressor	Calibration	Valeurs brutes	Valeurs calibrées
RandomForestClassifier	Classification	Valeurs capteurs	state
Flask API	Communication	Requête JSON	Réponse JSON
Flutter	Visualisation	Réponse API	Dashboard et notifications

3.7 Conclusion

Ce chapitre a présenté la conception et la mise en œuvre du système proposé. L'architecture générale repose sur une chaîne complète allant des capteurs à l'application mobile. Les capteurs DHT22, HC-SR501 et Flame IR LM339 permettent de suivre l'état de la chambre à travers la température, l'humidité, la présence ou le mouvement et le risque de flamme. L'Arduino Nano assure l'acquisition des signaux et fournit la base des données utilisées par la suite.

Les données sont organisées sous forme d'un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Le prétraitement permet de préparer les variables d'entrée, les valeurs calibrées et la classe cible. La calibration hybride associe une correction par biais moyen pour la température et RandomForestRegressor pour l'humidité et *flame_analog*. Le RandomForestClassifier est ensuite utilisé pour prédire l'état de la chambre.

L'intégration logicielle est assurée par une API Flask et une application Flutter avec notifications. Cette organisation permet de transformer les mesures capteurs en informations exploitables par l'utilisateur. Le chapitre suivant présentera les expérimentations, les résultats obtenus et la discussion autour de la calibration, de la classification et de l'intégration du système.

Chapitre 4

Expérimentation, résultats et discussion

4.1 Introduction

Les chapitres précédents ont présenté l'état de l'art des Smart Rooms, les principes de l'étalonnage des capteurs, les modèles d'intelligence artificielle retenus, ainsi que la conception du système proposé. Le présent chapitre complète cette progression en présentant les expérimentations réalisées, les résultats obtenus et leur discussion. Il s'agit d'évaluer la chaîne complète du système, depuis le dataset expérimental jusqu'à l'affichage mobile, en passant par la calibration hybride, la classification Random Forest et l'intégration API Flask avec Flutter.

Le système étudié est une Smart Room non intrusive destinée à la surveillance de l'état d'une chambre dans des cas d'usage liés aux enfants et aux personnes âgées. Il ne cherche pas à identifier les personnes ; il exploite uniquement des mesures environnementales et événementielles : température, humidité, présence ou mouvement, risque de flamme et alertes. Les expérimentations visent donc à vérifier si ces données permettent de distinguer différents états de la chambre et de générer une réponse logicielle exploitable.

Ce chapitre présente d'abord l'environnement expérimental matériel et logiciel, puis les scénarios de test considérés. Il analyse ensuite les performances de la calibration hybride et du modèle RandomForestClassifier. Une comparaison avec deux datasets Kaggle est proposée afin de situer certaines tendances relatives à la présence et au risque de feu. Enfin, l'intégration Flask et Flutter est discutée, avant de présenter les limites du système et les perspectives d'amélioration.

4.2 Description de l'environnement expérimental

L'environnement expérimental regroupe les composants matériels, les outils logiciels et les jeux de données nécessaires à l'évaluation du système. La partie matérielle repose sur Arduino Nano, DHT22, HC-SR501 et Flame IR LM339. La partie logicielle repose sur Python et plusieurs bibliothèques utilisées pour le traitement des données, l'apprentissage automatique, la sauvegarde des modèles, la visualisation et l'intégration applicative.

Les données utilisées dans cette étude sont organisées sous forme d'un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Ce dataset contient 300 lignes réparties en six classes : *presence*, *no_presence*, *hot_room*, *humid_room*, *danger_fire* et *fire_risk*. Cette répartition permet d'entraîner et d'évaluer un modèle de classification sur plusieurs situations représentatives de l'état de la chambre.

Les traitements sont organisés sous Python. Pandas et NumPy sont utilisés pour manipuler les données tabulaires. Scikit-learn permet d'entraîner les modèles RandomForestRegressor et RandomForestClassifier. Joblib sert à sauvegarder les modèles entraînés. Matplotlib est utilisé pour produire les figures d'analyse, telles que la distribution des classes, la matrice de confusion et les comparaisons de calibration. Flask et Flask-CORS permettent d'exposer les résultats vers l'application Flutter.

TABLEAU 4.1 – Environnement expérimental

Élément	**Description**	**Rôle dans l'expérimentation**
Arduino Nano	Carte de développement compacte	Acquisition des signaux capteurs
DHT22	Capteur de température et d'humidité	Mesure des conditions environnementales
HC-SR501	Capteur PIR de présence ou mouvement	Détection non visuelle de mouvement
Flame IR LM339	Capteur de risque de flamme	Détection digitale et analogique du risque de flamme
Python / Scikit-learn	Environnement de traitement et d'apprentissage	Calibration, classification et évaluation
Flask	Framework API web	Exposition des résultats sous forme JSON
Flutter	Framework d'application mobile	Affichage des mesures, état prédit et notifications
Datasets Kaggle	Room Occupancy Estimation et Manual Fire Dataset	Comparaison externe de certaines tendances

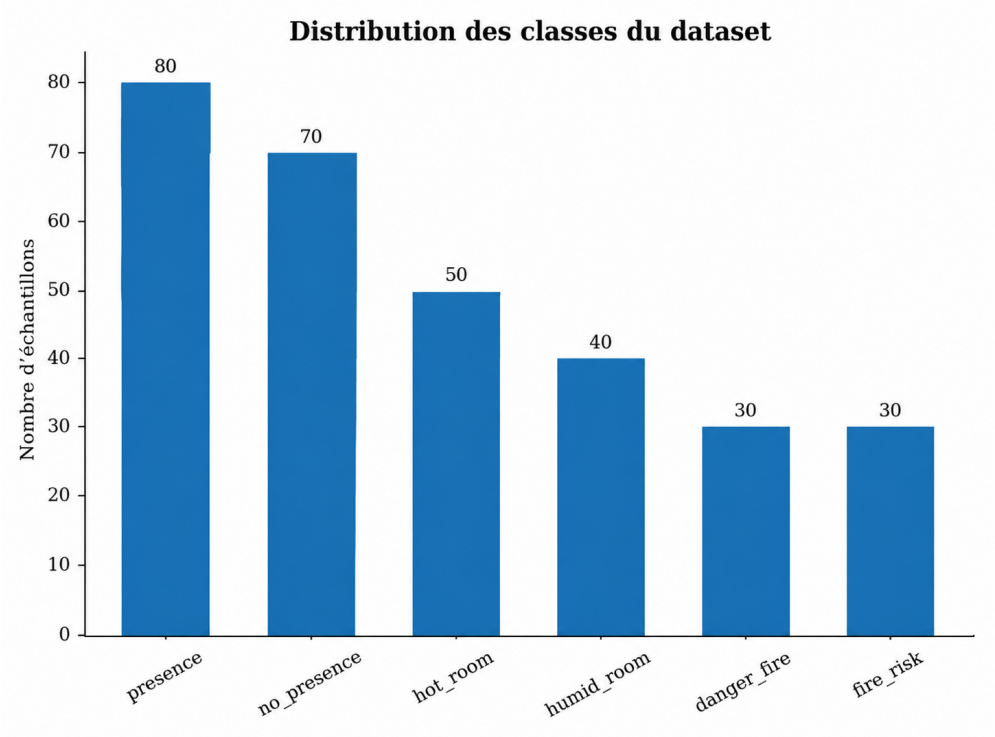


FIGURE 4.1 – Distribution des classes du dataset expérimental.

La figure 4.1 illustre la répartition des 300 observations. La classe *presence* contient 80 observations, *no_presence* en contient 70, *hot_room* en contient 50, *humid_room* en contient 40, tandis que *danger_fire* et *fire_risk* contiennent chacune 30 observations. Cette distribution couvre des situations normales et des situations d'alerte.

4.3 Scénarios de test enfants / adultes

Les scénarios enfants/adultes sont considérés comme des scénarios d'usage. Le système ne cherche pas à reconnaître biologiquement l'âge de la personne, mais à surveiller l'état de la chambre et à générer des alertes adaptées.

Les scénarios de test ont été définis pour représenter des situations simples et interprétables. Ils couvrent l'absence de mouvement, la présence, une température élevée, une humidité anormale, un risque de flamme et une situation critique liée au feu. Ces scénarios sont associés à des cas d'usage, mais les variables utilisées restent uniquement celles de l'environnement de la chambre.

Tableau 4.2 – Scénarios de test enfants / adultes

Scénario	**Cas d'usage**	**Variables observées**	**État attendu**	**Objectif**
Chambre enfant sans présence	Enfant	motion=0	no_presence	Vérifier absence
Chambre enfant avec présence	Enfant	motion=1	presence	Vérifier présence
Chambre personne âgée sans mouvement	Personne âgée	motion=0	no_presence	Suivre absence ou mouvement
Température élevée	Enfant/adulte	temperature élevée	hot_room	Alerte confort
Humidité élevée	Enfant/adulte	humidity élevée	humid_room	Alerte environnement
Risque de flamme	Enfant/adulte	flame_digital / flame_analog	fire_risk	Sécurité
Danger feu	Enfant/adulte	Flamme et conditions critiques	danger_fire	Alerte critique

Ces scénarios permettent d'évaluer la capacité du système à distinguer des états simples de la chambre. Les scénarios liés à la présence reposent principalement sur le capteur HC-SR501. Les scénarios environnementaux reposent sur le DHT22. Les scénarios de risque de feu exploitent le capteur Flame IR LM339, en combinant la sortie digitale et la valeur analogique. Le modèle de classification combine ensuite ces informations pour produire un état global.

4.4 Évaluation des performances des capteurs étalonnés

L'objectif de la calibration est d'améliorer la qualité des mesures avant leur utilisation par le modèle de classification. Dans le système proposé, la calibration est hybride : la température est corrigée par biais moyen, tandis que l'humidité et la variable *flame_analog* sont calibrées par RandomForestRegressor. Cette stratégie permet d'appliquer une correction simple lorsque l'erreur est stable, et un modèle de régression lorsque le signal est plus variable.

TABLEAU 4.3 – Résultats de calibration hybride

Variable	**Erreur avant calibration**	**Erreur après calibration**	**Amélioration**	**Méthode utilisée**
temperature	0.536	0.530	+1.24 %	Biais moyen
humidity	1.959	1.716	+12.39 %	RandomForestRegressor
flame_analog	18.033	9.313	+48.35 %	RandomForestRegressor

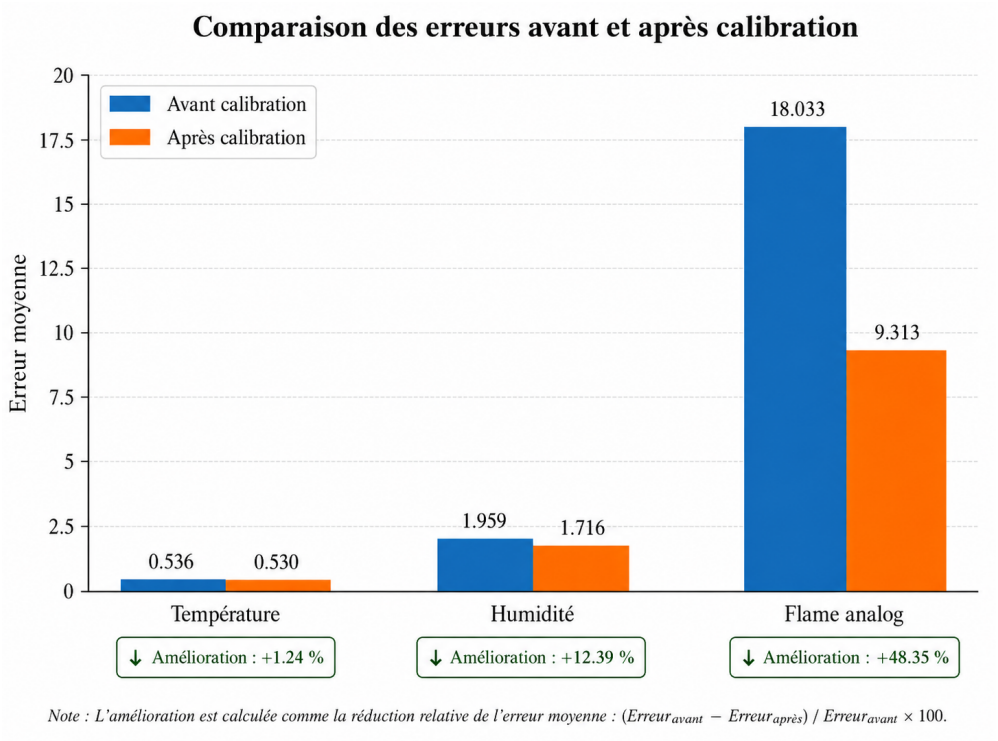


FIGURE 4.2 – Comparaison des erreurs avant et après calibration hybride.

Les résultats du tableau 4.3 et de la figure 4.2 montrent que la calibration améliore les trois variables étudiées. Pour la température, l'amélioration est faible mais positive, avec une réduction de l'erreur moyenne de 0.536 à 0.530, soit +1.24 %. Cette amélioration limitée s'explique par le fait que l'erreur initiale était déjà faible.

Pour l'humidité, l'erreur moyenne passe de 1.959 à 1.716, ce qui correspond à une amélioration de +12.39 %. Cette amélioration indique que le modèle de régression apporte une correction utile sur une variable plus sensible aux variations locales. Le résultat le plus marqué concerne *flame_analog*, dont l'erreur moyenne passe de 18.033 à 9.313, soit +48.35 %. Cette amélioration confirme l'intérêt d'utiliser RandomForestRegressor pour un signal analogique plus variable et influencé par plusieurs facteurs.

4.5 Évaluation des modèles d'intelligence artificielle

L'évaluation du modèle de classification porte sur le RandomForestClassifier entraîné à partir du dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Le modèle a été évalué sur les classes *danger_fire*, *fire_risk*, *hot_room*, *humid_room*, *no_presence* et *presence*. L'accuracy obtenue est de 0.99. L'évaluation a été réalisée sur un ensemble de test représentant 75 observations, séparées de l'ensemble d'apprentissage.

Tableau 4.4 – Résultats du RandomForestClassifier

Classe	**Precision**	**Recall**	**F1-score**	**Support**
danger_fire	0.89	1.00	0.94	8
fire_risk	1.00	0.86	0.92	7
hot_room	1.00	1.00	1.00	13
humid_room	1.00	1.00	1.00	10
no_presence	1.00	1.00	1.00	17
presence	1.00	1.00	1.00	20

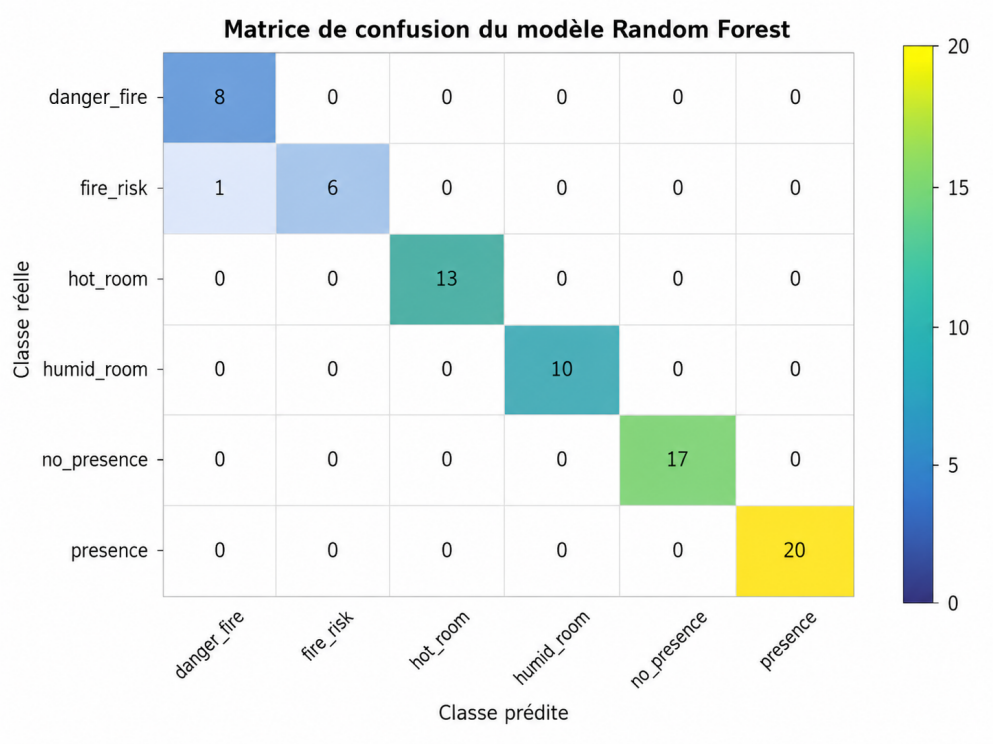


FIGURE 4.3 – Matrice de confusion du modèle Random Forest.

La matrice de confusion suit l'ordre des classes suivant : *danger_fire*, *fire_risk*, *hot_room*, *humid_room*, *no_presence* et *presence*. Elle est donnée par :

$$\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 13 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 \end{bmatrix}$$

Les résultats montrent une classification très satisfaisante. Les classes *hot_room*, *humid_room*, *no_presence* et *presence* sont correctement reconnues sur l'ensemble des observations de test. Une seule confusion apparaît entre *fire_risk* et *danger_fire*. Cette confusion reste acceptable dans un système de sécurité, car elle favorise une alerte plus prudente plutôt qu'une sous-estimation du danger.

4.6 Analyse et discussion des résultats

4.6.1 Analyse de la calibration

La calibration hybride montre que toutes les variables corrigées bénéficient d'une amélioration, mais avec des niveaux différents. La température présente une erreur initiale faible, ce qui explique une amélioration limitée. Une correction par biais moyen suffit dans ce cas, car l'objectif est de compenser un écart global plutôt qu'une relation complexe.

L'humidité bénéficie davantage de la calibration par RandomForestRegressor. Ce résultat est cohérent avec la nature de cette variable, qui peut être influencée par des variations locales et par les conditions de l'environnement. La valeur *flame_analog* présente l'amélioration la plus importante. Ce signal peut varier selon la distance, l'angle, la lumière infrarouge ambiante et le seuil du capteur. Le modèle de régression permet donc de mieux corriger ce type de signal qu'une correction linéaire simple.

4.6.2 Analyse de la classification

L'accuracy de 0.99 indique que le RandomForestClassifier parvient à distinguer efficacement les états de la chambre dans les conditions étudiées. Les performances parfaites obtenues pour plusieurs classes montrent que les variables capteurs sont suffisamment discriminantes pour les scénarios définis. La seule confusion entre *fire_risk* et *danger_fire* doit être interprétée avec prudence.

Dans un système de surveillance, une confusion orientée vers une alerte plus forte peut être considérée comme moins problématique qu'une confusion qui réduirait le niveau de danger. Le modèle classe une observation *fire_risk* comme *danger_fire*, ce qui produit une réponse plus prudente. Cette interprétation est cohérente avec l'objectif de sécurité du système.

4.6.3 Comparaison avec les datasets Kaggle

La comparaison avec les datasets Kaggle ne vise pas à affirmer une équivalence directe entre les jeux de données. Les contextes, les capteurs et les objectifs ne sont pas identiques. Cependant, cette comparaison permet de vérifier si les tendances observées restent logiques par rapport à des datasets externes liés à l'occupation de pièces et à des situations de feu.

TABLEAU 4.5 – Comparaison avec datasets Kaggle

Indicateur	Dataset expérimental	Kaggle Occupancy	Kaggle Manual Fire	Interprétation
Taux présence	55 %	18.77 %	Non applicable	Les contextes d'occupation sont différents
Taux feu / risque feu	20 %	Non applicable	46.69 %	Les objectifs des datasets ne sont pas identiques
Température moyenne	28.82 °C	25.45 °C	41.14 °C	Les températures reflètent des scénarios différents

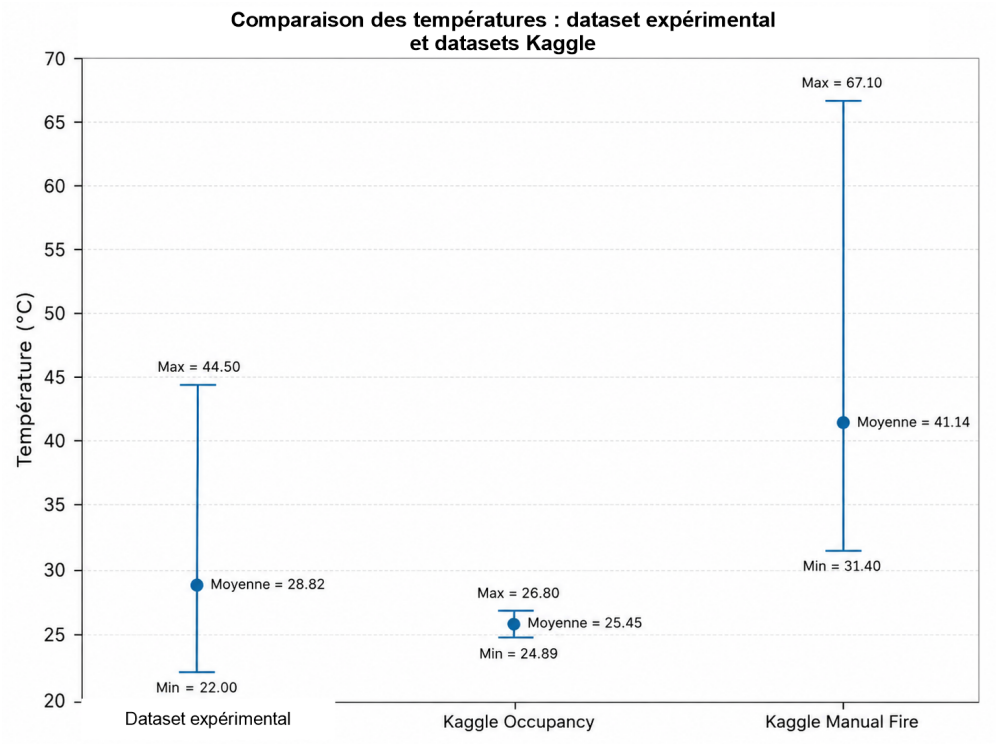


FIGURE 4.4 – Comparaison des températures entre le dataset expérimental et les datasets Kaggle.

Le dataset Room Occupancy Estimation valide l'idée de distinguer présence et absence à partir de variables environnementales ou d'occupation. Le Manual Fire Dataset valide l'intérêt de différencier des situations normales et des situations associées au feu. Les différences de taux s'expliquent par des objectifs expérimentaux distincts : le dataset expérimental cherche à couvrir plusieurs états de chambre, tandis que les datasets Kaggle répondent à des problématiques spécifiques.

4.6.4 Intégration API Flask et Flutter

L'intégration logicielle permet de transformer les résultats des modèles en informations exploitables par l'utilisateur. L'API Flask fournit une réponse JSON structurée contenant

les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités et le niveau d'alerte. Un exemple de réponse contient notamment *predicted_state* égal à *danger_fire*, *alert_level* égal à *critical*, et le message d'alerte : « Danger : risque de feu detecte dans la chambre. »

Dans cet exemple, les valeurs brutes sont : *temperature_raw*=32.4, *humidity_raw*=45.8, *motion*=1, *flame_digital*=0 et *flame_analog_raw*=260. Les valeurs calibrées sont : *temperature_calibrated*=32.3, *humidity_calibrated*=39.0 et *flame_analog_calibrated*=277. Les probabilités associées sont 0.7 pour *danger_fire*, 0.3 pour *fire_risk*, et 0.0 pour les autres classes.

Réponse JSON de l'API Flask (/sample)


```
{
   "alert_level": "critical",
   "alert_message": "Danger : risque de feu detecte dans la chambre.",
   "calibrated_values": {
      "flame_analog_calibrated": 277,
      "humidity_calibrated": 39.0,
      "temperature_calibrated": 32.3
   },
   "predicted_state": "danger_fire",
   "probabillities": {
      "danger_fire": 0.7,
      "fire_risk": 0.3,
      "hot_room": 0.0,
      "humid_room": 0.0,
      "no_presence": 0.0,
      "presence": 0.0
   },
   "raw_values": {
      "flame_analog_raw": 260,
      "flame_digital": 0,
      "humidity_raw": 45.8,
      "motion": 1,
      "temperature_raw": 32.4
   }
}
```


Figure 1 – Exemple de réponse JSON renvoyée par l'API Flask (/sample).

FIGURE 4.5 – Exemple de réponse JSON renvoyée par l'API Flask.

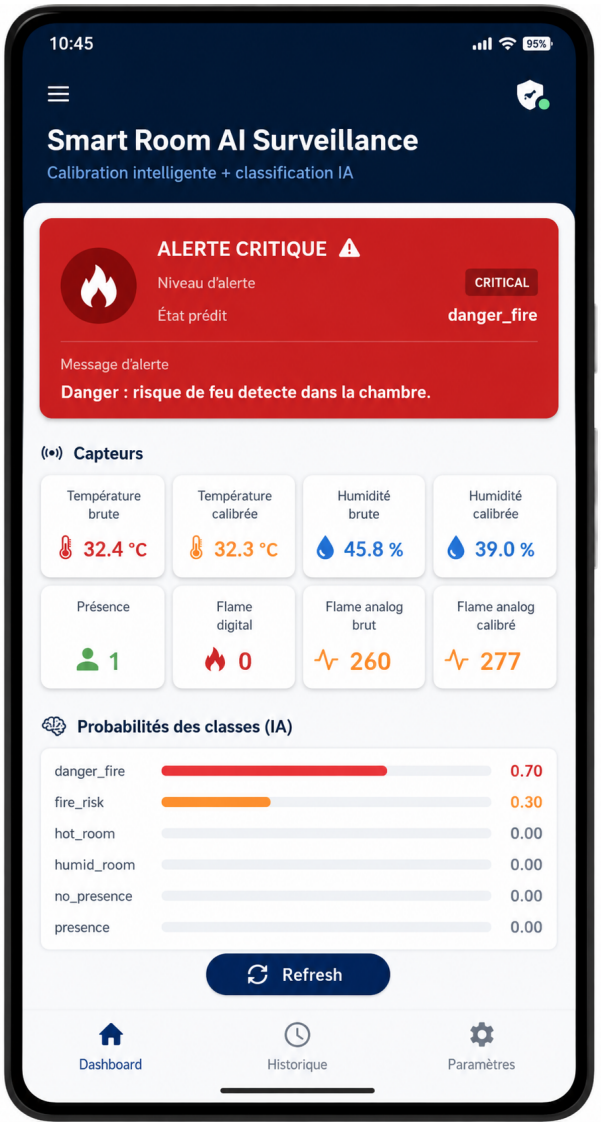


FIGURE 4.6 – Dashboard Flutter affichant les valeurs capteurs, l'état prédit et les probabilités.

4.6.5 Notifications et alerte critique

L'application Flutter affiche les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités et les messages d'alerte. Sur Chrome ou Web, les alertes sont affichées à travers SnackBar et AlertDialog. Sur Android, les notifications locales sont prévues avec *flutter_local_notifications*. Les niveaux *warning* ou *critical* déclenchent une notification, et le niveau *critical* déclenche un AlertDialog intitulé « Alerte critique ».



FIGURE 4.7 – Notification critique affichée dans l'application Flutter.



FIGURE 4.8 – Écran de test manuel de prédiction dans l'application Flutter.

TABLEAU 4.6 – Fonctionnalités API Flask et Flutter

Composant	**Fonction**	**Données affichées / retournées**	**Rôle**
Flask API	Prédiction et réponse JSON	raw_values, calibrated_values, predicted_state, probabilities, alert_level	Communication IA-interface
Flutter Dashboard	Visualisation	Valeurs, état, probabilités	Suivi utilisateur
SnackBar	Alerte visuelle	alert_message	Notification immédiate
AlertDialog	Alerte critique	Message danger	Attirer l'attention
flutter_local_notifications	Notification Android	warning / critical	Notification locale mobile

L'ensemble de cette intégration montre que les résultats de calibration et de classification peuvent être exploités dans une interface utilisateur. Le système ne se limite donc pas à produire un modèle, mais fournit une chaîne complète allant de la mesure à la notification.

4.7 Limites du système proposé

Malgré les résultats obtenus, le système proposé présente plusieurs limites. La première concerne le périmètre des scénarios étudiés. Le système se concentre sur la température, l'humidité, la présence ou le mouvement et le risque de flamme. Il ne couvre donc pas tous les événements possibles dans une chambre connectée, et son interprétation reste liée aux capteurs utilisés.

La deuxième limite concerne la nature des données. Le dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés permet d'évaluer la chaîne proposée, mais il doit être enrichi pour couvrir davantage de situations, de conditions environnementales et de variations temporelles. Une durée d'observation plus longue permettrait de mieux évaluer la stabilité du système.

La comparaison avec les datasets Kaggle reste également indirecte. Les datasets externes utilisés ne possèdent pas exactement les mêmes capteurs ni les mêmes objectifs. Ils permettent une comparaison logique, mais ne remplacent pas une validation complète dans plusieurs chambres réelles. De plus, les notifications Android doivent être validées sur un appareil réel afin de confirmer leur comportement dans des conditions d'utilisation mobile. Les notifications locales ont été préparées pour Android, mais une validation complète sur smartphone physique reste nécessaire pour confirmer leur comportement hors navigateur.

Enfin, le système privilégie une approche non intrusive. Ce choix limite volontairement la quantité de données personnelles collectées, mais réduit aussi la richesse des informations disponibles. Cette limite est assumée, car l'objectif du mémoire est de surveiller l'état de la chambre tout en respectant l'espace privé.

4.8 Conclusion et perspectives

Ce chapitre a présenté les expérimentations, les résultats obtenus et leur discussion. La calibration hybride améliore les mesures de température, d'humidité et de *flame_analog*, avec une amélioration particulièrement importante pour le signal analogique du capteur de flamme. Le RandomForestClassifier atteint une accuracy de 0.99 sur le dataset expérimental, avec une seule confusion entre *fire_risk* et *danger_fire*. Cette confusion est acceptable dans une logique de sécurité, car elle favorise une alerte plus prudente.

La comparaison avec les datasets Kaggle permet de situer certaines tendances liées à la présence et au risque de feu. Même si les datasets ne sont pas directement équivalents, ils renforcent la cohérence des scénarios étudiés. L'intégration Flask et Flutter montre que les résultats peuvent être présentés dans une interface mobile, avec affichage des valeurs, état prédit, probabilités et notifications.

Plusieurs perspectives peuvent être envisagées. Il serait utile d'enrichir le dataset expérimental avec davantage de scénarios, d'ajouter un capteur de gaz ou de fumée, de tester le système dans plusieurs chambres et sur une durée plus longue. Le déploiement sur un appareil Android réel permettrait de valider les notifications locales. L'intégration de Firebase Cloud Messaging pourrait aussi être étudiée pour des alertes à distance. Enfin, l'ajout d'un historique des mesures et l'amélioration de l'explicabilité du modèle renforceraient l'utilité du système dans un contexte de surveillance intelligente.

Conclusion générale

Ce mémoire s'inscrit dans le domaine des Smart Rooms, de l'Internet des Objets et de l'intelligence artificielle appliqués à la surveillance non intrusive des environnements domestiques. Le contexte général du travail concerne la surveillance de l'état d'une chambre dans des cas d'usage liés aux enfants et aux personnes âgées. Dans ce type d'espace privé, l'objectif n'est pas de surveiller directement l'identité ou les caractéristiques biologiques d'une personne, mais de suivre des indicateurs simples et utiles liés à l'environnement.

La chambre connectée étudiée dans ce mémoire repose sur des capteurs non intrusifs permettant de mesurer la température, l'humidité, la présence ou le mouvement, ainsi qu'un risque de flamme. Cette orientation répond à une exigence importante : proposer une solution capable de générer des alertes tout en limitant la collecte de données sensibles. Le système ne repose donc pas sur une caméra, une donnée biométrique ou une identification de la personne. Il se concentre uniquement sur l'état de la chambre et sur les alertes associées.

Dans ce contexte, les capteurs à faible coût présentent un intérêt pratique important, car ils permettent de concevoir un prototype accessible et reproductible. Toutefois, ces capteurs peuvent produire des mesures imparfaites ou variables. L'étalonnage devient alors une étape essentielle pour améliorer la qualité des données avant leur exploitation par un modèle de classification. C'est dans cette perspective que ce mémoire a proposé une approche combinant acquisition par capteurs, calibration hybride intelligente, classification Random Forest, API Flask et application Flutter.

La problématique principale de ce mémoire était de concevoir une Smart Room non intrusive capable de surveiller l'état d'une chambre à partir de capteurs simples, tout en améliorant la fiabilité des mesures et en produisant des alertes exploitables. Cette problématique implique plusieurs défis. Le premier concerne le choix des capteurs : ils doivent être suffisamment pertinents pour détecter des états de la chambre, mais rester non intrusifs. Le deuxième concerne la qualité des mesures, car des capteurs à faible coût peuvent être influencés par le bruit, les variations environnementales ou les limites propres au composant. Le troisième concerne l'interprétation des données, qui nécessite de passer

de valeurs capteurs brutes à un état global compréhensible.

La problématique peut ainsi être résumée par la question suivante : comment concevoir un système de chambre connectée qui surveille des variables environnementales et événementielles, corrige intelligemment certaines mesures et classe automatiquement l'état de la chambre, tout en préservant la simplicité matérielle et la non-intrusion ? Cette question a guidé l'ensemble du travail, depuis l'état de l'art jusqu'aux expérimentations.

La solution proposée repose sur une architecture complète de Smart Room. La partie matérielle utilise une carte Arduino Nano comme unité principale d'acquisition, un capteur DHT22 pour la température et l'humidité, un capteur HC-SR501 pour la présence ou le mouvement, et un capteur Flame IR LM339 pour la détection d'un risque de flamme. Ces composants ont été choisis pour leur simplicité, leur coût réduit, leur compatibilité avec un prototype embarqué et leur caractère non intrusif.

Les données utilisées dans le mémoire sont organisées sous forme d'un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Ce dataset permet de représenter plusieurs états de la chambre : *no_presence*, *presence*, *hot_room*, *humid_room*, *fire_risk* et *danger_fire*. Ces classes ne représentent pas des catégories biologiques de personnes ; elles décrivent uniquement l'état de l'environnement surveillé.

La solution intègre une calibration hybride intelligente. La température est corrigée par une méthode de biais moyen, adaptée lorsque l'erreur est relativement stable. L'humidité et la variable *flame_analog* sont calibrées par RandomForestRegressor, afin de mieux gérer des relations plus variables ou non linéaires. Après cette étape de correction, un RandomForestClassifier est utilisé pour classifier l'état global de la chambre.

L'intégration logicielle complète la chaîne de traitement. Une API Flask fournit les résultats sous forme de réponse JSON contenant les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités, le niveau d'alerte et le message d'alerte. Une application Flutter affiche ces informations sous forme de tableau de bord et permet de déclencher des notifications à travers SnackBar, AlertDialog et *flutter_local_notifications* pour Android.

Les résultats obtenus montrent d'abord l'intérêt de la calibration hybride. Pour la température, l'erreur moyenne passe de 0.536 avant calibration à 0.530 après calibration, soit une amélioration de +1.24 %. Cette amélioration est faible mais positive, ce qui s'explique par le fait que l'erreur initiale était déjà limitée. Pour l'humidité, l'erreur moyenne passe de 1.959 à 1.716, soit une amélioration de +12.39 %. Cette réduction montre que RandomForestRegressor apporte une correction utile sur une variable plus sensible aux variations environnementales.

Le résultat le plus marqué concerne *flame_analog*. L'erreur moyenne passe de 18.033 avant calibration à 9.313 après calibration, soit une amélioration de +48.35 %. Ce résultat confirme l'intérêt d'utiliser un modèle de régression pour un signal analogique plus variable, influencé notamment par la distance, l'angle, le seuil et les conditions lumineuses. La calibration hybride apparaît donc comme une solution pertinente, car elle évite d'appliquer une méthode unique à toutes les variables.

La classification par RandomForestClassifier a également donné des résultats satisfaisants. Le modèle atteint une accuracy de 0.99 sur un ensemble de test de 75 observations. Les classes considérées sont *no_presence*, *presence*, *hot_room*, *humid_room*, *fire_risk* et *danger_fire*. Une confusion a été observée entre *fire_risk* et *danger_fire*. Cette confusion reste acceptable dans une logique de sécurité, car elle favorise une alerte plus prudente plutôt qu'une sous-estimation du risque.

La comparaison avec les datasets Kaggle a permis de situer certaines tendances par rapport à des données externes. Le dataset Room Occupancy Estimation a été utilisé pour comparer la logique présence / absence, tandis que le Manual Fire Dataset a été utilisé pour comparer la logique liée au risque de feu. Le taux de présence est de 55 % dans le dataset expérimental contre 18.77 % dans Kaggle Occupancy. Le taux de risque feu est de 20 % dans le dataset expérimental, tandis que le taux de feu est de 46.69 % dans Kaggle Manual Fire. Les températures moyennes sont de 28.82 °C pour le dataset expérimental, 25.45 °C pour Kaggle Occupancy et 41.14 °C pour Kaggle Manual Fire.

Ces datasets externes ne sont pas directement équivalents au dataset expérimental, car leurs objectifs, leurs capteurs et leurs conditions ne sont pas identiques. Ils permettent néanmoins une comparaison logique avec des données liées à l'occupation de pièces et au risque de feu. Enfin, l'API Flask et l'application Flutter ont montré que les résultats de calibration et de classification peuvent être intégrés dans une interface exploitable par l'utilisateur, avec affichage des valeurs, des probabilités, de l'état prédit et des alertes.

Malgré les résultats obtenus, le système proposé présente plusieurs limites. Les scénarios étudiés restent limités à la température, à l'humidité, à la présence ou au mouvement, et au risque de flamme. Le système ne couvre donc pas l'ensemble des situations possibles dans une chambre connectée. Il ne réalise pas de reconnaissance biométrique et ne détecte pas les chutes. Ces limites sont cohérentes avec le choix d'une approche non intrusive, mais elles réduisent le périmètre fonctionnel du système.

La comparaison avec les datasets Kaggle reste également indirecte. Les datasets externes n'utilisent pas exactement les mêmes capteurs et ne poursuivent pas les mêmes objectifs. Ils servent donc de repères comparatifs, mais ne remplacent pas une validation expérimentale dans plusieurs environnements réels. De plus, le dataset expérimental doit être

enrichi avec davantage de situations, de conditions environnementales et de variations temporelles afin d'améliorer la robustesse des modèles.

Une autre limite concerne l'intégration mobile. Les notifications Android ont été préparées, mais elles doivent être validées sur un smartphone physique afin de confirmer leur comportement hors navigateur. Enfin, le système doit être testé dans plusieurs chambres et sur une durée plus longue pour évaluer sa stabilité, sa robustesse et sa capacité à fonctionner dans des conditions d'usage variées.

Plusieurs perspectives peuvent être envisagées à partir de ce travail. La première consiste à enrichir le dataset expérimental avec davantage de scénarios. Des situations plus variées permettraient de renforcer la généralisation du modèle et d'améliorer la robustesse de la calibration. Il serait également utile de collecter des données sur une période plus longue, afin d'étudier les variations temporelles des capteurs et la stabilité des prédictions.

Une deuxième perspective concerne l'amélioration matérielle du prototype. L'ajout d'un capteur de gaz ou de fumée permettrait de renforcer la détection des situations liées au feu. Des tests dans plusieurs chambres réelles permettraient aussi de mieux évaluer l'effet du positionnement des capteurs, de la ventilation, de la lumière et des conditions environnementales.

Sur le plan logiciel, le déploiement complet de l'application sur un smartphone Android réel constitue une étape importante. L'intégration de Firebase Cloud Messaging pourrait permettre des alertes à distance, utiles lorsque l'utilisateur n'est pas à proximité du système. L'ajout d'un historique des mesures et d'un tableau de bord avec suivi temporel permettrait également d'améliorer l'analyse des tendances.

Enfin, l'explicabilité du modèle Random Forest pourrait être approfondie. Une meilleure compréhension de l'importance des variables aiderait à interpréter les décisions du modèle, à justifier les alertes et à renforcer la confiance dans le système. La calibration pourrait également être améliorée avec plus de données, notamment pour les variables sensibles aux conditions environnementales.

En conclusion, ce mémoire a permis de concevoir et d'évaluer une Smart Room non intrusive destinée à la surveillance de l'état d'une chambre dans des cas d'usage liés aux enfants et aux personnes âgées. Le système proposé combine des capteurs simples, une calibration hybride intelligente, un modèle RandomForestClassifier, une API Flask et une application Flutter avec notifications. Cette chaîne complète montre qu'il est possible de transformer des mesures capteurs en informations exploitables, tout en respectant une logique de non-intrusion.

Les résultats obtenus confirment l'intérêt de l'approche proposée. La calibration améliore

les mesures, en particulier pour le signal *flame_analog*, et le modèle de classification atteint une accuracy élevée sur l'ensemble de test. L'intégration API et Flutter démontre la faisabilité d'une interface permettant de visualiser les valeurs, l'état prédit, les probabilités et les alertes. Même si le système présente encore des limites, il constitue une base solide pour développer des solutions de surveillance intelligente plus robustes, plus riches et adaptées aux environnements domestiques sensibles.

Références bibliographiques

Bibliographie

[1] Zimmerman, N., Presto, A. A., Kumar, S. P. N., Gu, J., Hauryliuk, A., Robinson, E. S., Robinson, A. L., & Subramanian, R. (2018). A machine learning calibration model using random forests to improve sensor performance for lower-cost air quality monitoring. *Atmospheric Measurement Techniques*, 11, 291–313. https://doi.org/10.5194/amt-11-291-2018

[2] Wang, J., Spicher, N., Warnecke, J. M., Haghi, M., Schwartze, J., & Deserno, T. M. (2021). Unobtrusive health monitoring in private spaces: The smart home. *Sensors*, 21(3), 864. https://doi.org/10.3390/s21030864

[3] Abdulmalek, S., Nasir, A., Jabbar, W. A., Almuhaya, M. A. M., Bairagi, A. K., Khan, M. A.-M., & Kee, S. H. (2022). IoT-based healthcare-monitoring system towards improving quality of life: A review. *Healthcare*, 10(10), 1993. https://doi.org/10.3390/healthcare10101993

[4] Bouchabou, D., Nguyen, S. M., Lohr, C., LeDuc, B., & Kanellos, I. (2021). A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18), 6037. https://doi.org/10.3390/s21186037

[5] Susnea, I., Dumitriu, L., Talmaciu, M., Pecheanu, E., & Munteanu, D. (2019). Unobtrusive monitoring the daily activity routine of elderly people living alone, with low-cost binary sensors. *Sensors*, 19(10), 2264. https://doi.org/10.3390/s19102264

[6] Yu, J., An, N., Hassan, T., & Kong, Q. (2019). A pilot study on a smart home for elders based on continuous in-home unobtrusive monitoring technology. *HERD: Health Environments Research & Design Journal*, 12(3), 206–219. https://doi.org/10.1177/1937586719826059

[7] Campbell, J. P., Buchan, J., & Chu, C. H. (2024). User perception of smart home surveillance among adults aged 50 years and older: Scoping review. *JMIR mHealth and uHealth*, 12, e48526. https://doi.org/10.2196/48526

[8] Cook, D. J., Crandall, A. S., Thomas, B. L., & Krishnan, N. C. (2013). CASAS: A smart home in a box. *Computer*, 46(7), 62–69. https://doi.org/10.1109/MC.2012.328

[9] Chen, L., Nugent, C. D., & Wang, H. (2012). A knowledge-driven approach to activity recognition in smart homes. *IEEE Transactions on Knowledge and Data Engineering*, 24(6), 961–974. https://doi.org/10.1109/TKDE.2011.51

[10] Han, P., Mei, H., Liu, D., Zeng, N., Tang, X., Wang, Y., & Pan, Y. (2021). Calibrations of low-cost air pollution monitoring sensors for CO, NO2, O3, and SO2. *Sensors*, 21(1), 256. https://doi.org/10.3390/s21010256

[11] Dahmen, J., Cook, D. J., Wang, X., & Honglei, W. (2017). A survey of smart home technologies that sense, assess, and respond to human needs. *Sensors*, 17(9), 2096. https://doi.org/10.3390/s17092096

[12] Rodrigues, M. J., Postolache, O., & Cercas, F. (2020). Physiological and behavior monitoring systems for smart environments: A review. *Sensors*, 20(8), 2186. https://doi.org/10.3390/s20082186

[13] Morita, P. P., Sahu, K. S., & Oetomo, A. (2023). Health monitoring using smart home technologies: Scoping review. *JMIR mHealth and uHealth*, 11, e37347. https://doi.org/10.2196/37347

[14] Linkous, L., Zohrabi, N., & Abdelwahed, S. (2019). Health monitoring in smart homes utilizing Internet of Things. *arXiv preprint arXiv:1910.07058*. https://arxiv.org/abs/1910.07058

[15] Kumar, V., & Sahu, M. (2021). Evaluation of nine machine learning regression algorithms for calibration of low-cost PM2.5 sensor. *Journal of Aerosol Science*, 157, 105809. https://doi.org/10.1016/j.jaerosci.2021.105809

[16] Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. https://doi.org/10.1023/A:1010933404324

[17] Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Annexe A

Annexes techniques du système proposé

Présentation générale des annexes

Ces annexes regroupent les éléments techniques complémentaires liés au système proposé dans le mémoire intitulé *Étalonnage intelligent des capteurs pour la surveillance des enfants et des personnes âgées dans une chambre connectée*. Elles ne remplacent pas les chapitres principaux, mais elles permettent de documenter plus précisément le montage matériel, la structure des données, les pseudo-algorithmes, les modèles d'intelligence artificielle, l'API Flask et l'application Flutter.

Le système étudié repose sur une chaîne complète allant des capteurs vers l'application mobile : acquisition des mesures, structuration du dataset expérimental, prétraitement, calibration hybride intelligente, classification de l'état de la chambre, génération d'une réponse JSON et affichage dans une interface Flutter avec notifications.

A.1 Annexe A : Matériel utilisé et câblage du prototype

A.1.1 Liste du matériel

Élément	**Rôle**	**Données ou fonction fournie**
Arduino Nano	Unité principale d'acquisition	Lecture des signaux issus des capteurs

DHT22	Capteur environnemental	Température et humidité
HC-SR501	Capteur PIR de présence	Détection de présence ou de mouvement
Flame IR LM339	Capteur de flamme	Sortie numérique et valeur analogique liées au risque de flamme
PC / Python	Traitement logiciel	Prétraitement, calibration, classification et sauvegarde des modèles
Flask API	Communication	Réponse JSON vers l'application mobile
Flutter	Interface utilisateur	Affichage des mesures, état prédit, probabilités et notifications

A.1.2 Récapitulatif des connexions

Capteur	**Broche utilisée**	**Description**
DHT22	DATA → D2	Lecture de la température et de l'humidité
HC-SR501	OUT → D3	Détection de présence ou de mouvement
Flame IR LM339	DO → D4	Sortie numérique de détection de flamme
Flame IR LM339	AO → A0	Valeur analogique du signal de flamme
Tous les modules	VCC → 5V	Alimentation des capteurs
Tous les modules	GND → GND	Masse commune

A.2 Annexe B : Structure du dataset expérimental

Les données utilisées dans cette étude sont organisées sous forme d'un dataset expérimental structuré selon le format de sortie du prototype Arduino et selon les scénarios de surveillance étudiés. Chaque observation représente une lecture cohérente des capteurs et l'état correspondant de la chambre.

A.2.1 Variables principales

Variable	Type	Description
time_ms	Numérique	Temps de lecture en millisecondes
temperature	Numérique	Température de la chambre
humidity	Numérique	Humidité de la chambre
motion	Binaire	Présence ou mouvement détecté : 0 ou 1
flame_digital	Binaire	Détection numérique de flamme : 0 ou 1
flame_analog	Numérique	Intensité analogique du capteur de flamme
state	Catégoriel	État associé à la chambre

A.2.2 États considérés

État	Signification	Exemple de situation
no_presence	Aucune présence détectée	Chambre sans mouvement
presence	Présence détectée	Mouvement détecté sans risque
hot_room	Température élevée	Chambre chaude
humid_room	Humidité élevée	Humidité anormale
fire_risk	Risque de flamme	Signal de flamme suspect
danger_fire	Danger feu	Risque critique nécessitant une alerte

A.2.3 Exemple de lignes du dataset

```
time_ms,temperature,humidity,motion,flame_digital,flame_analog,state
1000,24.5,50.0,0,1,830,no_presence
2000,25.2,48.0,1,1,805,presence
3000,35.8,42.0,1,1,680,hot_room
4000,26.0,84.0,0,1,780,humid_room
5000,30.0,45.0,1,0,380,fire_risk
6000,38.0,34.0,1,0,240,danger_fire
```

Listing A.1 – Exemple simplifié de structure CSV du dataset expérimental

A.3 Annexe C : Pseudo-algorithmes du système proposé

Cette annexe reprend les pseudo-algorithmes qui décrivent le fonctionnement général du système. Ils montrent le passage des entrées capteurs vers les sorties applicatives.

A.3.1 Algorithme 1 : Acquisition et structuration des données

Début

1. Initialiser les capteurs DHT22, HC-SR501 et Flame IR LM339.
2. Lire `temperature` et `humidity` depuis le DHT22.
3. Lire `motion` depuis le HC-SR501.
4. Lire `flame_digital` et `flame_analog` depuis le Flame IR LM339.
5. Construire une observation : `[time_ms, temperature, humidity, motion, flame_digital, flame_analog]`.
6. Ajouter l'observation au dataset expérimental.

Fin

A.3.2 Algorithme 2 : Calibration hybride intelligente

Début

1. Charger les valeurs brutes des capteurs.
2. Calculer `temperature_calibrated` par correction du biais moyen.
3. Appliquer `RandomForestRegressor` pour prédire `humidity_calibrated`.
4. Appliquer `RandomForestRegressor` pour prédire `flame_analog_calibrated`.
5. Construire les valeurs calibrées : `[temperature_calibrated, humidity_calibrated, flame_analog_calibrated]`.
6. Retourner les valeurs calibrées.

Fin

A.3.3 Algorithme 3 : Classification de l'état de la chambre

Début

1. Charger les variables d'entrée : `temperature`, `humidity`, `motion`, `flame_digital`, `flame_analog`.
2. Appliquer le modèle `RandomForestClassifier`.
3. Prédire l'état de la chambre : `no_presence`, `presence`, `hot_room`, `humid_room`, `fire_risk` ou `danger_fire`.
4. Calculer les probabilités de chaque classe.
5. Retourner `predicted_state` et `probabilities`.

Fin

A.3.4 Algorithme 4 : Génération de l'alerte et affichage Flutter

Début

1. Recevoir `predicted_state` et `probabilities`.
2. Si `predicted_state = danger_fire`, alors :
   - `alert_level` ← `critical` ;
   - `alert_message` ← `Danger : risque de feu detecte dans la chambre.`
3. Sinon si `predicted_state = fire_risk`, alors :
   - `alert_level` ← `warning` ;
   - `alert_message` ← `Attention : risque de flamme detecte.`
4. Sinon :
   - `alert_level` ← `normal` ;
   - `alert_message` ← `Etat normal de la chambre.`
5. Construire la réponse JSON : `raw_values`, `calibrated_values`, `predicted_state`, `probabilities`, `alert_level`, `alert_message`.
6. Envoyer la réponse à l'application Flutter.
7. Afficher les valeurs, l'état prédit et les probabilités.
8. Déclencher `SnackBar`, `AlertDialog` ou une notification locale selon `alert_level`.

Fin

A.4 Annexe D : Calibration hybride intelligente

La calibration hybride intelligente a pour objectif de corriger certaines valeurs capteurs avant la classification de l'état de la chambre. Elle combine une méthode statistique simple pour la température et des modèles de régression Random Forest pour l'humidité et la valeur analogique du capteur de flamme.

A.4.1 Correction par biais moyen pour la température

$$biais_{\text{temperature}} = \frac{1}{n} \sum_{i=1}^{n} (T_{\text{brute},i} - T_{\text{reference},i})$$

$$T_{\text{calibree}} = T_{\text{brute}} - biais_{\text{temperature}}$$

A.4.2 Principe de calibration par RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

features = [
    "temperature_raw",
    "humidity_raw",
    "motion",
    "flame_digital",
    "flame_analog_raw"
]

targets = ["humidity_reference", "flame_analog_reference"]

X = data[features]
y = data[targets]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

predictions = model.predict(X_test)
```

Listing A.2 – Extrait simplifié de calibration par RandomForestRegressor

A.4.3 Résumé des résultats de calibration

Variable	Erreur avant calibration	Erreur après calibration	Amélioration
temperature	0.536	0.530	+1.24 %
humidity	1.959	1.716	+12.39 %
flame_analog	18.033	9.313	+48.35 %

A.5 Annexe E : Classification Random Forest

Le modèle `RandomForestClassifier` est utilisé pour prédire l'état de la chambre à partir des variables capteurs. Les classes étudiées sont `no_presence`, `presence`, `hot_room`, `humid_room`, `fire_risk` et `danger_fire`.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix
import joblib

features = ["temperature", "humidity", "motion", "flame_digital", "flame_analog
    "]
target = "state"

X = data[features]
y = data[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)

predictions = classifier.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
```

```
print("Accuracy␣:", accuracy)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))

joblib.dump(classifier, "smart_room_random_forest_model_300.pkl")
```

Listing A.3 – Extrait simplifié d'entraînement du RandomForestClassifier

A.5.1 Résumé des performances de classification

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
danger_fire	0.89	1.00	0.94	8
fire_risk	1.00	0.86	0.92	7
hot_room	1.00	1.00	1.00	13
humid_room	1.00	1.00	1.00	10
no_presence	1.00	1.00	1.00	17
presence	1.00	1.00	1.00	20

L'accuracy obtenue est de 0.99 sur un ensemble de test de 75 observations. La confusion observée entre `fire_risk` et `danger_fire` reste acceptable dans une logique de sécurité, car elle favorise une alerte plus prudente.

A.6 Annexe F : API Flask

L'API Flask constitue l'interface entre les modèles d'intelligence artificielle et l'application Flutter. Elle reçoit ou prépare des valeurs capteurs, applique la calibration, effectue la classification et retourne une réponse JSON structurée.

A.6.1 Structure simplifiée de l'API

```
from flask import Flask, jsonify, request
from flask_cors import CORS
import joblib

app = Flask(__name__)
CORS(app)

calibration_model = joblib.load("sensor_calibration_hybrid_random_forest_model.
    pkl")
```

```
classification_model = joblib.load("smart_room_random_forest_model_300.pkl")

@app.route("/sample", methods=["GET"])
def sample():
    raw_values = {
        "temperature_raw": 32.4,
        "humidity_raw": 45.8,
        "motion": 1,
        "flame_digital": 0,
        "flame_analog_raw": 260
    }
    return predict_from_values(raw_values)

@app.route("/predict", methods=["POST"])
def predict():
    raw_values = request.get_json()
    return predict_from_values(raw_values)
```

Listing A.4 – Structure simplifiée de l'API Flask

A.6.2 Exemple de réponse JSON


```
{
  "alert_level": "critical",
  "alert_message": "Danger : risque de feu detecte dans la chambre.",
  "calibrated_values": {
    "flame_analog_calibrated": 277,
    "humidity_calibrated": 39.0,
    "temperature_calibrated": 32.3
  },
  "predicted_state": "danger_fire",
  "probabilities": {
    "danger_fire": 0.7,
    "fire_risk": 0.3,
    "hot_room": 0.0,
    "humid_room": 0.0,
    "no_presence": 0.0,
    "presence": 0.0
  },
  "raw_values": {
    "flame_analog_raw": 260,
    "flame_digital": 0,
```

```
    "humidity_raw": 45.8,
    "motion": 1,
    "temperature_raw": 32.4
  }
}
```

Listing A.5 – Exemple de réponse JSON retournée par l'API Flask

A.7 Annexe G : Application Flutter et notifications

L'application Flutter permet de visualiser les résultats issus de l'API Flask. Elle affiche les valeurs brutes, les valeurs calibrées, l'état prédit, les probabilités et le message d'alerte.

A.7.1 Structure recommandée du projet Flutter

```
smart_room_flutter_app/
  lib/
    main.dart
    dashboard_screen.dart
    smart_room_api_service.dart
    smart_room_response.dart
    notification_service.dart
    widgets/
      sensor_card.dart
      alert_card.dart
      probability_card.dart
```

Listing A.6 – Organisation simplifiée de l'application Flutter

A.7.2 Principe d'appel de l'API

```
baseUrl = "http://192.168.100.27:5000"

GET /sample
POST /predict

La reponse est convertie en objet SmartRoomResponse puis affichee dans le
    dashboard.
```

Listing A.7 – Principe simplifié d'appel de l'API Flask depuis Flutter

A.7.3 Gestion des notifications

Mécanisme	Utilisation	Objectif
SnackBar	Alerte rapide dans l'interface	Informer immédiatement l'utilisateur
AlertDialog	Alerte critique visible	Attirer l'attention lors d'un danger feu
flutter_local_notifications	Notification locale Android	Préparer l'affichage mobile hors interface active

A.8 Annexe H : Comparaison avec les datasets Kaggle

La comparaison externe a été réalisée avec deux datasets : *Room Occupancy Estimation* pour la logique présence/absence et *Manual Fire Dataset* pour la logique liée au risque de feu. Cette comparaison reste logique et indirecte, car les datasets externes n'utilisent pas exactement les mêmes capteurs ni les mêmes objectifs.

Indicateur	Dataset expérimental	Kaggle Occupancy	Kaggle Manual Fire
Taux de présence	55 %	18.77 %	Non applicable
Taux feu / risque feu	20 %	Non applicable	46.69 %
Température moyenne	28.82 °C	25.45 °C	41.14 °C

A.9 Annexe I : Commandes principales d'exécution

Cette annexe présente les principales commandes utilisées pour lancer les scripts et les composants logiciels du projet.

```
pip install pandas numpy scikit-learn matplotlib joblib flask flask-cors
python sensor_calibration_hybrid_ai.py
python smart_room_ai_with_hybrid_calibration.py
python kaggle_dataset_comparison.py
python smart_room_api.py
```

Listing A.8 – Commandes Python principales

```
cd smart_room_flutter_app
flutter pub get
flutter run
```

Listing A.9 – Commandes Flutter principales

A.10 Annexe J : Fichiers importants du projet

Fichier ou dossier	**Rôle**
arduino_smart_room_data_300.csv	Dataset expérimental initial
arduino_smart_room_data_300_hybrid_calibrated.csv	Dataset après calibration hybride
sensor_calibration_hybrid_ai.py	Script de calibration hybride intelligente
smart_room_ai_300.py	Script de classification Random Forest
smart_room_ai_with_hybrid_calibration.py	Chaîne complète calibration + classification
kaggle_dataset_comparison.py	Comparaison avec datasets Kaggle
smart_room_api.py	API Flask
smart_room_flutter_app/	Application Flutter
smart_room_random_forest_model_300.pkl	Modèle de classification sauvegardé
sensor_calibration_hybrid_random_forest_model.pkl	Modèle de calibration sauvegardé

Conclusion des annexes

Ces annexes complètent le mémoire en fournissant une vue technique détaillée du système proposé. Elles documentent les capteurs, le format des données, les pseudo-algorithmes, la calibration hybride, la classification, l'API Flask, l'application Flutter et les commandes principales. Elles permettent ainsi de mieux comprendre la mise en œuvre pratique de la chaîne complète : capteurs, dataset, calibration, classification, API, application mobile et notifications.