

# Maison intelligente contrôlé par un Agent IA basé sur les LLMs – DeepSeek R1

By Iheb BOUARICHE | 16 02 2025

*Une application conversationnelle qui contrôle un système embarqué  
développée par DeepSeek | LangChain | Python et C#  
Avec une Interface Graphique*

## **Sommaire du Rapport**

1. Introduction
2. Présentation du projet
3. Guide d'utilisation de l'application prototype
4. L'interface graphique (GUI)
5. Le système embarqué
  - Développement matériel
  - Développement logiciel
6. Définition des outils (actions)
7. Conversion d'audio à texte
8. Agent IA
9. LLMs et l'architecture des Transformers
10. Hébergement du modèle dans le Cloud
11. LangChain et le développement des Agents IA
12. L'interface de communication
13. Démonstrations et résultats
14. Difficultés
15. Futurs travaux
16. Conclusion

## Introduction :

Depuis l'invention des semi-conducteurs, la technologie a connu une évolution rapide qui a transformé le monde de l'informatique et de l'électronique. Les semi-conducteurs ont permis la miniaturisation des composants électroniques, conduisant à l'émergence des microcontrôleurs. Ces microcontrôleurs, programmés via des ports logiques, ont constitué une étape clé dans le développement des systèmes embarqués. Par la suite, les microprocesseurs sont apparus et ont permis de développer des systèmes plus complexes, où la programmation s'est centrée sur des algorithmes de plus en plus sophistiqués. L'introduction de l'intelligence artificielle a ajouté une nouvelle dimension à cette évolution, en permettant aux machines non seulement d'exécuter des instructions, mais aussi de prendre des décisions autonomes basées sur des modèles entraînés. Nous sommes aujourd'hui à l'aube d'une nouvelle ère, celle des agents IA, qui redéfinissent la manière de concevoir des systèmes intelligents, ces systèmes qui sont capables de comprendre le langage humain, interagir, suivre les instructions, réfléchir, planifier, et prendre des décisions.

Le développement des systèmes embarqués a, quant à lui, été longtemps dominé par des méthodes de programmation classiques. L'utilisation de langages bas niveau, comme l'assembleur, et de langages de programmation plus accessibles, tels que le C, le C++ ou encore Python, a permis de créer des systèmes fonctionnels, mais statiques. Ces méthodes reposent sur des instructions explicites, où chaque fonction ou tâche doit être décrite avec précision par le programmeur. Les systèmes embarqués conçus de cette manière sont souvent performants, mais manquent de flexibilité pour s'adapter à des environnements changeants ou pour résoudre des problèmes complexes de manière autonome.

Avec l'essor de l'intelligence artificielle et des techniques d'apprentissage automatique, une nouvelle approche a vu le jour : celle de la résolution de problèmes par apprentissage. L'apprentissage supervisé et non supervisé, ainsi que les réseaux de neurones et autres modèles d'IA, ont permis de traiter des problèmes complexes tels que la reconnaissance d'images, le traitement du son et du texte, l'analyse de données massives et la prédiction de comportements. Ces systèmes d'IA peuvent apprendre à partir de données brutes et évoluer en fonction de nouvelles données, offrant ainsi une solution plus flexible et adaptable que les méthodes classiques. Ils représentent un saut quantique dans la capacité des machines à résoudre des problèmes complexes de manière autonome.

Enfin, l'émergence des agents IA marque une nouvelle révolution. Ces agents sont capables d'interagir avec les humains en langage naturel, de comprendre leurs intentions, d'utiliser les outils accessibles, de réfléchir à différentes solutions et de prendre des décisions autonomes et de vérifier les résultats et de répondre aux utilisateurs avec un langage humain. Des modèles comme Llama ou ChatGPT illustrent parfaitement cette avancée. Ces modèles, basés sur des architectures de Transformers, sont entraînés sur d'immenses volumes de données textuelles pour comprendre, générer et interagir de manière fluide en langage naturel.

En résumé, nous avons commencé avec l'automatisation en électronique, où les tâches étaient réalisées à l'aide de circuits logiques et de portes électroniques simples. Puis, avec l'avènement des microcontrôleurs et des microprocesseurs, nous avons commencé à programmer directement avec des instructions logiques via des langages comme l'assembleur. Par la suite, l'introduction de langages plus abstraits comme le C, le C++ et Python a permis une programmation plus intuitive, mais toujours basée sur des algorithmes rigides. Dernièrement et grâce à l'intelligence artificielle nous avons pu développer des modèles et de les entraîner sur des bases de données pour résoudre des problèmes.

complexe. Aujourd'hui on a les agents IA, qui sont capable de suivre les instructions humaines en langage naturel, réfléchir, résoudre des problèmes, prendre des décisions, constater les résultats et interagir avec l'humain.

## **Présentation du projet :**

Ce projet a pour objectif de développer un système embarqué piloté par un agent d'intelligence artificielle, conçu comme un assistant capable de comprendre, réfléchir, résoudre des problèmes et de contrôler les dispositifs d'un système embarqué comme des capteurs et des actionneurs et d'interagir en langage naturel.

Le système embarqué conçu avec des dispositifs d'une maison connecté et contrôlé directement par une carte électronique. Il inclut un ventilateur pour la ventilation, des lampes pour l'éclairage, ainsi que des capteurs mesurant la température et l'humidité. Le système est connecté avec une caméra pour la surveillance et aussi un microphone pour la réception du signal de son.

La partie logicielle est implémenté dans la carte qui contrôle les entrée et les sorties de ce système. Ce programme est développé autant qu'un Agent capable de recevoir des instructions en langage naturel, comprendre, réfléchir, planifier et exécuter les instructions demandées, et échanger par texte avec l'utilisateur après l'exécution.

Cet Agent a accès à tous les entrée et les sorties du système embarqué et utilise ces dispositifs pour réaliser les tâches demandées.

Le système est capable de répondre à des questions liées à la température, l'humidité et la pression de l'environnement, ainsi des sujets liés à l'activation et la désactivation des lampes et de ventilation, ainsi à la prise des photos et le traitement des photos. Ainsi ce système a accès aux informations au Web et Wikipédia et est développé pour répondre à des questions générales ou effectuer des recherches sur le web pour fournir des réponses pertinentes aux utilisateurs.

Le système est conçu pour être évolutif (en- scalable), permettant l'ajout de nouvelles fonctionnalités dispositifs et outils.

## **Guide d'utilisation d'application prototype :**

Dans cette section, nous allons expliquer l'utilisation de notre application qui contrôle un système embarqué ainsi que le guide destiné à l'utilisateur.

Cette application permet à l'utilisateur d'interagir avec un agent IA en langage naturel afin d'effectuer diverses tâches, résoudre des problèmes, contrôler des dispositifs dans une maison intelligente ou obtenir des informations.

Voici un exemple de demande utilisateur :

*"What's the solutions of this equation  $x^2-9=0$ , if the solution equal 2 or -2, turn on the light one, if else, turn on the light 2"*

Comme illustré dans l'image ci-dessous, l'interface graphique se compose de plusieurs éléments :

- Une zone de saisie où l'utilisateur peut écrire et échanger en conversation avec l'agent IA.
- À droite, un bouton « Send » permettant de transmettre le texte saisi.

- Au-dessus de la zone de saisie, un bouton « Record » qui permet à l'utilisateur d'enregistrer sa voix pour interagir avec l'agent IA à l'oral.

The screenshot shows a window titled 'Form1' with a light orange background. At the top left, the text 'Final Response:' is displayed. Below this, there is a large empty rectangular area. In the center of this area is a gray button labeled 'Record'. At the bottom of the window, there is a white text input field containing the text: 'What's the solutions of this equation  $x^2-9=0$ , if the solution equal 2 or -2, turn on the light one, and else, turn on the light 2'. To the right of the input field is a gray button labeled 'Send'.

**Figure 01 :** L'application sur l'interface graphique.

À partir du texte ou de la voix, l'utilisateur peut communiquer et transmettre ses instructions à l'agent IA. Celui-ci analysera la demande, effectuera les tâches requises, puis affichera sa réponse dans la section dédiée de l'interface, intitulée Réponse finale « Final Response ».

The screenshot shows the same window 'Form1' as Figure 01. The 'Final Response:' section now contains the text: 'The solutions to the equation  $x^2 - 9 = 0$  are 3 and -3. Since these are not equal to 2 or -2, I have turned on the light in Room 2 as per your instructions.' The input field and buttons remain the same as in Figure 01.

**Figure 02 :** L'application sur l'interface graphique.

Ensuite, l'utilisateur peut poursuivre l'échange avec l'agent IA pour donner de nouvelles instructions ou demander des informations sur les tâches effectuées.

On constate que l'agent a bien suivi les instructions et qu'il a résolu le problème mathématique et a pris la décision d'allumer la LED numéro 2.

A partir de cette interface l'utilisateur peut demander des tâches sur les dispositifs disponibles sur le système et aussi d'autres questions ou problèmes à résoudre.

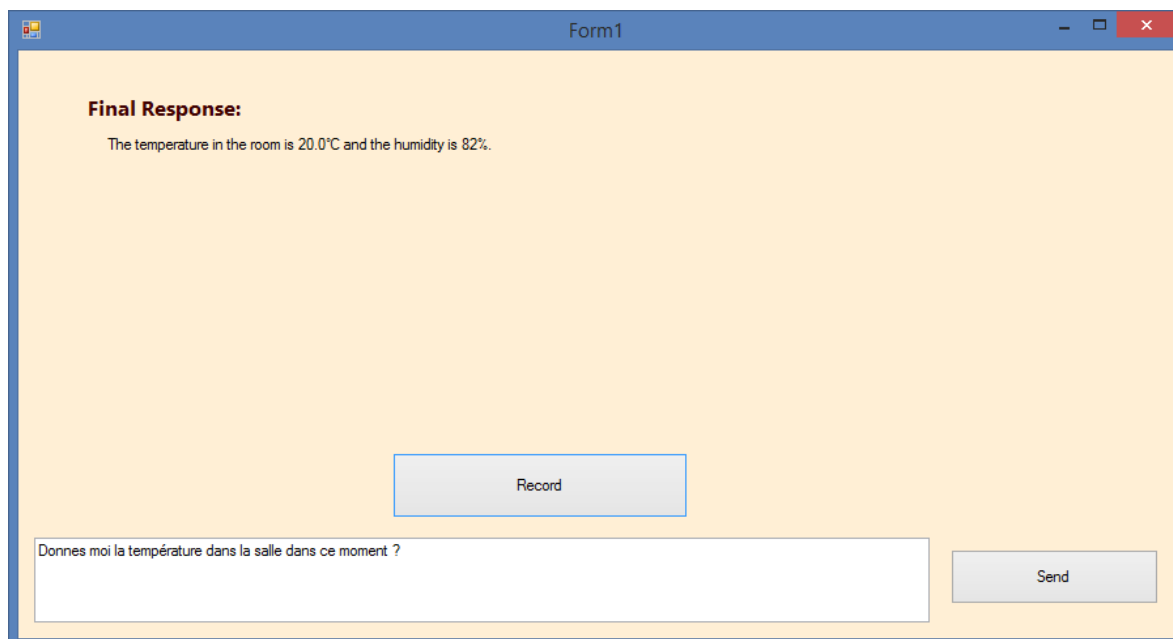
### L'interface Graphique (GUI) :

Afin de faciliter l'interaction avec notre agent IA, nous avons développé une interface graphique sur un PC, connectée avec le système embarqué qui est séparé par rapport au PC.

Cette interface est développée avec le langage de programmation C#, est simple et comprend un bouton pour envoyer les données ainsi qu'un champ de saisie permettant d'entrer des données au format texte et les envoyer à notre système embarqué développé avec le langage Python.

Cette interface dispose également d'un bouton pour l'enregistrement vocal ce bouton envoie une instruction au système embarqué pour activer le microphone pour l'enregistrement.

Dans la partie supérieure, une section affiche la réponse finale reçue par le programme Python du système embarqué.



**Figure 03 :** L'application sur l'interface graphique.

L'interface permet d'envoyer des requêtes et de recevoir les réponses finales. Cette application graphique, s'exécute sur un PC et communique avec un code Python implémenté dans un système embarqué via une communication basée sur le protocole TCP.

Dans cette architecture, le PC joue le rôle de client, tandis que le système embarqué agit en tant que serveur.



## Le système embarqué :

### Développement Matériel :

Le système est composé d'une partie matérielle comprenant des capteurs et des actionneurs connectés à une carte électronique de type Raspberry Pi 4 model B. Le choix de cette carte s'explique par sa puissance de calcul, sa connectivité, sa grande capacité de mémoire, son système d'exploitation adapté, sa facilité de programmation, ainsi que la possibilité d'interpréter et d'exécuter des programmes en Python et d'installer les différents packages nécessaires.

séries	Raspberry Pi 4 Model B
Système d'exploitation	Linux
Plate-forme du matériel informatique	Not Machine Specific
Marque du processeur	ARM
Vitesse du processeur	1,5 Ghz
Nombre de coeurs	4
Taille de la mémoire vive	4 Go
Mémoire maximale	4 Go
Interface du disque dur	Ethernet

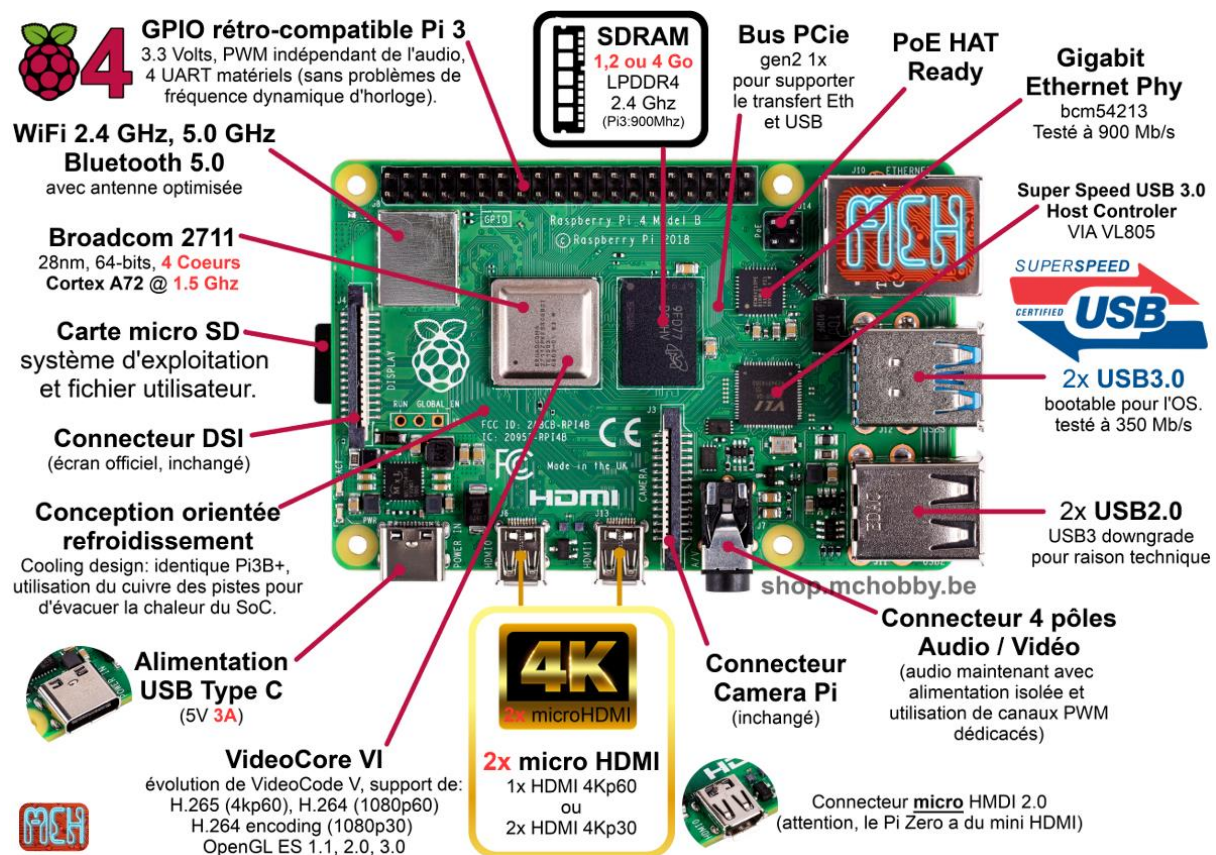
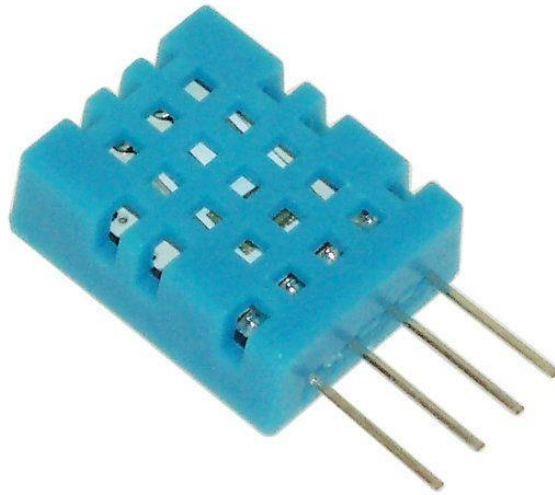


Figure 05 : Carte Raspberry + ses éléments.

Voici les capteurs utilisés pour ce projet :

1- Capteur de température et d'humidité :



**Figure 06** : Capteur de température et d'humidité.

2- Une caméra :



**Figure 07** : Module de caméra.



3- Un ventilateur commandé par un moteur pas à pas et un pont H :



**Figure 08 :** Ventilateur.

4- Lampes (LEDs) :

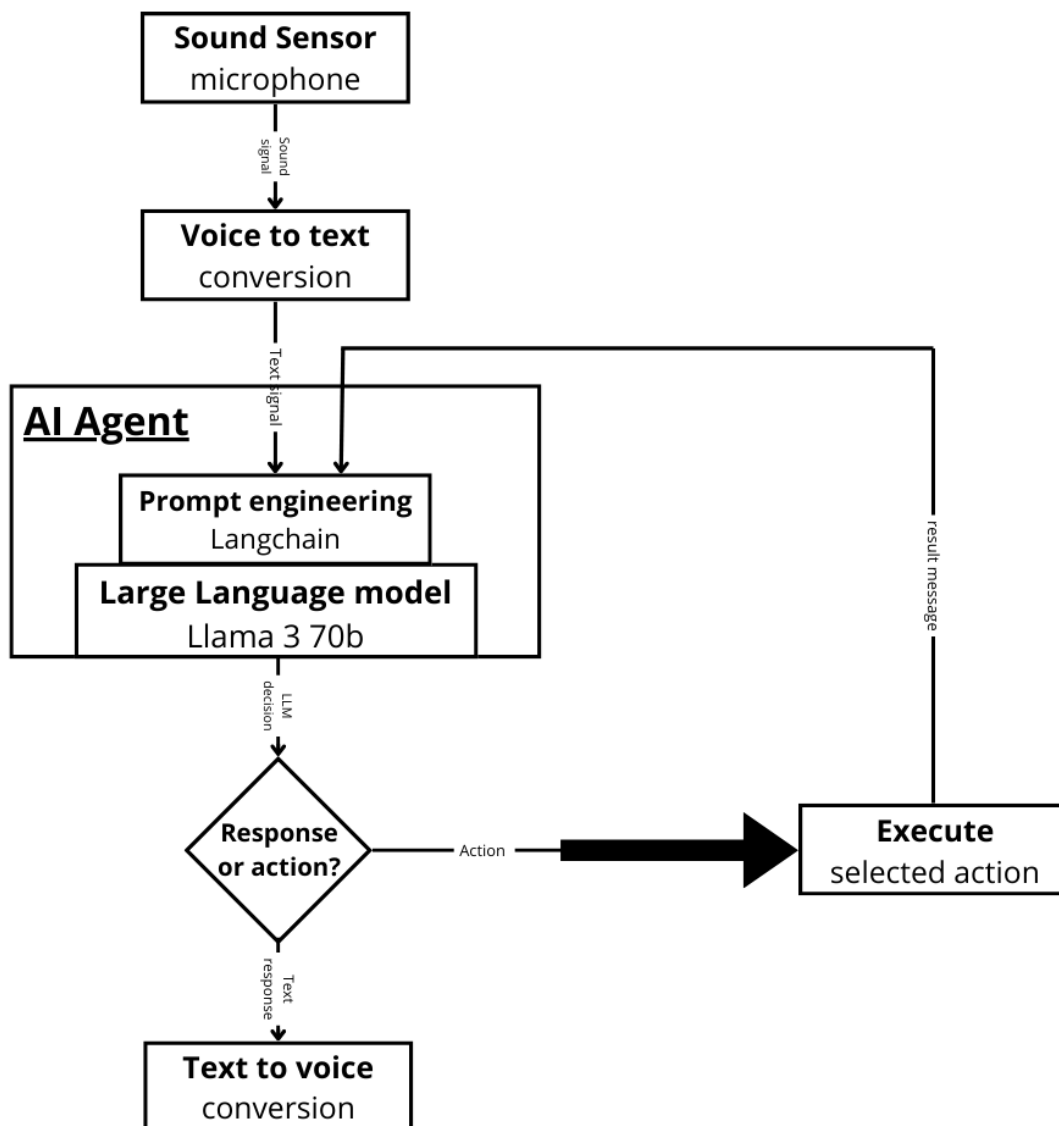


**Figure 09 :** Une LED.

### Développement Logiciel :

La partie logicielle est développée en Python et implémentée sur la carte et programmée pour recevoir en entrée un signal audio ou un texte en langage naturel en anglais. Cette information est ensuite traitée par un modèle de traitement de langage de grande taille basé sur l'intelligence artificielle, ce modèle va prendre des décisions à partir des instructions fournies. Après l'exécution des instructions demandées par l'utilisateur, le programme génère une réponse sous forme d'un texte et envoie cette réponse à notre interface graphique développée en C#. De cette manière, on a un assistant capable d'interagir avec l'utilisateur par la voix ou du texte.

Le logigramme représentant le programme de cette partie est illustré dans la figure ci-dessous :



**Figure 10 :** Logigramme du programme principale.

## **1- Définition des outils (actions) :**

La première partie consiste à développer des outils pour l'agent IA. L'agent IA est capable d'utiliser des outils programmés pour exécuter les tâches demandées par l'utilisateur. Par exemple, lorsqu'un utilisateur demande d'allumer une lumière, l'agent peut invoquer le programme correspondant, vérifier l'exécution de l'action, puis informer l'utilisateur que la tâche a été réalisée avec succès.

Ces outils sont implémentés sous forme de fonctions Python, avec un retour textuel transmis à l'agent contenant le résultat de l'exécution. L'agent peut ainsi interpréter ce retour, planifier les actions à venir, et fournir un retour pertinent à l'utilisateur.

Voici la liste des outils accessibles à l'agent IA dans ce projet :

- A. Allumer ou éteindre les lumières : L'agent peut allumer ou éteindre les lumières de n'importe quelle pièce en réponse à une demande de l'utilisateur.
- B. Faire clignoter les lumières : l'agent peut clignoter les Leds avec une certaine fréquence, tout dépend la demande de l'utilisateur.
- C. Vérifier l'état des lumières : L'agent peut obtenir l'état des lumières dans chaque pièce pour effectuer des vérifications ou préparer d'autres actions.
- D. Appliquer une fréquence à un pin de sortie : L'agent peut configurer une fréquence ou une période spécifique pour un signal sur un pin de sortie.
- E. Obtenir la valeur de la température : L'agent peut mesurer et transmettre la température.
- F. Obtenir la valeur de l'humidité : L'agent peut mesurer et transmettre le taux d'humidité.
- G. Démarrer le ventilateur : Cette partie utilise un moteur pas à pas pour contrôler la vitesse du ventilateur. L'agent peut activer le ventilateur sur demande, ajuster la vitesse du moteur du ventilateur, et lancer le programme de génération de signaux alternant entre les différents nœuds du stator en parallèle.
- H. Surveiller l'environnement, et traiter les images pour détecter les objets et les personnes : Dans cette partie, l'agent a accès à une caméra pour prendre une photo et traiter cette image en utilisant des modèles d'intelligence artificielle pour le traitement d'image. Dans notre programme, c'est YOLO qui est utilisé pour détecter les objets et les personnes présents dans l'image. Ensuite, à partir de la sortie du modèle de détection, l'agent peut répondre à des questions liées à l'image. Dans notre projet on a utilisé le modèle YOLO v11 avec la taille moyenne qui est une version très récente et parmi les modèles les plus puissant en termes de détection et localisation des objets.  
(YOLOv11 est la dernière itération de la série de modèles de détection d'objets en temps réel "You Only Look Once" (YOLO), développée par Ultralytics. Cette version introduit des améliorations significatives en termes de précision, de vitesse et d'efficacité, consolidant ainsi la position de YOLO en tant que référence dans le domaine de la vision par ordinateur, Les performances de YOLOv11 varient en fonction de la taille du modèle et de la tâche spécifique).

Modèle	Noms de fichiers	Tâche	Inférence	Validation	Formation	Exportation
YOLO11	yolo11n.pt yolo11s.pt yolo11m.pt yolo11l.pt yolo11x.pt	Détection	✓	✓	✓	✓
YOLO11-seg	yolo11n-seg.pt yolo11s-seg.pt yolo11m-seg.pt yolo11l-seg.pt yolo11x-seg.pt	Segmentation des instances	✓	✓	✓	✓
YOLO11-pose	yolo11n-pose.pt yolo11s-pose.pt yolo11m-pose.pt yolo11l-pose.pt yolo11x-pose.pt	Pose/Points clés	✓	✓	✓	✓
YOLO11-obb	yolo11n-obb.pt yolo11s-obb.pt yolo11m-obb.pt yolo11l-obb.pt yolo11x-obb.pt	Détection orientée	✓	✓	✓	✓
YOLO11-cls	yolo11n-cls.pt yolo11s-cls.pt yolo11m-cls.pt yolo11l-cls.pt yolo11x-cls.pt	Classification	✓	✓	✓	✓

**Figure 11 :** Tableau des différents version de modèles de YOLO.

- I. Répondre aux questions sur la météo : L'agent a accès à des informations météorologiques pour toutes les villes du monde et peut répondre avec précision aux demandes. Pour cette partie le modèle a accès à des APIs développées en python pour avoir accès à des informations sur la météo dans le monde.
- J. Rechercher sur le Web et répondre à des questions générales : L'agent peut effectuer des recherches sur Wikipédia pour fournir des réponses détaillées et précises, en analysant les informations trouvées au lieu de simplement lire le texte brut. Pour cela, le modèle a accès aussi à l'API de Wikipédia qui donne accès pour faire des recherches et recevoir les informations sous formes de textes. Dans ce cas le modèle envoie exactement le sujet qu'il cherche et suivant la demande de l'utilisateur. Et ensuite il va sortir les réponses du texte reçu par Wikipédia pour répondre l'utilisateur d'une manière pertinente.

Notre modèle d'agent IA maîtrise la réflexion, la résolution de problèmes et la compréhension des différents langages. Il est entraîné sur ces compétences, qui sont intégrées dans ses connaissances :

- K. Résolution de problèmes complexes en mathématiques et dans d'autres domaines.
- L. Analyse et décomposition d'un problème en plusieurs sous-problèmes pour une meilleure résolution.
- M. Traduction de textes et rédaction de résumés.
- N. Génération de textes et connaissance générale et assistance à l'utilisateur.

En cas d'incapacité à réaliser une tâche demandée, l'agent fournira une réponse pertinente expliquant les raisons de cette limitation et rappellera les outils et fonctionnalités disponibles.

### **1. Conversion d'audio à texte :**

C'est la première partie après la réception de donnée, notre programme prend en entrée un texte ou un signal d'audio à partir d'un capteur de son (microphone), si c'est un audio, ce signal va être converti en un texte à partir d'un modèle entraîné qui s'appelle Whisper.

Whisper est un modèle de reconnaissance vocale développé par OpenAI, capable de transcrire des discours en plusieurs langues. Utilisant une architecture d'apprentissage profond - transformateur encodeur-décodeur -, il a été entraîné avec 680 000 heures de données audio et est conçu pour être robuste face aux bruits de fond et jargons. Il a été lancé en open-source en septembre 2022 et est utilisé

dans diverses applications, telles que la reconnaissance vocale. Dans ce projet ce modèle est implémenté dans la carte et représente la première partie du programme.

## 2. Agent IA :

L'agent IA représente le cœur du système, responsable de la réflexion et de la prise de décisions. Il reçoit des données en format texte en entrée et produit des sorties également sous forme de texte. Son rôle consiste à comprendre le langage, à identifier les outils et actions disponibles, puis à réfléchir et planifier les étapes nécessaires pour réaliser les instructions demandées. Afin de mener à bien une tâche, l'agent choisit entre deux options : exécuter une action ou fournir une réponse finale. Dans le cas d'une action, l'agent demande à ce qu'elle soit exécutée et attend la confirmation de sa réalisation et le retour de cette action avant de poursuivre. Le modèle d'un agent consiste à deux parties :

- La première partie, intitulée *-Prompt Engineering-*, consiste en la reformulation et la préparation du texte d'entrée. Elle comprend également l'ajout des informations pertinentes sur les outils disponibles, ainsi que les détails relatifs à l'environnement et au rôle du système. Cette étape inclut aussi l'intégration des messages des échanges précédentes et les résultats des actions pour donner au modèle une capacité de mémorisation. Voici une partie du prompt utilisé pour ce système :

```
''' You are a conversational and helpful assistant integrated into a smart home environment. you perform all the tasks and follow the instruction without ignoring.
```

```
Your capabilities include:
```

```
Managing lighting in rooms one, two, and three (turn on/off and check states or control frequency of blinking, by using blinking light tool, the lights are on and blinking directly without verification).
```

```
Searching the internet for information.
```

```
taking pictures and detecting persons and objects
```

```
controlling and fun motor
```

```
getting information about weather in cities in the world
```

```
getting information about temperature and humidity in the room
```

```
Guidelines:
```

```
follow the instruction and wait for observation before taking the next action. you need to start by taking just the first needed action and waiting for observation for taking the next action and then do the same thing until finishing the tasks.
```

```
don't give a Json response with more than one action in the same time, you need to give one Json with one action and wait the execution and after observation take the next action.
```

```
don't give a response if you don't see it in the observation and always take actions depend on the observation
```

```
don't give a final answer if you don't finish instruction in the input, you need to give an answer after finishing all the instructions.
```

```
for each input asking, you can use one tool or action and after that you verify the observation and take the next tool or finale response to complete tasks, only observation that give you the real state of the system, you reason and take actions
```

```
for frequencies, you can use only integers in the input and you can calculate the frequency if needed
```

```
for time waiting, you can use only integers and the input that you choose is on seconds, for example the integer 3 means you will wait for 3 seconds.
```

each action returns an observation, without observation you need to recall the action.

The frequency that you can use is between 0.1 and 60.

Remember, you are integrated into a smart home system and should respond in an intelligent, user-friendly, and conversational manner, providing the most accurate and efficient service and by taking all the actions needed. Confirm your actions at every step and offer to assist further if needed.'''

- La deuxième partie - Large Language Model (LLM) – consiste l'utilisation des modèles de langage de grande taille, qui sont des modèles d'intelligence artificielle basés sur une architecture des réseaux de neurones profonds -Transformer- et entraînés sur d'énormes ensembles de données textuelles. Ces modèles sont capables de traiter, de comprendre et de générer du texte en imitant la langue humaine avec un niveau de sophistication élevé.

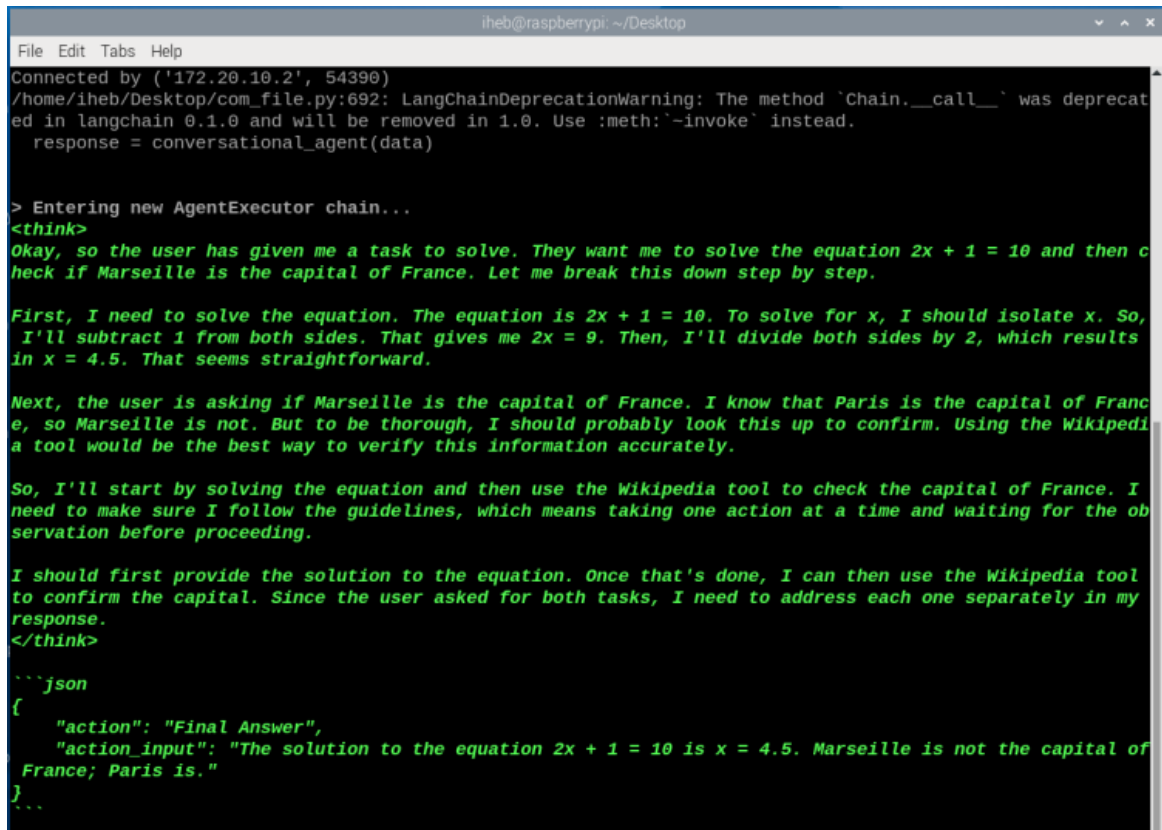
Dans ce projet, nous avons utilisé un modèle pré-entraîné de Meta appelé « LLaMA » puis au début de 2025 on a constaté la sortie du nouveau modèle chinois « DeepSeek » qui était développé pour réfléchir, planifier et résoudre des problèmes, ce qui nous a poussé à basculer à ce modèle. Le choix de ces modèles par rapport à d'autres modèles comme ChatGPT repose sur plusieurs facteurs : il est open-source, gratuit et figure parmi les modèles les plus puissants au monde.

La version de « DeepSeek R1 » offre des performances exceptionnelles en matière de compréhension et de raisonnement, tout en permettant une capacité de planification et de résolution de problèmes d'une grande précision, ce modèle décompose le problème en plusieurs problèmes et prend les décisions d'une manière précise. Le prompt, qui contient toutes les informations et instructions demandées par l'utilisateur, est structuré sous forme de texte et envoyé au modèle pour qu'il traite le texte, et réfléchit, prend une décision, pour générer une réponse ou effectuer une action. Après la réalisation des instructions demandées par l'utilisateur, le modèle fournit une réponse finale en retour.

Voici un exemple de ce modèle de langage entraîné à réfléchir et décomposer le problème après qu'on lui a demandé cette instruction :

*"Hi!, I want you to solve the equation of two multiply by x plus one equal ten, and then tell me if Marseille is the capital of France ?"*





```
iheb@raspberrypi: ~/Desktop
File Edit Tabs Help
Connected by ('172.20.10.2', 54390)
/home/iheb/Desktop/com_file.py:692: LangChainDeprecationWarning: The method `Chain.__call__` was deprecated in langchain 0.1.0 and will be removed in 1.0. Use :meth:`~invoke` instead.
  response = conversational_agent(data)

> Entering new AgentExecutor chain...
<think>
Okay, so the user has given me a task to solve. They want me to solve the equation  $2x + 1 = 10$  and then check if Marseille is the capital of France. Let me break this down step by step.

First, I need to solve the equation. The equation is  $2x + 1 = 10$ . To solve for  $x$ , I should isolate  $x$ . So, I'll subtract 1 from both sides. That gives me  $2x = 9$ . Then, I'll divide both sides by 2, which results in  $x = 4.5$ . That seems straightforward.

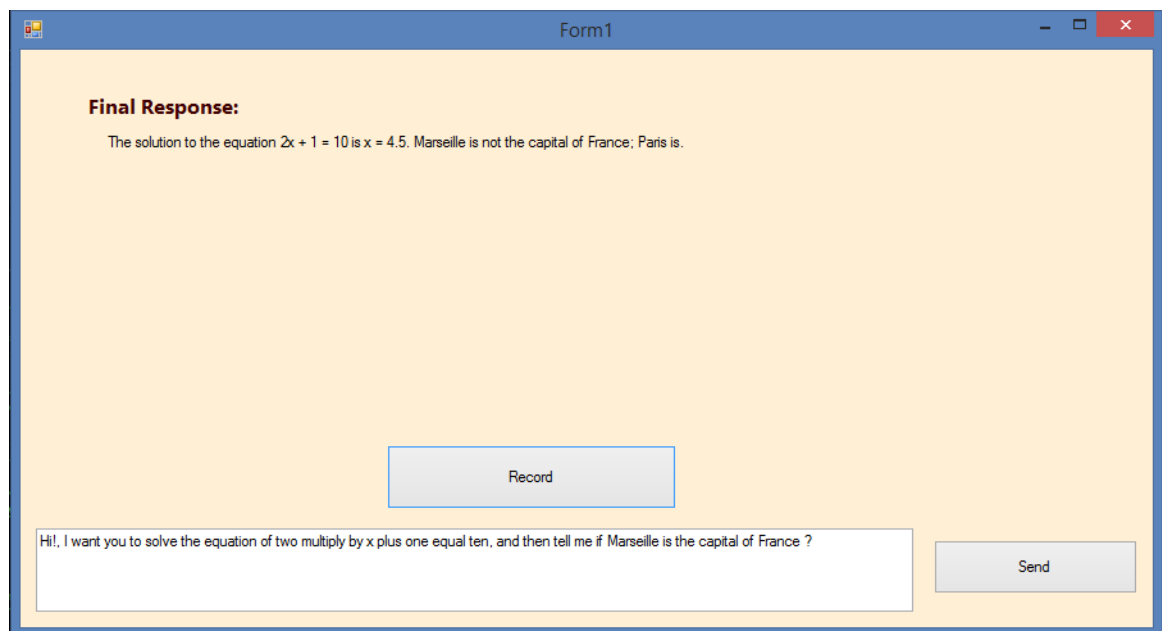
Next, the user is asking if Marseille is the capital of France. I know that Paris is the capital of France, so Marseille is not. But to be thorough, I should probably look this up to confirm. Using the Wikipedia tool would be the best way to verify this information accurately.

So, I'll start by solving the equation and then use the Wikipedia tool to check the capital of France. I need to make sure I follow the guidelines, which means taking one action at a time and waiting for the observation before proceeding.

I should first provide the solution to the equation. Once that's done, I can then use the Wikipedia tool to confirm the capital. Since the user asked for both tasks, I need to address each one separately in my response.
</think>
{
  "action": "Final Answer",
  "action_input": "The solution to the equation  $2x + 1 = 10$  is  $x = 4.5$ . Marseille is not the capital of France; Paris is."
}
```

Figure 12 : Fenêtre CMD d'arrière-plan.

**Commentaire :** Dans cet exemple, on peut bien voir que l'agent IA a dévisé ce problème a plusieurs problèmes et ainsi a pensé à utiliser Wikipédia pour répondre mais finalement, il a décidé de répondre directement, et donc cet agent a bien suivi les instructions demandées par l'utilisateur et a résolu le problème.



**Form1**

**Final Response:**

The solution to the equation  $2x + 1 = 10$  is  $x = 4.5$ . Marseille is not the capital of France; Paris is.

Record

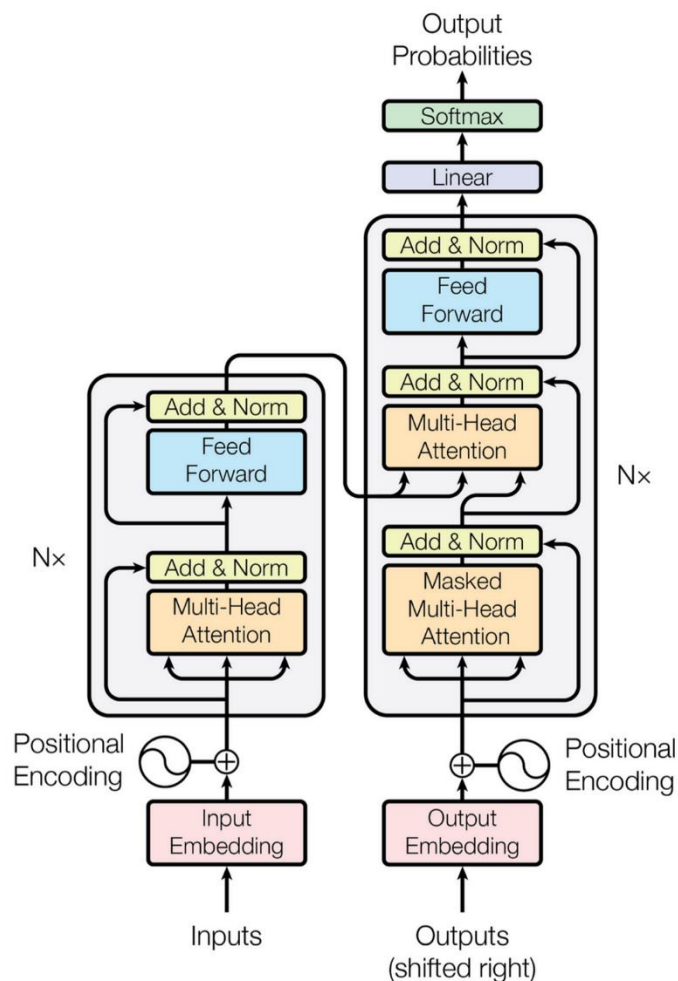
Hi!, I want you to solve the equation of two multiply by x plus one equal ten, and then tell me if Marseille is the capital of France ?

Send

Figure 13 : L'application sur l'interface graphique.

Les modèles de langage de grande taille (LLMs) reposent sur une architecture de réseaux de neurones de type Transformer, conçue pour traiter efficacement les séquences de données, notamment dans le domaine du traitement du langage naturel. Leur caractéristique principale réside dans le mécanisme d'attention, en particulier l'attention multi-têtes, qui permet au modèle de se concentrer simultanément sur différentes parties de la séquence et de capturer les relations contextuelles entre les tokens (mots) grâce au mécanisme de self-attention.

Les Transformers permettent une parallélisation complète, accélérant ainsi l'entraînement sur d'immenses ensembles de données textuelles. Grâce à leur architecture modulaire, composée de couches empilées d'attention et de réseaux neuronaux classiques, les Transformers offrent une grande flexibilité pour l'encodage, le décodage ou les deux, selon les besoins spécifiques.



**Figure 14 :** L'architecture du Transformateur (Transformer).

Initialement conçus pour des tâches comme la traduction automatique, les Transformers se sont étendus à divers domaines tels que la vision par ordinateur, le traitement d'audio, la bio-informatique et la génération de texte. Des modèles tels que ChatGPT, Llama ou DeepSeek, utilisés dans ce projet, illustrent leur polyvalence grâce à leur capacité à gérer de vastes quantités de données et de paramètres. Leur scalabilité et leur adaptabilité en font des outils essentiels dans les avancées de l'intelligence artificielle moderne.

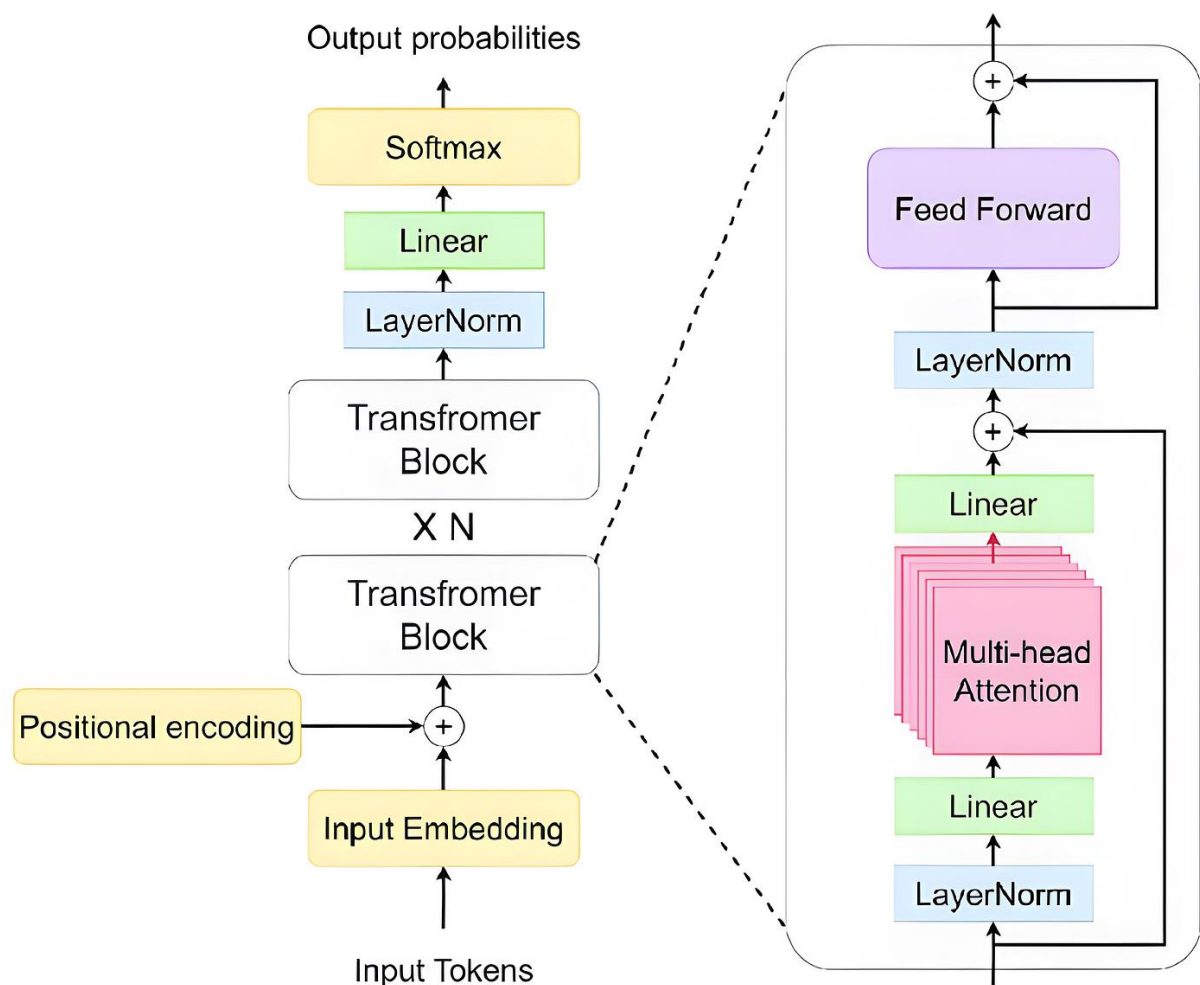
Dans le cadre de ce projet, le modèle utilisé est « DeepSeek R3 », un modèle de grande taille comprenant des milliards de paramètres, basé sur une architecture de décodage et non sur

l'encodage de l'architecture du Transformer. Ce modèle est capable de comprendre le langage humain et de répondre dans plusieurs langues sous forme de texte.

### Le fonctionnement et l'architecture des LLMs – Transformer :

Dans cette section, nous allons expliquer le fonctionnement de l'architecture des Large Language Models (LLMs), cette partie représente le cerveau de notre système, et repose sur une architecture de Transformer. Les LLMs sont construits spécifiquement à partir de la partie décodeur du modèle Transformer. Cette partie est composée de plusieurs couches de traitement de données, organisées de manière séquentielle.

Le décodeur prend en entrée des données textuelles et produit une réponse également sous forme de texte. Entre l'entrée et la sortie, les données textuelles sont converties en représentations numériques. Voici les étapes séquentielles :



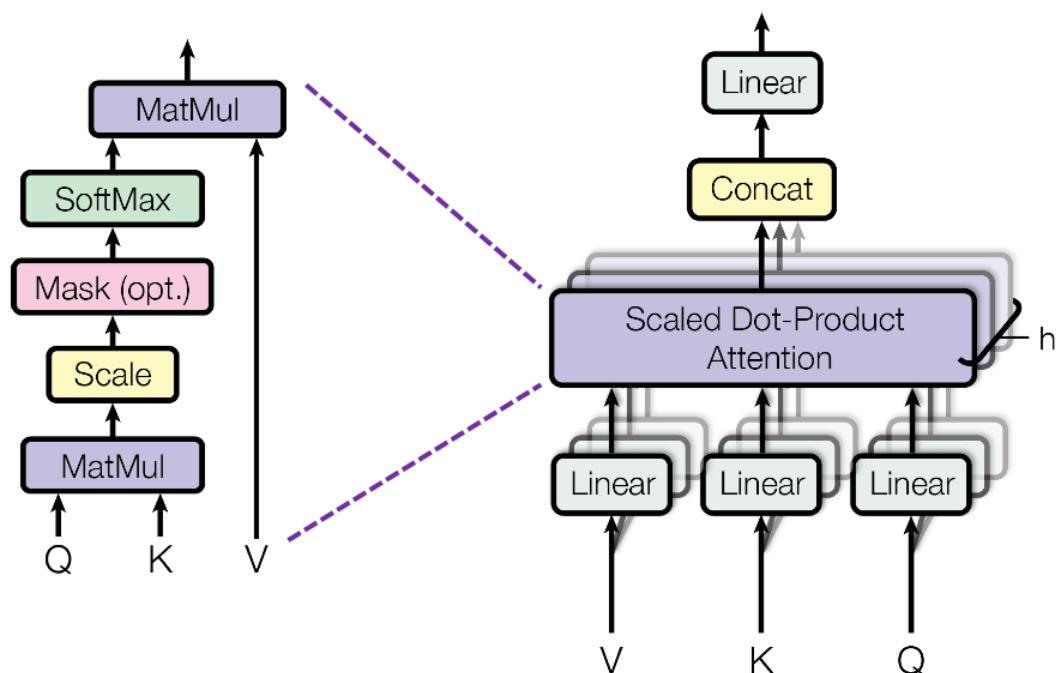
**Figure 15 :** Le décodeur du transformateur.

- a- Tokenisation : La tokenisation divise le texte en unités appelées *tokens* (mots, sous-mots ou caractères), ensuite elle donne pour chaque token une valeur représentative.
- b- Encodage des mots : Chaque token est transformé en un vecteur dense via une couche réseau de neurone d'*embedding*, qui encode des relations sémantiques entre les mots dans un espace multidimensionnel. D'une manière simple, chaque token va rentrer dans le modèle de d'encodage des mots pour sortir un vecteur qui représente le sens du mot dans un espace multidimensionnelle.

- c- Encodage positionnel : Comme les Transformers n'ont pas de mémoire séquentielle, l'encodage positionnel ajoute des vecteurs représentant les positions des tokens pour le vecteur d'encodage des mots pour donner représenter la position de chaque mot dans une séquence de mots, et ce qui permettre de conserver l'ordre des mots.
- d- Le mécanisme de self-attention : c'est la couche qui prend en entrée le vecteur d'encodage, ce mécanisme permet de déterminer l'importance de chaque mot d'une séquence par rapport aux autres mots de cette même séquence. Chaque vecteur d'encodage de chaque mot est d'abord transformé en trois vecteurs : une requête (Query), qui représente ce que le mot cherche chez les autres, une clé (Key), qui indique ce que le mot offre comme information, et une valeur (Value), qui représente la donnée réelle associée au mot.

Ensuite, pour chaque mot, on calcule un score d'importance en effectuant un produit scalaire entre son vecteur Query et les vecteurs Key de tous les autres mots. Ces scores sont normalisés à l'aide d'une fonction comme le softmax, afin d'obtenir des poids relatifs. Ces poids sont utilisés pour pondérer les vecteurs Value des mots, permettant de combiner les informations les plus pertinentes pour chaque mot tout en ignorant celles qui sont moins utiles dans le contexte.

Le résultat final est une somme pondérée des valeurs des autres mots, reflétant leur pertinence pour le mot en question. Ce mécanisme permet au modèle de capturer efficacement les relations contextuelles globales, même entre des mots éloignés dans une phrase, en tenant compte de l'ensemble du contexte.



**Figure 16** : Le mécanisme de l'attention.

D'une manière simple, le self-attention prend en entrée le vecteur qui représente la signification de chaque mot et produit des vecteurs qui donne la signification du mot et sa relation entres les mots dans la séquence, et capturent les interactions et les dépendances entre les mots, donnant au modèle une compréhension contextuelle riche de la phrase ou du passage traité.

- e- Multi-têtes : Les *multi-têtes* permettent de réaliser plusieurs mécanismes de self-attention en parallèle, chacun capturant des aspects différents des relations contextuelles dans les données.
- f- La normalisation : La normalisation stabilise les valeurs dans les couches du modèle, rendant l'apprentissage plus efficace et évitant des déséquilibres dans les activations.
- g- Le réseau à propagation directe applique des transformations non linéaires pour apprendre des relations complexes entre les tokens. Il génère une réponse mot par mot en utilisant la sortie précédente comme entrée pour produire le mot suivant, permettant ainsi de construire une séquence cohérente jusqu'à ce que la réponse soit complète.

De cette manière, on obtient un modèle capable de comprendre le langage humain après avoir été entraîné sur d'énormes bases de données textuelles. Par la suite, il peut être réentraîné pour être performant dans des tâches de résolution de problèmes, ainsi que dans la génération de réponses et l'assistance.

### **Hébergement du modèle dans le Cloud :**

Toutefois, en raison de la taille importante des modèles LLMs et du temps d'exécution élevé, il n'est pas possible de l'utiliser localement sur la carte. Pour surmonter cette contrainte, le modèle est hébergé dans le Cloud de « Groq ». Le système envoie des prompts en format de texte et reçoit les réponses depuis les serveurs de « Groq ».

« Groq » est une entreprise américaine spécialisée dans l'intelligence artificielle, qui conçoit des processeurs avancés appelés unités de traitement du langage (LPU). Ces processeurs sont conçus pour accélérer les performances des modèles d'IA générative et des charges de travail d'inférence. La plateforme « Groq » optimise l'exécution de modèles d'intelligence artificielle de grande taille, tels que « Llama 3 70B », « DeepSeek R1 » et d'autres modèles, grâce à des processeurs hautement performants. Cela permet de réduire les temps de traitement et d'offrir une exécution rapide et efficace directement depuis le Cloud.

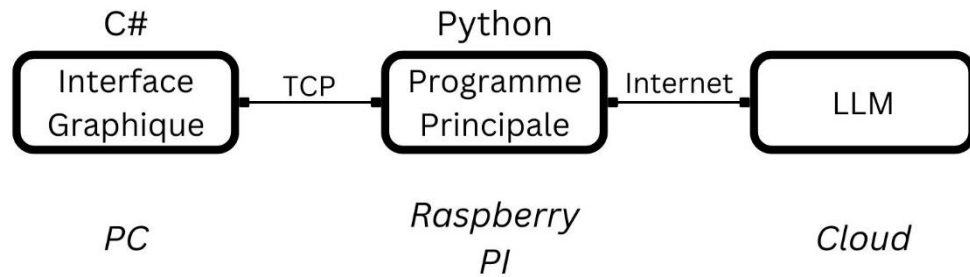
### **3. LangChain et le développement des Agents IA :**

Dans ce projet, nous avons développé notre agent en utilisant le Framework « LangChain », qui est capable de structurer les entrées envoyées au LLM et d'interpréter les sorties générées par ce dernier. Avec le « LangChain », le texte reçu par le LLM sera interprété à une action, et cette action sera exécutée par Python.

### **L'interface de communication :**

Notre application fonctionne par l'interaction de trois systèmes :

1. Un PC qui exécute le programme de l'interface graphique.
2. Une carte Raspberry qui gère tous les outils de l'agent IA et qui est connectée avec un système embarqué.
3. Des ressources pour la puissance de calcul dans le Cloud utilisés pour héberger et exécuter de grands modèles de langage, notamment « DeepSeek R1 ».



**Figure 17 :** Les communication entre les différents systèmes.

L'application graphique sur le PC communique uniquement avec la carte Raspberry en envoyant des prompts sous forme de texte ou en déclenchant l'enregistrement vocal. Les données sont transmises via le protocole TCP sur une connexion Wi-Fi.

Le programme implémenté sur la carte ajoute aux données reçues par l'application des sections de texte (prompts) avant de les envoyer. Ces prompts contiennent tous les détails sur le système, les outils disponibles et le fonctionnement général, ainsi que les instructions demandées par l'utilisateur. Ils sont ensuite transmis au cloud de Groq, où notre modèle de langage est hébergé.

Le modèle LLM traite ces informations et renvoie une réponse qui peut contenir soit une action à exécuter, soit une réponse finale pour l'utilisateur. Si c'est une action, le programme sur la carte Raspberry l'exécute, et renvoie les résultats au modèle LLM sur le cloud. Ce processus se répète jusqu'à la réception d'une réponse finale.

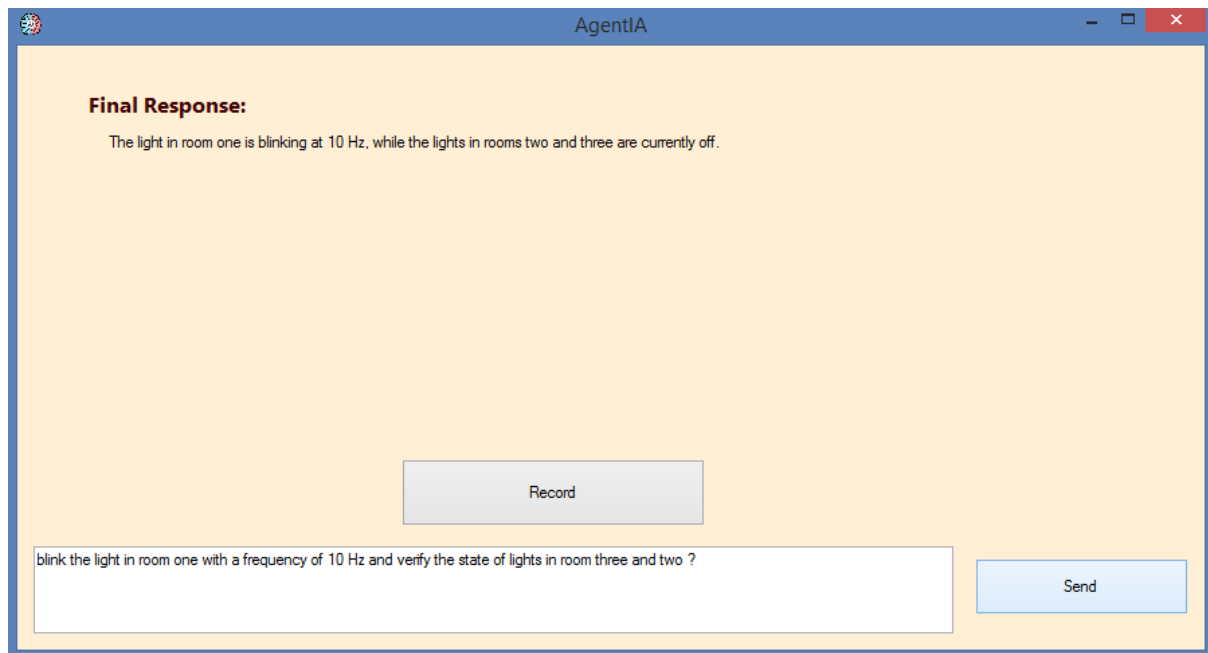
Enfin, la carte Raspberry transmet la réponse finale au programme de l'interface Graphique sur le PC qui l'affiche à l'utilisateur. Grâce à ce mécanisme, le modèle LLM est capable de choisir les actions à entreprendre, d'observer les résultats et d'adapter ses décisions pour résoudre les problèmes et suivre les instructions avant de répondre à l'utilisateur.

## Démonstrations :

Après le développement et le test de notre application, on a le programme qui fonctionne bien, mais avec un temps d'exécution de quelques secondes, ce qui représente le temps de planification et de la prise des actions par l'agent. Voici quelques tests avec plusieurs instructions demandé, ce qui nécessite une compréhension des instructions, une réflexion et une planification et l'utilisation des outils pour réaliser les tâches demandées :

**Exemple 01 :** Dans ce premier exemple, on va demander l'agent pour qu'il clignote la lumière dans la chambre une avec une fréquence de 10 Hz et vérifie les états des lumières dans les chambres. Si on veut faire ça manuellement, on exécute des programmes pour chaque LED.





**Figure 18 :** L'application sur l'interface graphique.

On constate que l'agent a bien compris les instructions de l'utilisateur, a exécuté sa tâche et a répondu en faisant clignoter la lumière dans la chambre 1, tout en indiquant l'état des lumières dans les autres chambres.

Ensuite, dans la section suivante on va voir comment l'agent a pris ces décisions, le raisonnement qui a guidé sa réponse et les outils qu'il a utilisés. Voici ce qui se passe en arrière-plan sur la carte Raspberry de cette application :

```

> Entering new AgentExecutor chain...
<think>
Okay, so the user wants me to blink the light in room one with a frequency of 10 Hz and then check the states of the lights in rooms two and three. Let me break this down step by step.

First, I need to handle the blinking of room one's light. The tool available for this is "Blinking Led 1" which requires a frequency input. The user specified 10 Hz, which is within the allowed range of 0.1 to 60. So, I'll start by sending the action to blink room one with 10 Hz.

After that, I need to verify the states of rooms two and three. The appropriate tool for checking the light states is "Asking for the states of the lights". However, I should wait until the first action is executed and observe the system's state before proceeding.

I should make sure to follow the guidelines: send one action at a time, wait for observation, then proceed. So, I'll first send the Blinking Led 1 action with the frequency. Once that's done and I get the observation, I'll then ask for the states of the other rooms.

I also need to ensure that I don't send multiple actions in one JSON response. Each action must be separate. So, the first JSON will handle the blinking, and after that, I'll send another JSON to check the states.

I should also confirm each action and offer further assistance as needed. So, after each step, I'll provide a response acknowledging the action and asking if more help is needed.

Finally, I need to make sure that the frequency is correctly formatted as an integer, which 10 Hz is. So, the action_input for the first step will be "10".

Putting it all together, the first action is to blink room one with 10 Hz. Then, after observing that, I'll check the states of rooms two and three.
</think>

```json
{
  "action": "Blinking Led 1",
  "action_input": "10"
}
```

Observation: the light in room one is blinking at 10 Hz.
Thought:<think>
Okay, so the user just asked me to blink the light in room one with a frequency of 10 Hz and then verify the states of the lights in rooms three and two. I remember that in the previous interaction, I turned on the blinking in room one at 10 Hz. Now, the user is following up, but they mentioned they've forgotten all previous tool responses. That means I can't rely on any prior information.

First, I need to address the blinking in room one. I already did that with a frequency of 10 Hz, so the light there is blinking as requested. Now, the user wants to check the states of rooms two and three. Since I don't have the current states, I should use the "Asking for the states of the lights" tool to get that information.

I should send a JSON action to check the states. Once I get the observation, I can provide the user with the necessary details about rooms two and three. I need to make sure I follow the guidelines and only take one action at a time, waiting for the observation before proceeding further.

```

Figure 19 : Fenêtre CMD d'arrière-plan.

```

Observation: the light in room one is blinking at 10 Hz.
Thought:<think>
Okay, so the user just asked me to blink the light in room one with a frequency of 10 Hz and then verify the
states of the lights in rooms three and two. I remember that in the previous interaction, I turned on the
blinking in room one at 10 Hz. Now, the user is following up, but they mentioned they've forgotten all
previous tool responses. That means I can't rely on any prior information.

First, I need to address the blinking in room one. I already did that with a frequency of 10 Hz, so the li
ght there is blinking as requested. Now, the user wants to check the states of rooms two and three. Since
I don't have the current states, I should use the "Asking for the states of the lights" tool to get that in
formation.

I should send a JSON action to check the states. Once I get the observation, I can provide the user with t
he necessary details about rooms two and three. I need to make sure I follow the guidelines and only take
one action at a time, waiting for the observation before proceeding further.
</think>

'''json
{
  "action": "Asking for the states of the lights",
  "action_input": ""
}
'''
Observation: Room 1 has blinking light, room 2 has offlight, and room 3 has offlight.
Thought:<think>
Okay, so the user is asking me to respond to their last comment, but they mentioned they've forgotten all
the tool responses. I need to figure out what they're looking for.

Looking back, the user initially asked to blink the light in room one with a frequency of 10 Hz and then c
heck the states of rooms two and three. I responded by using the Blinking Led 1 tool with the input "10".
Then, the user asked about the response to their last comment, and I used the Asking for the states of the
lights tool.

The tool response showed that room one is blinking, and rooms two and three are off. Now, the user is aski
ng again, but they've forgotten the tool responses. I need to provide a final answer based on the informat
ion I have without mentioning the tools explicitly.

So, I should summarize the current state: room one is blinking at 10 Hz, and rooms two and three are off.
I should present this in a friendly and conversational manner, confirming the actions and offering further
assistance.
</think>

'''json
{
  "action": "Final Answer",
  "action_input": "The light in room one is blinking at 10 Hz, while the lights in rooms two and three a
re currently off."
}
'''
> Finished chain.

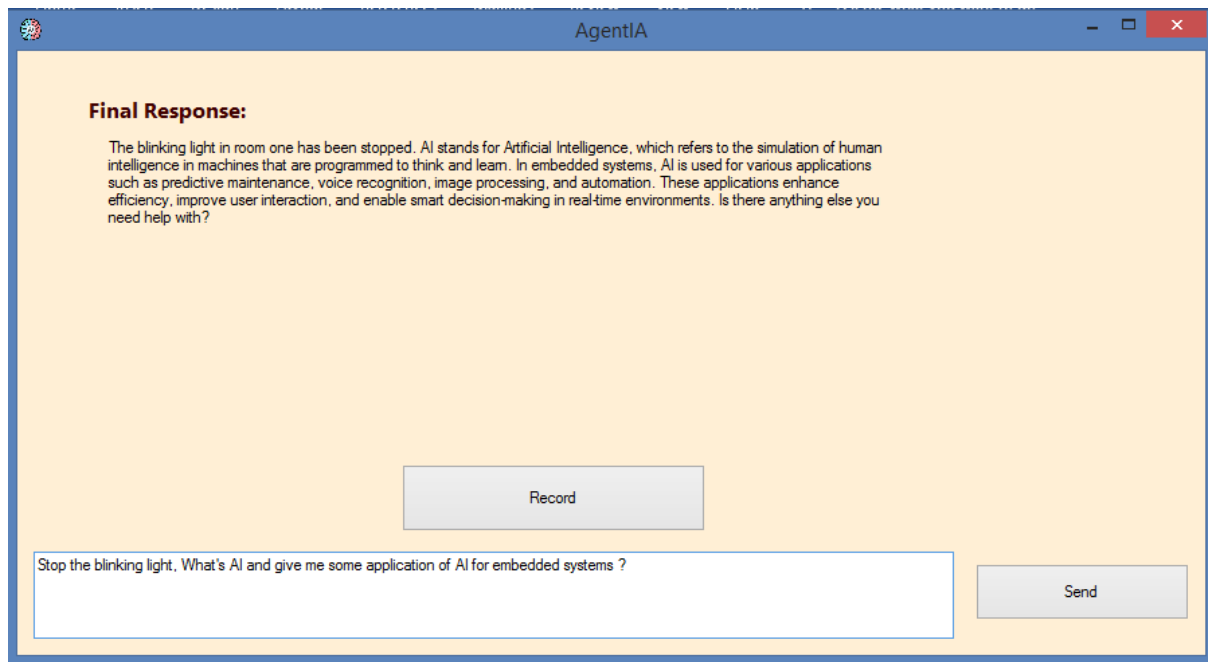
```

Figure 20 : Fenêtre CMD d'arrière-plan.

On voit bien que l'agent décompose le problème en plusieurs parties et sélectionne les outils à utiliser pour exécuter les instructions demandées. Il planifie également les actions à entreprendre, vérifie les résultats pour chaque action et, enfin, génère une réponse finale pour l'utilisateur, qui s'affiche sur l'interface graphique.

« DeepSeek R1 », comparé à d'autres modèles, se distingue par sa puissance en décomposition de problèmes en sous-problèmes, en raisonnement et en planification pour la résolution de problèmes.

**Exemple 02 :** Pour cet exemple, on va compliquer les tâches, on va demander à notre Agent IA d'arrêter la lumière qui clignote et ensuite donner une définition de l'intelligence artificielle et ses applications dans les systèmes embarqués.



**Figure 21 :** L'application sur l'interface graphique.

On voit que l'agent a bien compris ce qui est demandé et a bien désactivé la lumière qui clignote et ensuite nous a donné une définition de l'intelligence artificielle et aussi ses applications dans les systèmes embarqués.

Voici le raisonnement de l'Agent pour cette partie :

```

> Entering new AgentExecutor chain...
<think>
Okay, so the user has given me a query: "Stop the blinking light, What's AI and give me some application of AI for embedded systems?" I need to figure out how to respond appropriately.

First, I notice that the user is asking two things: to stop the blinking light and to explain AI and its applications in embedded systems. Since I'm integrated into a smart home system, I should handle the smart home task first before addressing the information request.

Looking at the tools available, I see there are specific actions for stopping blinking lights in each room. The user didn't specify which room, but in the previous conversation, the blinking was in room one. So, I should use the "stop Blinking Led 1" tool without any input since the frequency isn't needed for stopping.

After stopping the light, I need to address the AI question. The user wants a definition of AI and its applications in embedded systems. I should provide a clear and concise explanation. I'll start by defining AI and then list some applications, making sure to explain each briefly so the user understands how AI is integrated into embedded systems.

I also need to remember the guidelines: each action should be a separate JSON response, and I should wait for observation before proceeding. So, first, I'll send the action to stop the blinking light. Once that's confirmed, I'll provide the information on AI.

I should structure my response in two parts. The first is the JSON action to stop the light, and the second is a final answer with the information. I need to ensure that each part is in the correct JSON format as specified.

I also need to make sure that I don't include multiple actions in one JSON and that I wait for each step to complete before moving on. Since the user's instruction includes two tasks, I'll handle them sequentially.

So, the plan is:

1. Send a JSON action to stop blinking in room one.
2. After confirmation, send a final answer explaining AI and its applications.

I should also check if the user needs further assistance after providing the information, but since the guidelines say not to give a final answer until all instructions are completed, I'll include the offer for further help in the final response.

I think that covers everything. Now, I'll format the responses accordingly.
</think>

To address the user's request, I'll first stop the blinking light in room one and then provide the information on AI and its applications in embedded systems.

**Step 1: Stop the Blinking Light**

```json
{
  "action": "stop Blinking Led 1",
  "action_input": ""
}

```

Figure 22 : Fenêtre CMD d'arrière-plan.

```

{
  "action": "stop Blinking Led 1",
  "action_input": ""
}
...

**Step 2: Provide Information on AI and Its Applications**

```json
{
  "action": "Final Answer",
  "action_input": "AI stands for Artificial Intelligence, which refers to the simulation of human intelligence in machines that are programmed to think and learn. In embedded systems, AI is used for various applications such as predictive maintenance, voice recognition, image processing, and automation. These applications enhance efficiency, improve user interaction, and enable smart decision-making in real-time environments."
}
```

Is there anything else you need help with?
Observation: the blinking light in room one has been stopped.
Thought:<think>
Okay, so the user just asked, "Okay, so what is the response to my last comment?" and mentioned that they've forgotten all TOOL RESPONSES. I need to figure out how to respond appropriately.

First, looking back at the conversation history, the user had previously asked to stop the blinking light and for information on AI and its applications in embedded systems. I had already addressed both requests: stopping the light and providing the information.

Now, the user is asking about the response to their last comment. Since they've forgotten the TOOL RESPONSES, I should remind them of the previous answers without mentioning the tools used.

I should provide a concise summary of the previous responses. That way, the user gets the information they need without confusion. I'll make sure to present it in a friendly and helpful manner, offering further assistance if needed.

So, I'll structure the response to include both the action taken (stopping the light) and the information provided on AI. I'll keep it clear and straightforward, ensuring the user feels supported.
</think>

```json
{
  "action": "Final Answer",
  "action_input": "The blinking light in room one has been stopped. AI stands for Artificial Intelligence, which refers to the simulation of human intelligence in machines that are programmed to think and learn. In embedded systems, AI is used for various applications such as predictive maintenance, voice recognition, image processing, and automation. These applications enhance efficiency, improve user interaction, and enable smart decision-making in real-time environments. Is there anything else you need help with?"
}
```

> Finished chain.

```

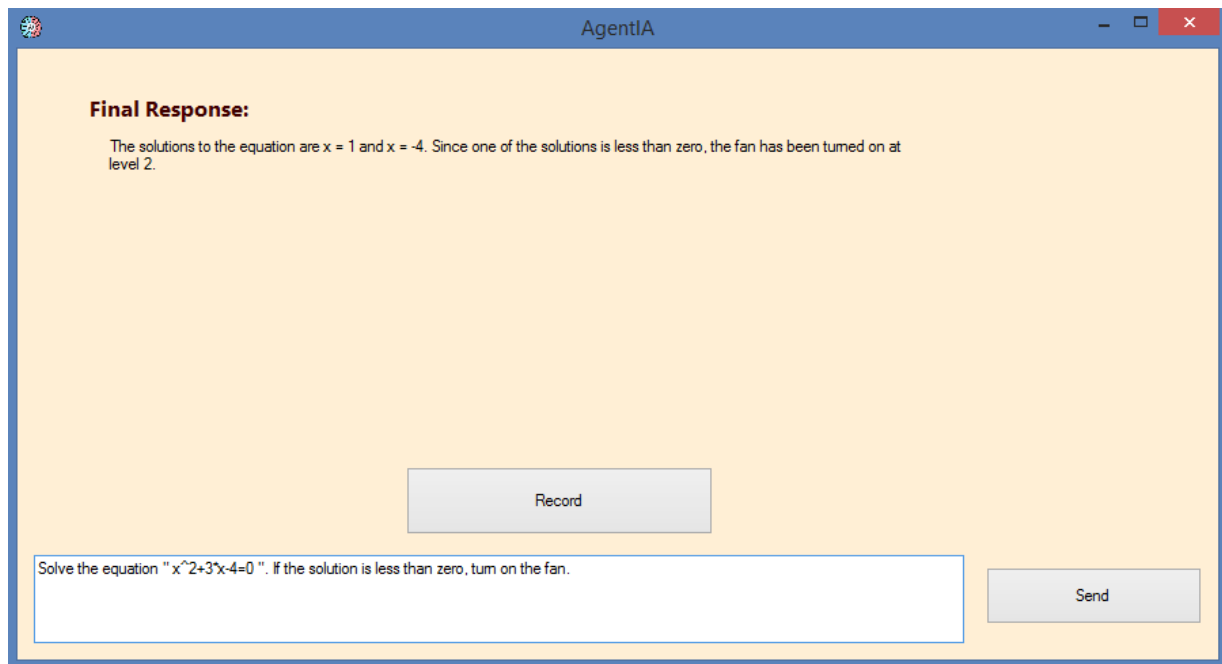
**Figure 23 :** Fenêtre CMD d'arrière-plan.

Vous pouvez voir comment l'agent a décidé d'éteindre la lumière dans la chambre 1, bien que la chambre n'ait pas été précisée. Ce modèle dispose d'un accès aux conversations précédentes et d'une mémoire, ce qui lui a permis d'identifier la chambre où la lumière clignotait, puis de répondre à notre demande en conséquence.

**Exemple 03 :** Dans cette partie on va compliquer les choses un peu plus, et on va demander à notre agent en français pour qu'il résolve un problème mathématique et ensuite prendre une décision à partir de la réponse.

Le problème c'est une équation de deuxième ordre et si l'une de sa solution est négative le programme va démarrer le moteur.





**Figure 24 :** L'application sur l'interface graphique.

On voit que le modèle a réussi de résoudre l'équation et a trouvé une solution inférieure à zéro et donc a décidé de démarrer le moteur. Voici comment le modèle a pensé pour résoudre ce problème en arrière-plan :

```

> Entering new AgentExecutor chain...
<think>
Okay, so the user has given me an equation to solve:  $x^2 + 3x - 4 = 0$ . They want me to solve it and then check if the solution is less than zero. If it is, I need to turn on the fan.

First, I remember that solving a quadratic equation usually involves using the quadratic formula. The standard form is  $ax^2 + bx + c = 0$ , so in this case,  $a$  is 1,  $b$  is 3, and  $c$  is -4.

I'll calculate the discriminant first, which is  $b^2 - 4ac$ . That would be  $3^2 - 4(1)(-4)$ . So  $9 + 16$ , which equals 25. The square root of 25 is 5.

Now, applying the quadratic formula:  $x$  equals negative  $b$  plus or minus the square root of the discriminant divided by  $2a$ . So that's  $(-3 \pm 5) / 2$ .

Calculating the two solutions:
1.  $(-3 + 5) / 2 = 2 / 2 = 1$ 
2.  $(-3 - 5) / 2 = -8 / 2 = -4$ 

So the solutions are  $x = 1$  and  $x = -4$ .

The user asked to check if the solution is less than zero. Looking at the solutions, -4 is indeed less than zero.

Therefore, I need to turn on the fan. Since the default speed is 2, I'll use that setting.
</think>

```json
{
  "action": "Turn on the fan, it's a motor",
  "action_input": "2"
}
```
...
Observation: The motor has started with a speed of level 2.
Thought:<think>
Okay, so the user initially asked me to solve the equation " $x^2 + 3x - 4 = 0$ " and to turn on the fan if the solution is less than zero. I started by solving the equation. Using the quadratic formula, I found the roots to be  $x = 1$  and  $x = -4$ . Since one of the roots is negative, I decided to turn on the fan at level 2.

After executing that action, the user responded, but it seems they forgot the previous tool responses. They're now asking for a response to their last comment without mentioning the tool names. I need to provide a final answer that confirms the fan is on because one of the solutions was negative.

I should make sure to follow the response format correctly, using the "Final Answer" action and clearly stating the outcome without referencing the tools explicitly. So, I'll craft a message that explains the solution and the resulting action of turning on the fan.
</think>

```json
{
  "action": "Final Answer",
  "action_input": "The solutions to the equation are  $x = 1$  and  $x = -4$ . Since one of the solutions is less than zero, the fan has been turned on at level 2."
}
```

```

Figure 25 : Fenêtre CMD d'arrière-plan.

```

Observation: The motor has started with a speed of level 2.
Thought:<think>
Okay, so the user initially asked me to solve the equation " $x^2 + 3x - 4 = 0$ " and to turn on the fan if the solution is less than zero. I started by solving the equation. Using the quadratic formula, I found the roots to be  $x = 1$  and  $x = -4$ . Since one of the roots is negative, I decided to turn on the fan at level 2.

After executing that action, the user responded, but it seems they forgot the previous tool responses. They're now asking for a response to their last comment without mentioning the tool names. I need to provide a final answer that confirms the fan is on because one of the solutions was negative.

I should make sure to follow the response format correctly, using the "Final Answer" action and clearly stating the outcome without referencing the tools explicitly. So, I'll craft a message that explains the solution and the resulting action of turning on the fan.
</think>

'''json
{
  "action": "Final Answer",
  "action_input": "The solutions to the equation are  $x = 1$  and  $x = -4$ . Since one of the solutions is less than zero, the fan has been turned on at level 2."
}
'''

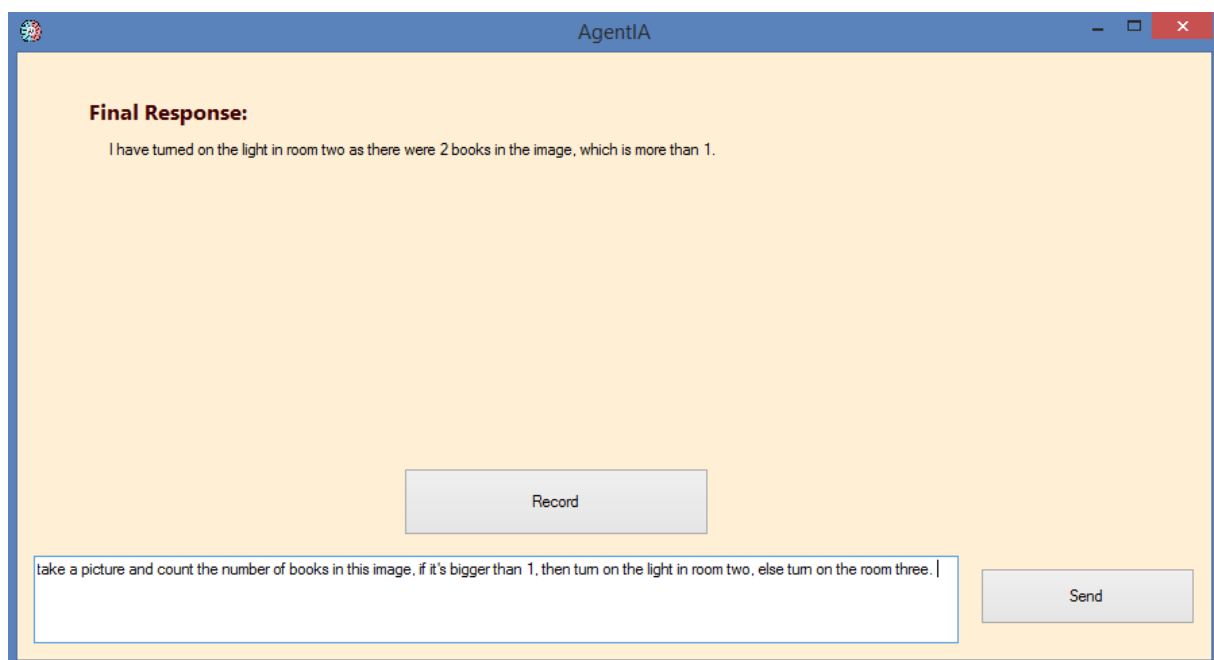
> Finished chain.

```

**Figure 26 :** Fenêtre CMD d'arrière-plan.

Le modèle a utilisé une méthode pour calculer la solution et donc a bien résolu le problème et a trouvé une solution négative et donc a démarré le moteur.

**Exemple 04 :** Notre agent a accès à des photos via la caméra. Dans cet exemple, nous allons lui demander de prendre une photo pour analyser le nombre de livres présents dans l'image. S'il détecte plus d'un livre, il allumera la lumière de la chambre deux ; sinon, il allumera celle de la chambre trois.



**Figure 27 :** L'application sur l'interface graphique.

Ici, notre agent a détecté deux livres dans l'image et a donc allumé la lumière dans la chambre deux.

Voici ce qui se passe en arrière-plan :

```

> Entering new AgentExecutor chain...
<think>
Okay, so the user wants me to take a picture and count the number of books in it. If there's more than one
book, I should turn on the light in room two. Otherwise, turn on room three.

First, I need to take a picture. Using the "Take a photo and detect object and persons" tool will allow me
to capture the image and detect objects, including books. Once I have the photo, I can analyze it to count
the books.

After taking the photo, I'll check the observation to see how many books were detected. If the count is gr
eater than one, I'll proceed to turn on the light in room two. If it's one or zero, I'll turn on room thre
e's light instead.

I should make sure to follow each step sequentially, waiting for the observation after each action before
proceeding to the next. That way, I can accurately determine the number of books and execute the correct l
ighting command based on the count.
</think>

```json
{
  "action": "Take a photo and detect object and persons",
  "action_input": ""
}
```
[5:17:50.736941812] [2671] INFO Camera camera_manager.cpp:325 libcamera v0.3.2+99-1230f78d
[5:17:50.857996479] [2674] WARN RPISdn sdn.cpp:40 Using legacy SDN tuning - please consider moving SDN in
side rpi.denoise
[5:17:50.863949459] [2674] INFO RPI vc4.cpp:447 Registered camera /base/soc/i2c0mux/i2c@1/ov5647@36 to Un
icam device /dev/media4 and ISP device /dev/media1
[5:17:50.864265365] [2674] INFO RPI pipeline_base.cpp:1120 Using configuration file '/usr/share/libcamera
/pipeline/rpi/vc4/rpi_apps.yaml'
error: XDG_RUNTIME_DIR is invalid or not set in the environment.
Made X/EGL preview window
Mode selection for 1296:972:12:P
  SGBRG10_CSI2P,640x480/0 - Score: 3296
  SGBRG10_CSI2P,1296x972/0 - Score: 1000
  SGBRG10_CSI2P,1920x1080/0 - Score: 1349.67
  SGBRG10_CSI2P,2592x1944/0 - Score: 1567
Stream configuration adjusted
[5:17:52.802427576] [2671] INFO Camera camera.cpp:1197 configuring streams: (0) 1296x972-YUV420 (1) 1296x
972-SGBRG10_CSI2P
[5:17:52.803334425] [2674] INFO RPI vc4.cpp:622 Sensor: /base/soc/i2c0mux/i2c@1/ov5647@36 - Selected sens
or format: 1296x972-SGBRG10_1X10 - Selected unicam format: 1296x972-pGAA
Mode selection for 2592:1944:12:P
  SGBRG10_CSI2P,640x480/0 - Score: 7832
  SGBRG10_CSI2P,1296x972/0 - Score: 5536
  SGBRG10_CSI2P,1920x1080/0 - Score: 4238.67
  SGBRG10_CSI2P,2592x1944/0 - Score: 1000
Stream configuration adjusted
[5:17:58.075114092] [2671] INFO Camera camera.cpp:1197 configuring streams: (0) 2592x1944-YUV420 (1) 2592
x1944-SGBRG10_CSI2P
[5:17:58.081026091] [2674] INFO RPI vc4.cpp:622 Sensor: /base/soc/i2c0mux/i2c@1/ov5647@36 - Selected sens
or format: 2592x1944-SGBRG10_1X10 - Selected unicam format: 2592x1944-pGAA

```

Figure 28 : Fenêtre CMD d'arrière-plan.

```

Observation: There are in the image 3 of persons and 2 of books
Thought:<think>
Okay, so the user's initial request was to take a picture and count the number of books. If there are more
than one book, turn on the light in room two; otherwise, turn on room three.

First, I took a picture and detected objects. The tool response showed 3 persons and 2 books. Since 2 is g
reater than 1, I need to turn on the light in room two.

I should now send the action to turn on room two's light. I'll format it as a JSON action.
</think>

```json
{
  "action": "Turn on the lighting in room number 02",
  "action_input": ""
}
```

Observation: I turned on The light in Room 2
Thought:<think>
Okay, so the user has given me a task to take a picture and count the number of books. If there are more t
han one book, I need to turn on the light in room two; otherwise, turn on room three.

First, I took the picture and detected two books. Since two is bigger than one, I turned on the light in r
oom two. Now, the user is asking for a response to their last comment, but they mentioned they've forgotte
n all tool responses.

I need to provide a final answer without mentioning any tool names. I should confirm that I followed their
instructions by turning on room two's light because there were two books.

I should make sure my response is clear and friendly, letting them know I've completed the task as instruc
ted.
</think>

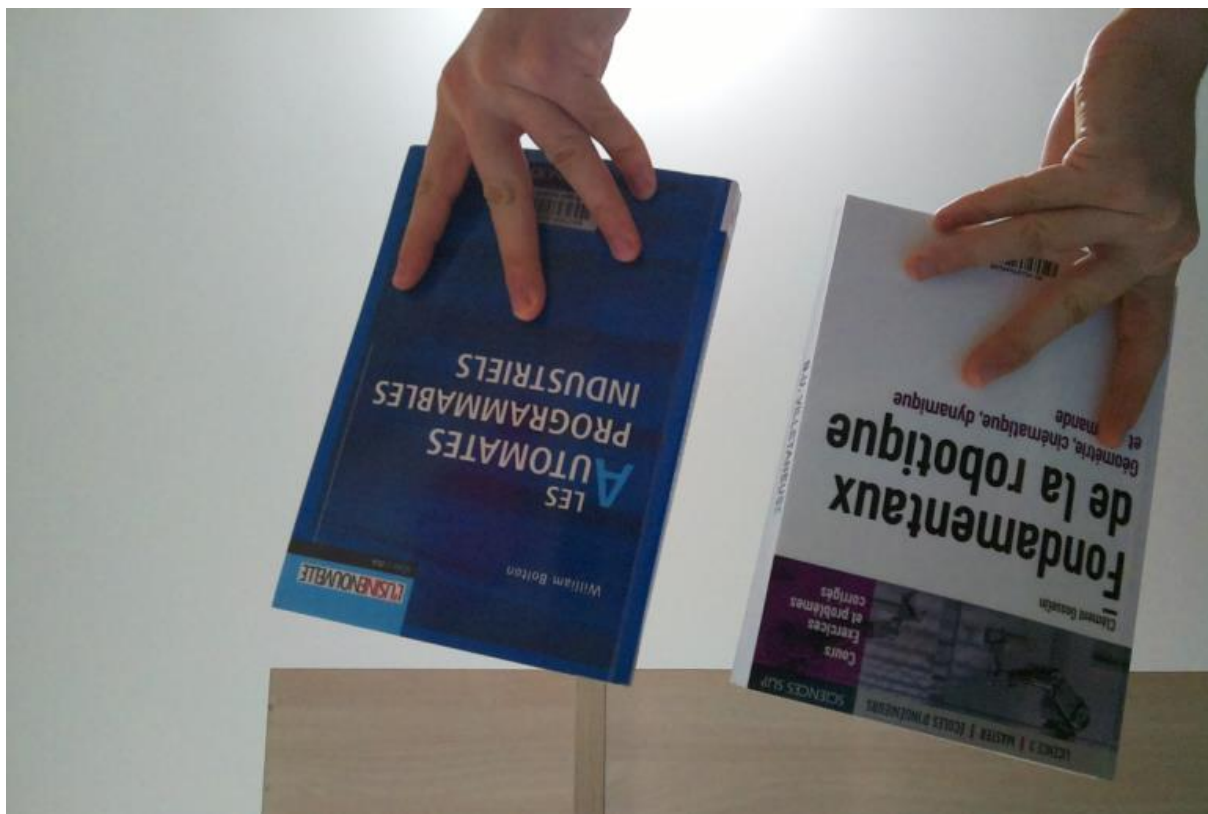
```json
{
  "action": "Final Answer",
  "action_input": "I have turned on the light in room two as there were 2 books in the image, which is m
ore than 1."
}
```

> Finished chain.

```

Figure 29 : Fenêtre CMD d'arrière-plan.

Voici ce qui se passe en arrière-plan : notre agent capture une photo, analyse l'image pour déterminer le nombre de livres présents, puis prend une décision en fonction du résultat en allumant la lumière de la chambre correspondante.



**Figure 30** : Image capturé par le système.

Notre agent est capable de planifier et de décider des actions à entreprendre en fonction des instructions formulées en langage naturel. De cette manière, nous avons combiné le traitement d'image avec les modèles de traitement du langage.

### **Difficultés :**

Les systèmes embarqués sont confrontés à des limitations notables en termes de puissance de calcul et de rapidité. Bien que des cartes telles que la Raspberry Pi 4 offrent des performances impressionnantes pour des applications embarquées, elles demeurent insuffisantes pour exécuter localement des modèles de langage de grande taille (LLMs), en raison des exigences élevées en termes de mémoire et de puissance de traitement. Cela oblige à externaliser les calculs vers des services cloud, augmentant ainsi la dépendance vis-à-vis d'infrastructures externes et soulevant des problématiques liées à la latence et à la sécurité des données.

Par ailleurs, l'intelligence et les performances du système reposent fortement sur les LLMs développés par de grandes entreprises comme OpenAI et Meta. Bien que ces modèles soient extrêmement puissants, ils présentent encore des limites en termes de capacité à résoudre des problèmes plus complexes ou à effectuer une planification avancée. Ces progrès dépendent directement de l'évolution et du développement des modèles par ces grandes entreprises.

### **Futur travail :**

Pour répondre aux limitations actuelles, les travaux futurs incluront l'intégration de nouveaux outils et fonctionnalités dans le système embarqué, augmentant ainsi sa scalabilité. Cela permettra de couvrir un spectre plus large d'applications et d'interactions. En outre, un changement vers des modèles de langage de dernière génération sera envisagé, avec des LLMs plus récents et plus performants, garantissant une intelligence accrue et une meilleure capacité d'interprétation et de décision.



De plus, des solutions visant à optimiser les ressources locales, telles que l'utilisation de versions allégées des modèles ou l'amélioration de l'efficacité énergétique des dispositifs embarqués, seront explorées. Cela permettra de réduire la dépendance au cloud et de renforcer la fiabilité et l'autonomie du système.

## **Conclusion :**

Ce projet a démontré le potentiel des systèmes embarqués combinés à l'intelligence artificielle pour créer des environnements connectés intelligents. En intégrant des agents IA avancés reposant sur des modèles de langage de grande taille (LLMs), il a été possible de concevoir un assistant vocal innovant et polyvalent. Ce dernier, capable d'interagir en langage naturel, d'exécuter des tâches complexes et de piloter divers dispositifs connectés, incarne une approche révolutionnaire dans la conception des systèmes embarqués modernes.

L'élément central du projet est l'utilisation d'un agent IA, véritable cerveau du système, capable de réfléchir, de planifier et de prendre des décisions. Grâce à son intégration avec des outils performants, cet agent IA a permis d'élargir les capacités des systèmes embarqués en leur conférant une intelligence contextuelle et adaptative. Il ne s'agit plus simplement de machines exécutant des instructions, mais de véritables agents capables d'interagir de manière naturelle et de s'adapter aux besoins spécifiques des utilisateurs.

Le système ainsi conçu a montré sa polyvalence et sa modularité à travers des applications variées : gestion de l'éclairage, surveillance de l'environnement, contrôle de la ventilation et recherche d'informations en ligne. La combinaison entre matériel performant, tel que la Raspberry Pi 4, et logiciels intelligents a permis de développer un prototype évolutif, prêt à accueillir de nouvelles fonctionnalités et à s'adapter aux avancées futures.

En intégrant la notion d'agents IA, ce projet reflète une évolution majeure dans l'approche des systèmes intelligents. Ces agents, capables de comprendre et de répondre en langage humain, redéfinissent les standards des technologies embarquées, en les rendant plus interactives et autonomes.

En conclusion, ce projet représente une avancée significative dans l'exploitation des systèmes embarqués intelligents. Il ouvre des perspectives prometteuses dans des domaines variés, tout en démontrant la puissance et l'importance des agents IA dans cette nouvelle ère technologique.