



ÉCOLE NATIONALE SUPÉRIEURE DE  
L'ÉLECTRONIQUE ET DE SES APPLICATIONS

PROJET 2A

---

Outil de détection des pleurs d'un bébé

---

*Élèves :*

MHAMDI AMENI

WERHENI HEDIA

MAYIMONA ZANDU MERVEILLE

*Enseignant :*

JEBRI AYOUB

8 avril 2024

# Table des matières

<b>1</b>	<b>INTRODUCTION</b>	<b>4</b>
<b>2</b>	<b>GESTION DE PROJET</b>	<b>5</b>
2.1	Cahier des Charges . . . . .	5
2.2	Diagramme de GANTT . . . . .	6
<b>3</b>	<b>ARCHITECTURE DU SYSTÈME</b>	<b>8</b>
3.1	Diagramme de Classes UML . . . . .	8
3.2	Diagramme de Séquence UML . . . . .	9
3.3	Diagramme d'État UML . . . . .	10
3.4	Diagramme de Cas d'Utilisation UML . . . . .	11
<b>4</b>	<b>SOLUTIONS MATÉRIELLES</b>	<b>12</b>
4.1	Microphone . . . . .	12
4.2	Haut-Parleur . . . . .	13
4.3	Microcontrôleur . . . . .	13
<b>5</b>	<b>SOLUTIONS LOGICIELLES</b>	<b>15</b>
5.1	Environnement Logiciel « Visual Studio Code » . . . . .	15
5.2	Langages de Programmation et Bibliothèques Utilisées . . . . .	15
5.2.1	Python . . . . .	15
5.2.2	Bibliothèques Keras . . . . .	15
5.2.3	Bibliothèques Sounddevice et Librosa . . . . .	15
<b>6</b>	<b>RÉSEAUX DE NEURONNES</b>	<b>16</b>
6.1	Fondements Théoriques . . . . .	16
6.2	Architecture et Entraînement du Réseau de Neurones . . . . .	16
<b>7</b>	<b>APPLICATION : DÉTECTION DE PLEURS</b>	<b>18</b>
7.1	Détection du Son en Temps Réel . . . . .	18
7.2	Caractéristiques Audio et Prétraitement . . . . .	18
7.3	Détection de Pleurs de Bébé en Temps Réel . . . . .	19
<b>8</b>	<b>Amélioration de l'apprentissage automatique</b>	<b>20</b>
<b>9</b>	<b>Mise en œuvre sur Raspberry Pi 3</b>	<b>21</b>
9.1	Configuration d'un environnement virtuel . . . . .	22
9.2	Activation des GPIO pour le haut-parleur . . . . .	23
<b>10</b>	<b>BILAN DU PROJET</b>	<b>24</b>
<b>11</b>	<b>BIBLIOGRAPHIE</b>	<b>25</b>

## Table des figures

1	Cahier des charges . . . . .	5
2	Diagramme de Gantt initial . . . . .	6
3	Diagramme de Gantt réalisé . . . . .	7
4	Diagramme de classe . . . . .	8
5	Diagramme de séquence . . . . .	9
6	Diagramme d'état . . . . .	10
7	Diagramme de cas d'utilisation . . . . .	11
8	Choix du microphone . . . . .	12
9	Choix du haut-parleur . . . . .	13
10	Choix du microcontrôleur . . . . .	13
11	RaspberryPi 3 . . . . .	14
12	Réseau de Neurones . . . . .	16
13	Résultat de l'entraînement . . . . .	17
14	Nouveau résultat de l'entraînement . . . . .	20
15	Fenetre d'installation Raspberry Pi . . . . .	21
16	Activation de l'environnement virtuel . . . . .	22
17	Activation des GPIO . . . . .	23
18	Activation des GPIO . . . . .	24

## Table des codes

1	Entrainement . . . . .	17
2	Détection . . . . .	18
3	Caracteristiques audio . . . . .	18
4	Test . . . . .	19
5	Nouveau caracteristiques audio . . . . .	20
6	Commande d'installation et d'initialisation de l'environnement virtuel . . .	22
7	Pin utilisé . . . . .	23

# 1 INTRODUCTION

Le domaine de la surveillance a connu des avancées significatives avec l'intégration de technologies modernes. L'un de ces progrès est représenté par notre projet axé sur un "Outil de détection des pleurs d'un bébé". Cette solution novatrice vise à offrir une réponse efficace aux besoins des parents en matière de surveillance et de réconfort pour leurs nourrissons.

Le système repose sur une architecture clé, mettant en œuvre des composants tels qu'un microphone pour l'enregistrement du son, un réseau de neurones pour l'analyse des données audio, et un haut-parleur pour apaiser le bébé en jouant une berceuse. L'utilisation d'une approche de réseau de neurones démontre l'application des techniques d'intelligence artificielle dans un contexte quotidien.

L' "Outil de détection des pleurs d'un bébé" capture les pleurs du bébé à travers l'utilisation d'un microphone. Les données audio ainsi obtenues sont ensuite analysées par un réseau de neurones, permettant de détecter la présence de pleurs. En cas de détection, le système active un haut-parleur qui diffuse une berceuse, apportant ainsi un réconfort auditif au bébé.

Pour comprendre plus en détail les aspects techniques du projet, nous explorerons les choix matériels du côté du microphone, la conception du système mécanique, et le développement de l'interface utilisateur permettant la visualisation des données issues du microphone. Ces éléments seront discutés respectivement dans les chapitres suivants de ce rapport.

En résumé, notre projet s'inscrit dans le contexte de la technologie au service de la parentalité, en offrant une solution automatisée et intelligente pour la détection et la réponse aux pleurs des bébés. Ce rapport détaillera chaque étape du développement, du choix des composants à la mise en œuvre de l'interface utilisateur, contribuant ainsi à l'amélioration de la qualité de vie des parents et de leurs nourrissons.

## 2 GESTION DE PROJET

### 2.1 Cahier des Charges

Le Cahier des Charges (CdC) constitue un document essentiel définissant les objectifs, les contraintes et les spécifications du projet "Outil de détection des pleurs d'un bébé". Il sert de référence pour l'ensemble de l'équipe et assure une compréhension commune des attentes et des exigences du projet.

Fonction	Expression	Critères	Niveaux d'exigence	Moyen de contrôle
FP1	Ecouter un son	Il faut pouvoir capturer tous les sons dans une chambre	F0	Micro
FP2	Émettre un son	Il faut pouvoir émettre un son régulier	F0	Haut parleur
FP3	Permettre de détecter les pleurs d'un bébé	Il faut une base de donnée précise	F0	Base de donnée
FC2	Déclencher une berceuse	Il faut que le déclenchement de la musique se fasse assez vite après les premiers instants de pleure du bébé	F0	Code
FC3	Ne doit pas être dangereux pour le bébé	Il faut que la musique ne soit pas trop forte pour le bébé et que l'appareil émet un minimum d'onde	F1	Choix des matériaux et composants Limite du seuil de danger auditif pour un bébé
FC4	Doit avoir un coût raisonnable	Il faut respecter le budget imposé	F2	Nomenclature
FC5	Doit être de taille et poids réduits	Il faut utiliser des composants de tailles réduites pour que l'appareil puisse être tenu dans la main	F3	Choix des composants et matériaux
FC6	Doit s'adapter aux goût des usagers	Il faut avoir la possibilité de pouvoir changer la musique et que le système ne fasse pas peur au bébé	F4	Choix des matériaux

FIGURE 1 – Cahier des charges

## 2.2 Diagramme de GANTT



FIGURE 2 – Diagramme de Gantt initial

Le planning initial du projet est illustré dans le diagramme de Gantt provisoire ci-dessus. Il offre une vision globale des diverses tâches et de leur programmation dans le temps. Néanmoins, comme c'est le cas pour tout projet, des modifications pourraient être requises au cours de son déroulement.



FIGURE 3 – Diagramme de Gantt réalisé

Le schéma de Gantt ultime présenté ci-dessus illustre la planification actualisée du projet, tenant compte des ajustements et des évolutions qui ont émergé au fil du temps. Cette version définitive constitue une représentation considérablement affinée de la distribution temporelle des diverses phases du projet. En examinant attentivement ce diagramme, il devient évident que chaque étape a été soigneusement élaborée, prenant en considération les défis rencontrés et les modifications apportées pour garantir une exécution optimale. La précision accrue de cette représentation temporelle offre une vision approfondie de la progression du projet, soulignant la manière dont les ajustements stratégiques ont été intégrés de manière cohérente. Ce schéma revêt une importance cruciale dans la documentation du parcours du projet et permettra d'illustrer de manière convaincante les processus d'adaptation et d'optimisation continus qui ont contribué au succès de l'ensemble du projet.



## 3 ARCHITECTURE DU SYSTÈME

### 3.1 Diagramme de Classes UML

Le diagramme de classes ci-dessous offre une représentation statique des classes et des interfaces du système, ainsi que des relations qui existent entre elles.

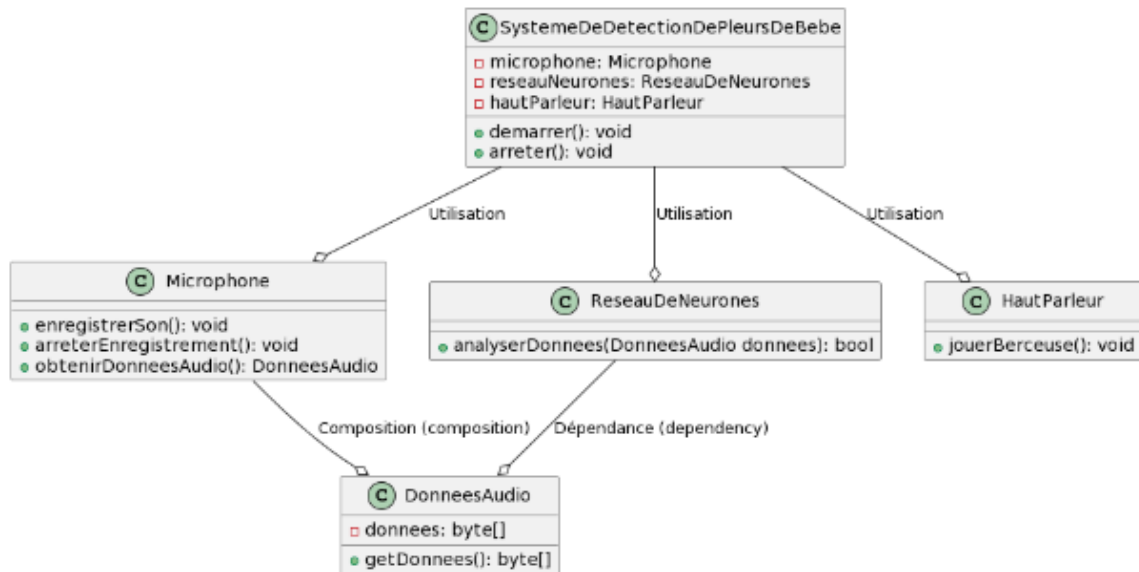


FIGURE 4 – Diagramme de classe

Ce diagramme permet de visualiser la structure du système, notamment les différentes entités et leurs interactions. Les classes représentent les objets du système, tandis que les associations définissent les relations entre ces objets.

### 3.2 Diagramme de Séquence UML

Le diagramme de séquence vise à représenter, dans un ordre chronologique, les échanges et les interactions entre les divers acteurs du système.

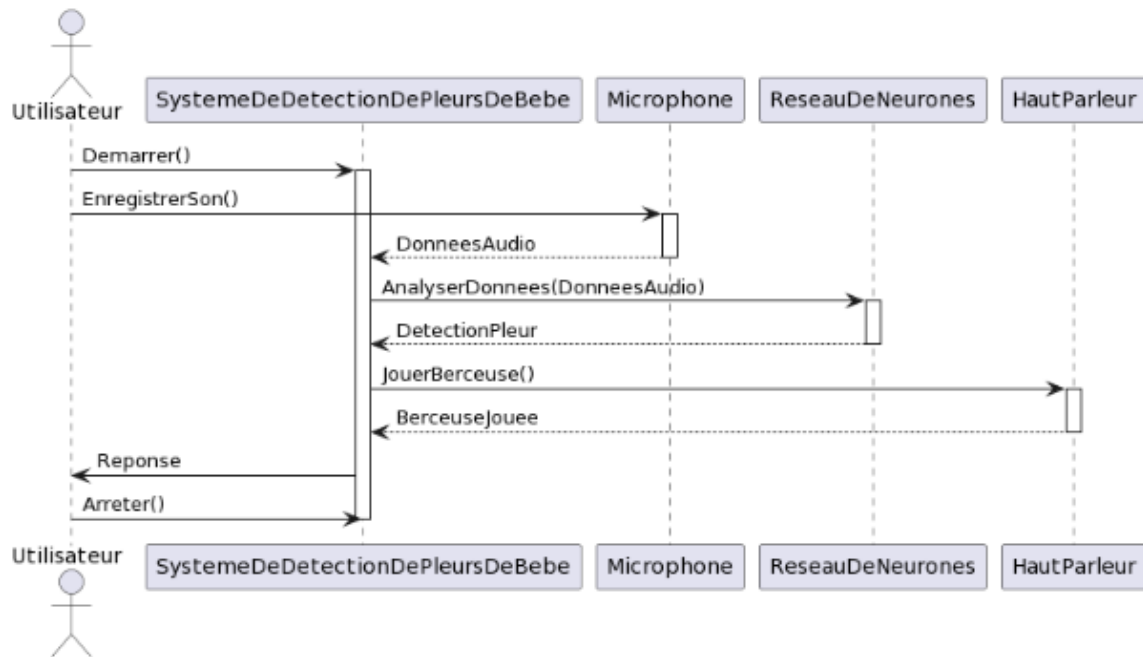


FIGURE 5 – Diagramme de séquence

Ce schéma permet de mieux comprendre le flux d'exécution des différentes fonctionnalités du système, mettant en évidence les messages échangés entre les objets ou acteurs.

### 3.3 Diagramme d'État UML

Le diagramme d'état suivant illustre les différents états qu'un objet peut occuper au cours de son cycle de vie.

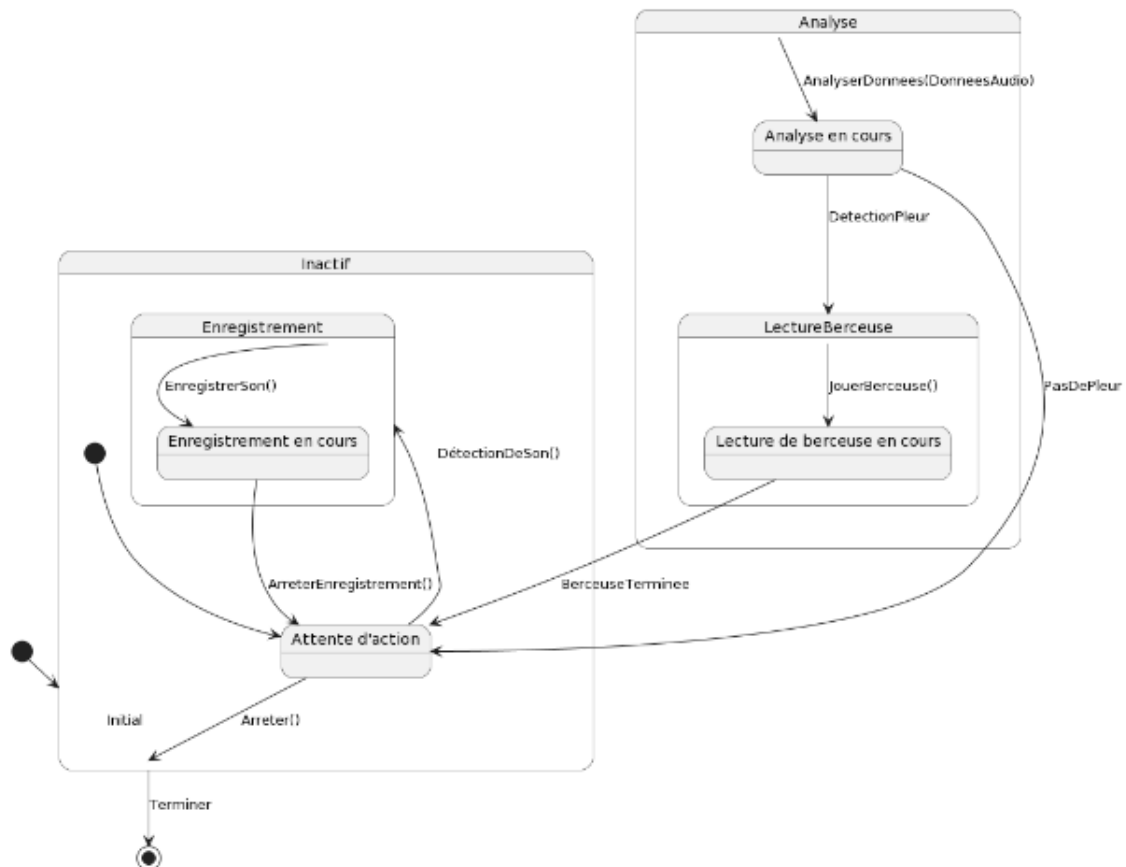


FIGURE 6 – Diagramme d'état

Cela permet d'observer de manière dynamique comment les objets du système évoluent en réaction aux influences externes, offrant ainsi un aperçu du comportement en constante adaptation de ces objets.

### 3.4 Diagramme de Cas d'Utilisation UML

Le diagramme de cas d'utilisation ci-dessous permet de recueillir, d'analyser et d'organiser les besoins du système en présentant les principales fonctionnalités sous forme de cas d'utilisation.

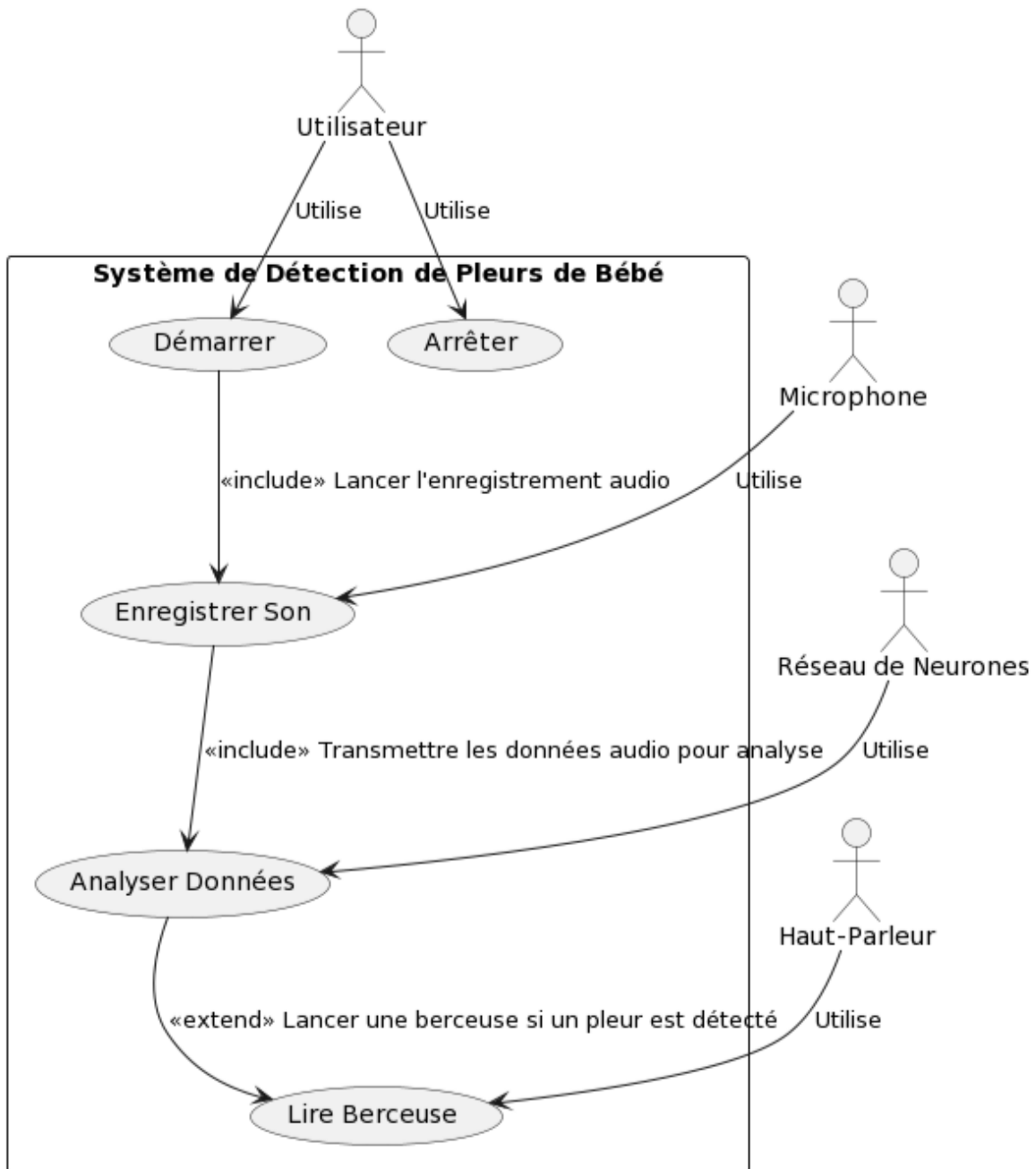


FIGURE 7 – Diagramme de cas d'utilisation

Ce diagramme offre une vision globale des interactions entre les acteurs (utilisateurs ou systèmes externes) et le système, identifiant ainsi les différentes façons dont les utilisateurs interagissent avec ce dernier.

## 4 SOLUTIONS MATÉRIELLES

### 4.1 Microphone

Il faut utiliser un microphone compatible avec la carte Raspberry Pi.

<a href="#">microphone</a>	<a href="#">prix</a>	<a href="#">photos</a>
Mini microphone portable à clipser	1.03 euros	
VIEWLON Micro-Cravate sans Fil	34 euros	
Mini microphone USB Raspberry Pi	4.45 euros	

FIGURE 8 – Choix du microphone

Nous avons choisi la Mini microphone USB Raspberry pour des raisons multiples grâce à sa petite taille (peut être intégré facilement) en plus il est la plus simple à mettre (pilote gratuit et Free Driver).

- Caractéristique du Microphone :
- Sensibilité :  $-67\text{dB} \pm 4\text{dB}$
- Impédance :  $\leq 2.2\text{K}\Omega$
- Tension locale : 4.5V
- Rapport S/N : Plus de 67dB
- Réponse en fréquence : 100 16kHz

## 4.2 Haut-Parleur

Il faut utiliser un haut-parleur compatible avec la carte Raspberry Pi.




Haut-parleur	prix	photos
Module Speaker pHAT PIM254	15.99 euros	
Module HP amplifié SKU00076	10.60 euros	
Module ReSpeaker linéaire 107990056	32.60 euros	

FIGURE 9 – Choix du haut-parleur

Nous avons choisi le haut-parleur mode HP amplifié SKU00076 car c'est le moins cher et ses émissions sonores sont raisonnable pour l'environnement et la sécurité du bébé.

Caractéristique de l'émetteur :

- Alimentation : 5 Vcc via la carte Raspberry Pi
- Consommation maxi : 500 mA
- Impédance HP :  $8\Omega$
- Puissance : 500 mW
- Dimensions : 34 x 22 x 12 mm

## 4.3 Microcontrôleur

Nom de la carte	Arduino Nano	ESP32	Raspberry Pi 3 B
MCU	Atmel ATmega168 ou ATmega328	Xtensa Dual-Core 32 bits LX6 D	ARM 7100
Fréquence d'horloge	16 MHz	240 MHz	1.2GHz
Pins d'E/S	20 GPIO	30 GPIO	40 GPIO

FIGURE 10 – Choix du microcontrôleur

Nous avons choisi la carte Raspberry Pi 3 B vu la disponibilité de cette carte dans les labos, en plus il s'agit d'une carte facilement programmable dispose de nombre important des GPIO. Ce module intègre également la connectivité Bluetooth 4.

Caractéristique du Microprocesseur :

- 1Go RAM
- 4 x USB ports
- Full size HDMI
- Alimentation : 5V 2A
- Mise à niveau de la source d'alimentation Micro USB commutée jusqu'à 2,5A

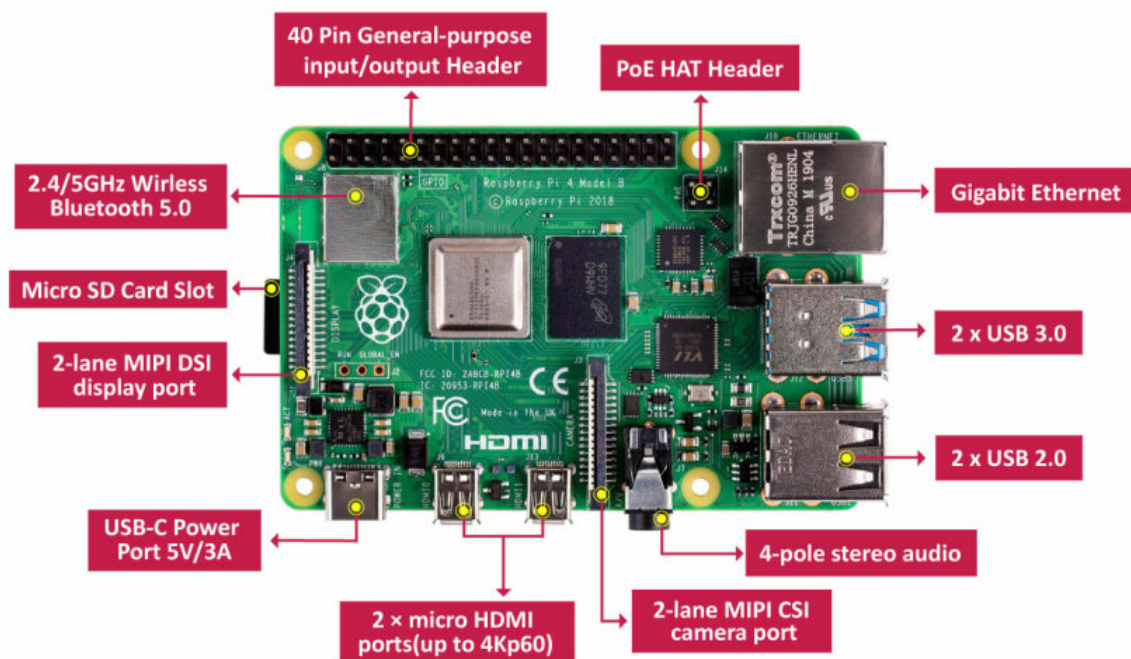


FIGURE 11 – RaspberryPi 3

## 5 SOLUTIONS LOGICIELLES

### 5.1 Environnement Logiciel « Visual Studio Code »

Visual Studio Code est un éditeur de code open source créé par Microsoft. Lancé en 2015, il s'est rapidement imposé comme l'un des éditeurs de code les plus populaires parmi les développeurs. Sa polyvalence et son extensibilité en font un choix prisé pour une variété de langages de programmation. Visual Studio Code propose des fonctionnalités avancées telles que la coloration syntaxique, l'achèvement automatique du code, le débogage intégré, la gestion de version, ainsi qu'une interface utilisateur épurée et personnalisable. Il prend en charge de nombreux langages de programmation et offre une multitude d'extensions pour personnaliser l'environnement de développement en fonction des besoins spécifiques de chaque développeur.

### 5.2 Langages de Programmation et Bibliothèques Utilisées

#### 5.2.1 Python

Python est un langage de programmation open source, interprété et généraliste. Il a été conçu par Guido van Rossum et a été introduit en 1991. Python se distingue par sa syntaxe claire et lisible, favorisant la lisibilité du code et la productivité des développeurs. Il prend en charge plusieurs paradigmes de programmation, notamment la programmation orientée objet, impérative et fonctionnelle. Python est polyvalent et largement utilisé dans divers domaines tels que le développement web, la science des données, l'intelligence artificielle et l'automatisation de tâches. Sa grande communauté, sa documentation exhaustive et sa bibliothèque standard étendue en font un choix populaire pour les programmeurs de tous niveaux.

#### 5.2.2 Bibliothèques Keras

Keras est une bibliothèque open source en Python dédiée à l'apprentissage en profondeur (deep learning). Elle sert de surcouche haut niveau pour simplifier la création, la formation et le déploiement de modèles de réseaux de neurones artificiels. Développée à l'origine par François Chollet, Keras se caractérise par une syntaxe simple et cohérente, facilitant la définition d'architectures de réseaux neuronaux. Elle est conçue pour être modulaire, extensible et compatible avec d'autres bibliothèques d'apprentissage profond, notamment TensorFlow.

#### 5.2.3 Bibliothèques Sounddevice et Librosa

Sounddevice est une bibliothèque open source en Python spécialisée dans l'accès et la gestion des périphériques audio. Elle fournit des fonctionnalités permettant de lire et d'écrire des données audio à partir et vers des périphériques tels que des microphones et des haut-parleurs.

Librosa est une bibliothèque open source en Python dédiée au traitement et à l'analyse de signaux audio. Conçue spécifiquement pour faciliter l'extraction d'informations pertinentes à partir de données audio, Librosa offre une surcouche haut niveau pour simplifier les tâches liées à l'analyse musicale, à l'extraction de caractéristiques audio et à d'autres opérations de traitement du son.



## 6 RÉSEAUX DE NEURONES

### 6.1 Fondements Théoriques

Les réseaux de neurones, inspirés du cerveau humain, résolvent des problèmes complexes en modélisant des relations non linéaires entre les données. Ces réseaux artificiels sont constitués de couches de neurones qui effectuent des calculs et transmettent des signaux en attribuant des poids aux connexions et en appliquant des fonctions d'activation.

Dans ce projet, un réseau de neurones est utilisé pour la classification binaire (pleurs vs. non-pleurs) à partir de caractéristiques extraites de segments audio, représentées par une matrice (128, 80). Le modèle exploite ces caractéristiques pour discerner les modèles sonores associés aux pleurs et aux non-pleurs.

### 6.2 Architecture et Entraînement du Réseau de Neurones

L'architecture du réseau de neurones dans ce projet est conçue pour exploiter les caractéristiques extraites des segments audio. La première couche, avec une forme d'entrée de (128, 80) représentant ces caractéristiques, est suivie d'une couche entièrement connectée de 64 neurones avec une activation ReLU. Cette disposition vise à capturer des motifs complexes. Pour prévenir le surajustement, une couche de dropout est introduite, suivie d'une deuxième couche entièrement connectée de 32 neurones avec une activation ReLU, renforçant ainsi la capacité du modèle à apprendre des représentations hiérarchiques. Une autre couche de dropout est ajoutée avant la couche de sortie, constituée d'un seul neurone avec une activation sigmoïde pour la classification binaire. [2]

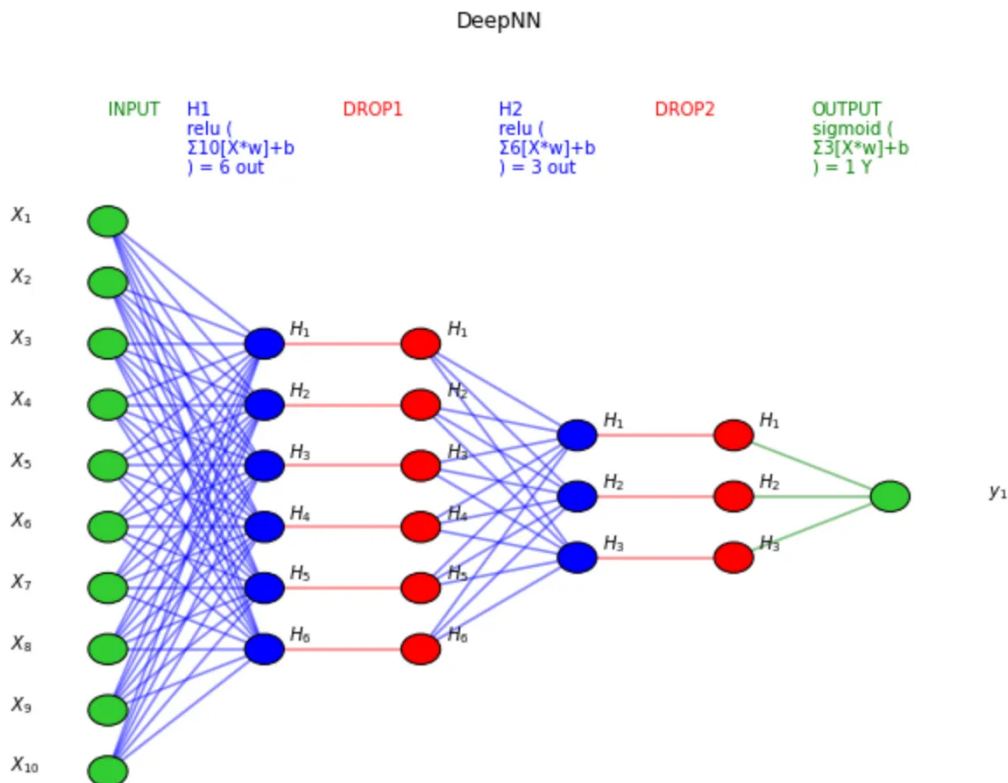


FIGURE 12 – Réseau de Neurones

Le processus d'entraînement se déroule sur les données d'entraînement, utilisant une fonction de perte binaire croisée et l'optimiseur Adam. Pendant 100 époques, le modèle apprend sur des lots de taille 32, et sa performance est évaluée régulièrement sur les données de validation pour surveiller son évolution.

```

1 X_train, X_test, y_train, y_test = train_test_split(donnees_segments,
    etiquettes_segments, test_size=0.2, random_state=42)
2
3 modele = Sequential()
4 modele.add(Flatten(input_shape=(128, 80)))
5 modele.add(Dense(units=64, activation='relu'))
6 modele.add(Dropout(0.3))
7 modele.add(Dense(units=32, activation='relu'))
8 modele.add(Dropout(0.3))
9 modele.add(Dense(units=1, activation='sigmoid'))
10
11
12 modele.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate
    =0.001), metrics=['accuracy'])
13
14 modele.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(
    X_test, y_test))
15
16 perte, precision = modele.evaluate(X_test, y_test)
17 print(f"Perte de test : {perte:.4f}, Precision de test : {precision:.4f}")

```

Code 1 – Entraînement

Une fois l'entraînement terminé, le modèle est évalué sur les données de test pour calculer la perte et la précision. Ces mesures offrent un aperçu de la capacité du modèle à généraliser aux données non vues, confirmant ainsi sa robustesse en dehors de l'ensemble d'entraînement.

```

Epoch 97/100
77/77 [=====] - 1s 17ms/step - loss: 0.1565 - accuracy: 0.9281 - val_loss: 5.5704 - val_accuracy: 0.7783
Epoch 98/100
77/77 [=====] - 1s 17ms/step - loss: 0.5157 - accuracy: 0.9294 - val_loss: 5.6515 - val_accuracy: 0.7718
Epoch 99/100
77/77 [=====] - 1s 17ms/step - loss: 0.2232 - accuracy: 0.9162 - val_loss: 4.9486 - val_accuracy: 0.7882
Epoch 100/100
77/77 [=====] - 1s 17ms/step - loss: 0.1731 - accuracy: 0.9281 - val_loss: 5.0320 - val_accuracy: 0.7882
20/20 [=====] - 0s 3ms/step - loss: 5.0320 - accuracy: 0.7882
Perte de test : 5.0320, Précision de test : 0.7882

```

FIGURE 13 – Résultat de l'entraînement

Après 100 époques d'entraînement, le modèle affiche une perte d'entraînement de 0.1731 avec une précision de 92.81%. Cependant, sur l'ensemble de validation, la perte est de 5.0320 avec une précision de 78.82%. Ces résultats indiquent que le modèle pourrait être confronté à un surajustement, car il semble bien performer sur les données d'entraînement mais moins bien sur de nouvelles données.

## 7 APPLICATION : DÉTECTION DE PLEURS

### 7.1 Détection du Son en Temps Réel

La détection du son en temps réel est réalisée à l'aide de la bibliothèque Sounddevice, qui permet la capture continue du flux audio du microphone. Un seuil de détection est préalablement défini afin d'initier le processus d'enregistrement lorsqu'une intensité sonore supérieure à ce seuil est détectée.

```

1 #Fonction de detection du son
2 def detection(indata, frames, time, status):
3     volume_norm = abs(indata).max()
4     global enregistrement_en_cours
5
6     if volume_norm > seuil and not enregistrement_en_cours:
7         enregistrement_en_cours = True
8     elif volume_norm <= seuil and enregistrement_en_cours:
9         enregistrement_en_cours = False

```

Code 2 – Détection

### 7.2 Caractéristiques Audio et Prétraitement

Les caractéristiques audio sont extraites des enregistrements à l'aide de la bibliothèque Librosa, générant des segments spectrogrammes. Ces segments sont ensuite prétraités pour être conformes aux exigences d'entrée du modèle de réseau de neurones.

```

1 #Fonction qui donne les caractéristique d'un fichier audio
2 def caracteristiques_audio(fichier_audio, taille_segment):
3     donnees_audio, _ = librosa.load(fichier_audio, sr=sr)
4     spectrogramme = librosa.feature.melspectrogram(y=donnees_audio, sr=sr)
5     segments = []
6     debut = 0
7     while debut + taille_segment <= spectrogramme.shape[1]:
8         segment = spectrogramme[:, debut:debut+taille_segment]
9         segments.append(segment)
10        debut += taille_segment
11    return segments

```

Code 3 – Caractéristiques audio

### 7.3 Détection de Pleurs de Bébé en Temps Réel

Le processus démarre avec la mise en place d'un flux audio en utilisant la fonction `InputStream` de la bibliothèque `SoundDevice`. Ce flux est ensuite constamment surveillé à l'aide d'une boucle infinie. Lorsqu'une condition d'enregistrement est activée, le script capture le son pendant une période spécifiée et l'enregistre dans un fichier au format MP3.

Après l'enregistrement, le script extrait des segments audio de taille fixe et les utilise comme entrée pour un modèle de prédiction préalablement entraîné. Les prédictions obtenues sont ensuite analysées pour déterminer le pourcentage de prédictions positives par rapport au nombre total de prédictions.

```
1 #Test
2 with sd.InputStream(callback=detection):
3     try:
4         while True:
5             if enregistrement_en_cours:
6                 print("Son detecte. Enregistrement en cours...")
7                 record = sd.rec(int(sec * sr), samplerate=sr, channels=2)
8                 sd.wait()
9                 while os.path.exists(enregistrement):
10                     i += 1
11                     nom_enregistrement = 'son' + str(i) + '.mp3'
12                     enregistrement = os.path.join(dossier_enregistrement,
13 nom_enregistrement)
14                     write(enregistrement, sr, record)
15                     print("Enregistrement termine.")
16                     segments_test = caracteristiques_audio(enregistrement,
17 taille_segment)
18                     predictions = modele.predict(np.array(segments_test))
19
20                     nombre_predictions_positives = np.sum(predictions)
21                     nombre_total_predictions = len(predictions)
22
23                     pourcentage_predictions_positives = (
24 nombre_predictions_positives / nombre_total_predictions) * 100
25                     print(f"Pourcentage de predictions positives : {
26 pourcentage_predictions_positives:.2f}%")
27 except KeyboardInterrupt:
28     print("Arret manuel de l'enregistrement.")
```

Code 4 – Test

## 8 Amélioration de l'apprentissage automatique

Les MFCC sont des représentations couramment utilisées pour la caractérisation des signaux audio dans le domaine de la reconnaissance vocale et de la musique. Ils capturent les caractéristiques fréquentielles du signal audio en utilisant une échelle logarithmique pour représenter la fréquence. Les MFCC sont calculés en appliquant une transformation de cosinus discrète (DCT) aux coefficients de la transformée de Fourier à court terme (STFT) des données audio. [1]

```

1 # Fonction pour obtenir les caracteristiques audio d'un fichier
2 def caracteristiques_audio(fichier_audio, taille_segment):
3     donnees_audio, _ = librosa.load(fichier_audio, sr=SR)
4     spectrogramme = librosa.feature.melspectrogram(y=donnees_audio, sr=SR)
5     mfccs = librosa.feature.mfcc(y=donnees_audio, sr=SR, n_mfcc=13)
6
7     segments_spectrogramme = []
8     debut = 0
9     while debut + taille_segment <= spectrogramme.shape[1]:
10         segment = spectrogramme[:, debut:debut+taille_segment]
11         segments_spectrogramme.append(segment.flatten())
12         debut += taille_segment
13
14     segments_mfccs = []
15     debut = 0
16     while debut + taille_segment <= mfccs.shape[1]:
17         segment = mfccs[:, debut:debut+taille_segment]
18         segments_mfccs.append(segment.flatten())
19         debut += taille_segment
20
21     return segments_spectrogramme, segments_mfccs

```

Code 5 – Nouveau caracteristiques audio

La fonction `caracteristiques_audio` est désormais capable de générer deux ensembles de caractéristiques distincts : les segments de spectrogramme et les segments de MFCC. Chaque segment est aplati en un vecteur pour être utilisé comme entrée dans notre modèle de réseau de neurones.

```

Epoch 200/200
77/77 [=====] - 2s 20ms/step - loss: 0.1366 - accuracy: 0.9290 - val_loss: 1.4648 - val_accuracy: 0.8785
20/20 [=====] - 0s 4ms/step - loss: 1.4648 - accuracy: 0.8785
Perte de test : 1.4648, Précision de test : 0.8785

```

FIGURE 14 – Nouveau résultat de l'entraînement

Suite à cette modification, nous avons entraîné à nouveau notre modèle en utilisant les nouvelles caractéristiques basées sur les MFCC. Les résultats ont montré une amélioration significative de la précision du modèle. La perte de test a diminué à 1.4648 et la précision de test a augmenté à 87.85%. Comparé aux résultats précédents, cette amélioration démontre l'efficacité des MFCC comme caractéristiques audio pour la détection des pleurs de bébé.

## 9 Mise en œuvre sur Raspberry Pi 3

Pour la mise en œuvre sur la Raspberry Pi 3, nous avons choisi d'utiliser Raspberry Pi OS (anciennement Raspbian), un système d'exploitation GNU/Linux standard conçu et optimisé pour les Raspberry Pi. Sa structure, basée sur la dernière version de la distribution Debian, en fait une option idéale pour notre projet, offrant une compatibilité étendue avec un large éventail de logiciels et de bibliothèques disponibles dans l'écosystème Debian.



FIGURE 15 – Fenêtre d'installation Raspberry Pi

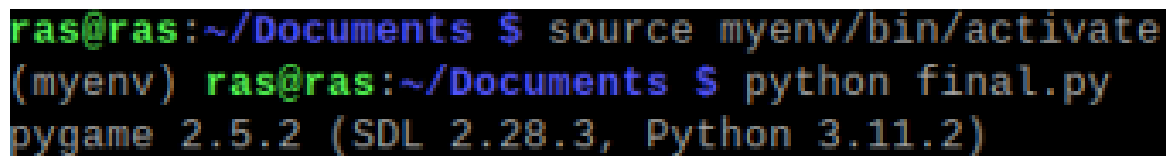
Cependant, lors de la configuration sur la Raspberry Pi 3, nous avons rencontré quelques obstacles, notamment des problèmes d'installation des bibliothèques requises. Ces ajustements sont nécessaires pour garantir le bon fonctionnement du projet sur cette plateforme spécifique.

## 9.1 Configuration d'un environnement virtuel

Étant donné que l'installation des bibliothèques nécessaires n'a pas fonctionné de manière conventionnelle, nous avons opté pour la création d'un environnement virtuel Python à l'aide de `virtualenv`. Cela nous a permis de travailler dans un environnement isolé et de gérer les dépendances spécifiques au projet sans interférer avec d'autres installations système.

```
1 sudo apt-get install python3-venv
2 python3 -m venv myenv
3 source myenv/bin/activate
```

Code 6 – Commande d'installation et d'initialisation de l'environnement virtuel



```
ras@ras:~/Documents $ source myenv/bin/activate
(myenv) ras@ras:~/Documents $ python final.py
pygame 2.5.2 (SDL 2.28.3, Python 3.11.2)
```

FIGURE 16 – Activation de l'environnement virtuel

## 9.2 Activation des GPIO pour le haut-parleur

Sur la Raspberry Pi, les GPIO (General Purpose Input/Output) doivent être activés pour permettre le contrôle du module du haut-parleur. Cela se fait généralement via la configuration du système. [3]

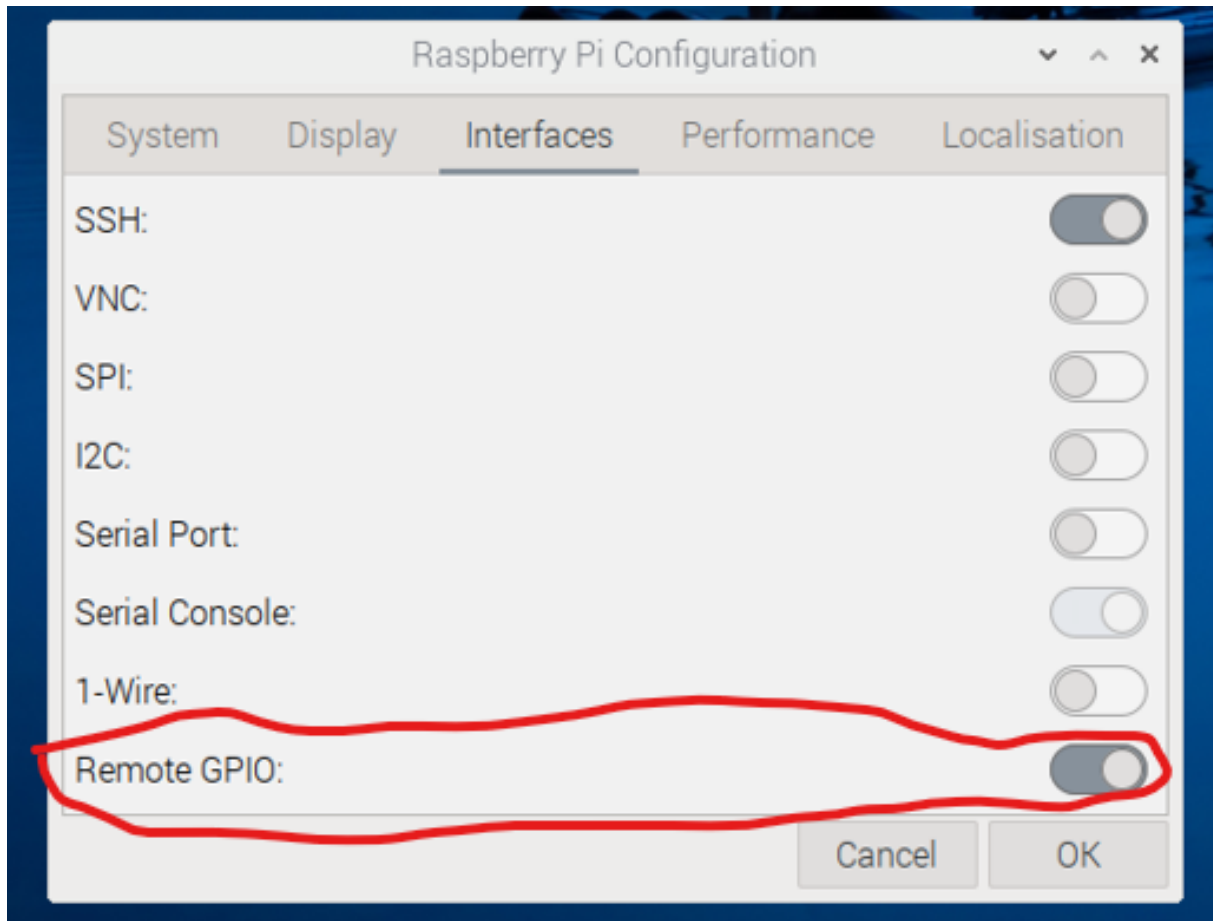


FIGURE 17 – Activation des GPIO

```
1 SPEAKER_PIN = 3
2
3 # Configuration des broches GPIO
4 GPIO.setmode(GPIO.BOARD)
5 GPIO.setup(SPEAKER_PIN, GPIO.OUT)
```

Code 7 – Pin utilisé



## 10 BILAN DU PROJET

Le projet "Outil de détection des pleurs d'un bébé" est désormais finalisé, marquant une étape importante dans son développement. La fusion réussie des composants matériels et logiciels a abouti à la création d'un système opérationnel capable de détecter en temps réel les pleurs d'un bébé, offrant ainsi une solution pratique pour les parents.

Initialement, le modèle de réseau de neurones, utilisant uniquement le spectrogramme du son, affichait une précision de 92.81% sur l'ensemble d'entraînement. Cependant, la baisse à 78.82% sur l'ensemble de validation suggérait la possibilité d'un surajustement. Afin de renforcer l'efficacité du modèle, l'ajout des caractéristiques MFCCs a entraîné une évolution significative, portant la précision à 87.85% sur l'ensemble de validation, renforçant ainsi l'efficacité du modèle.

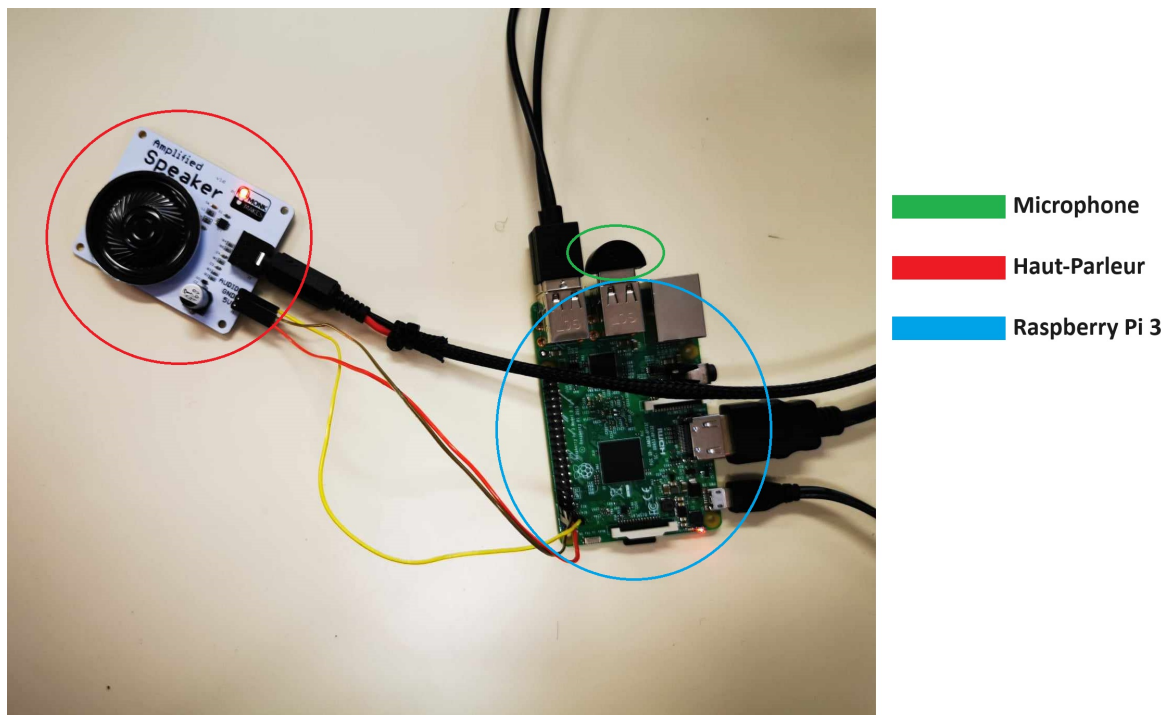


FIGURE 18 – Activation des GPIO

L'intégration réussie du haut-parleur dans l'environnement Raspberry Pi constitue une avancée majeure. Cette fonctionnalité permet une réponse proactive aux pleurs détectés, offrant une solution complète de détection et de réconfort pour le bien-être du bébé et la tranquillité d'esprit des parents.

En somme, le projet a pleinement atteint ses objectifs en proposant un outil fonctionnel et efficace pour la détection des pleurs de bébé, couplé à une réponse apaisante adaptée.

## 11 BIBLIOGRAPHIE

### Références

- [1] Uday KIRAN. *MFCC Technique for Speech Recognition*. 2023. URL : <https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/>.
- [2] Mauro Di PIETRO. *Deep Learning with Python : Neural Networks (complete tutorial)*. 2021. URL : <https://towardsdatascience.com/deep-learning-with-python-neural-networks-complete-tutorial-6b53c0b06af0>.
- [3] Tsiferana RABETOKOTANY. *Comment utiliser les port GPIO Raspberry Pi*. 2022. URL : <https://www.raspberrypi-france.fr/comment-utiliser-les-port-gpio-raspberry-pi/>.