

RAPPORT DE STAGE TECHNICIEN

Sujet : détection des maladies des plantes

Élaboré par :

Nom et prénom : Khandoudi Amina

Filière : Cycle de formation ingénieur en génie
informatique

Parcours : Systèmes embarqués et IoT

Niveau : 2eme années

Encadrée par : Dr. Tarhouni Mounira

Période du stage : 01/08/2024-31/08/2024

Table Des Matières

1. Introduction	
1.1. Remerciements	4
1.2. Introduction	4
2. Présentation du Système	
2.1. Architecture Générale du Système.....	5
2.2. Composants Utilisés.....	5
2.2.1 Environnement hardware	6
2.2.2 Environnement software	7
2.3. Fonctionnalités Principales	10
2.3.1 Surveillance des Conditions Climatiques	10
2.3.2 Détection des Maladies des Feuilles	10
2.3.3 Déploiement sur une Dashboard.....	10
3. Développement du Système	
3.1. Module de Surveillance des Conditions Environnementales	11
3.1.1. Fonctionnement	11
3.2.1. Schéma globale de l'application	11
3.2. Module de Détection des Maladies.....	12
3.2.1. Architecture globale de système	12
3.2.2. Déploiement de modèle CNN sur le Raspberry Pi5.	16
3.3. Développement de l'interface utilisateur	17
3.3.1. Configuration de Firebase	17
3.3.2. Configuration de Node-RED.....	17
4. Résultats et Discussion	
4.1. Résultat de Modèle CNN	23
4.1.1. Rapport de classification.....	23
4.1.2. Matrice de confusion	24
4.2. Résultat final de déploiement de projet.....	25
4.2.1. Déploiement de modèle CNN sur le Raspberry Pi5.	25
4.2.2. Résultat de mesure temps réel d'humidité et température	25
4.2.3. Résultat d'exécution de script final de tous le projet.....	26
5. Conclusion	27

Table Des Figures

Figure 1 : Raspberry Pi 5	6
Figure 1 : Capteur Température Humidité AHT10	6
Figure 3 : Carte Micro SD	7
Figure 4 : Logo Python	7
Figure 5: Logo VScode	7
Figure 6 : Logo Firebase	8
Figure 7: Logo Real VNC	8
Figure 8 : Logo Rasbrien OS	8
Figure 9: Logo Node-RED	9
Figure 10 : Logo Google Collab	9
Figure 11: Logo TensorFlow	9
Figure 12: Branchement de raspberry pi 5 avec le capteur AHT10	11
Figure 13: Branchements du capteur AHT10 avec le Raspberry pi 5	12
Figure 14 : Architecture globale de système de prédiction d'état de plante	12
Figure 15: Division de dataset	13
Figure 16 : Architecture d'un CNN	14
Figure 17: Matrice de confusion	15
Figure 18 : Conversion du modelé en format TFLite	16
Figure 19: Écriture du modèle TFLite dans un fichier	16
Figure 20 : Code de configuration de Firebase	17
Figure 21 : Code de configuration du capteur AHT10	17
Figure 22 : Installation de Node-RED sur le Raspberry pi 5	18
Figure 23 : Démarrage de Node-RED	18
Figure 24: Interface de Login de Node-RED	19
Figure 25 : conception graphique d'interface utilisateur	19
Figure 26: Configuration d'une jauge pour l'humidité	20
Figure 27 : Configuration d'une jauge pour l'humidité	20
Figure 28: Configuration d'un graphe pour la température	21
Figure 29 : Configuration d'un graphe pour la température	21
Figure 30: Configuration d'un affichage textuel pour les classes des plantes	22
Figure 31: Configuration d'un affichage textuel pour les classes des plantes	22
Figure 32 : Rapport de classification	23
Figure 33: Matrice de confusion	24
Figure 34 : Déploiement de modèle TFLite sur le Raspberry Pi 5	25
Figure 35: Mesure temps réel d'humidité et température	25
Figure 36 : Résultat d'exécution de script final du projet	26
Figure 37 : Interface Utilisateur du projet	26

1. Introduction

1.1 Remerciements :

Je tiens à exprimer ma profonde gratitude au **laboratoire Hatem Bettahar IResCoMath Lab**, où j'ai eu l'opportunité d'effectuer mon stage technicien. Je remercie également **Madame Mounira Tarhouni**, mon encadrante, pour son accompagnement précieux, ses conseils avisés et son soutien tout au long de ce projet. Leur expertise et leur engagement ont grandement contribué à l'avancement et à la réussite de ce travail.

1.2 Introduction:

L'agriculture est un secteur vital pour l'économie mondiale, mais elle doit faire face à des défis croissants liés aux changements climatiques, à l'augmentation de la population et à la gestion durable des ressources naturelles. Pour répondre à ces enjeux, l'**agriculture intelligente (Smart Agriculture)** s'impose comme une solution innovante. Grâce aux avancées technologiques telles que l'Internet des Objets (IoT), l'intelligence artificielle et les systèmes de surveillance automatisés, il est désormais possible d'optimiser la gestion des cultures et d'améliorer les rendements tout en minimisant l'impact environnemental.

Cependant, un des problèmes majeurs rencontrés par les agriculteurs est la **surveillance des conditions environnementales et la détection précoce des maladies des plantes**. Les fluctuations de température et d'humidité influencent directement la croissance des cultures, tandis que la propagation rapide des maladies peut causer des pertes importantes. Les méthodes traditionnelles de surveillance et de diagnostic sont souvent coûteuses, peu précises et nécessitent une intervention humaine constante, ce qui limite leur efficacité.

Pour répondre à ces défis, ce projet propose le développement d'un **système IoT intelligent** permettant une **surveillance en temps réel des conditions climatiques** grâce à des capteurs mesurant la température et l'humidité. Il intègre également un **module de vision par ordinateur** utilisant l'apprentissage automatique pour détecter les maladies des feuilles, ainsi qu'une **interface utilisateur intuitive** permettant aux agriculteurs de visualiser les données et de prendre des décisions éclairées. Cette solution vise ainsi à améliorer la gestion des cultures, à réduire les pertes et à favoriser une agriculture plus efficace et durable.

2. Présentation du Système

2.1 Architecture Générale du Système:

L'architecture générale du système repose sur l'intégration cohérente de plusieurs modules matériels et logiciels interconnectés. Ces modules collaborent pour collecter, analyser et afficher les données pertinentes sur l'environnement agricole et l'état des cultures.

2.1.1 Schéma Global:

Le système est structuré autour de trois composantes principales :

- Capteurs (AHT10): pour collecter les données de l'environnement externe.
- Unité de traitement (Raspberry Pi 5): Cet élément central collecte les données des capteurs, les traite, et exécute le modèle d'analyse des maladies des feuilles.
- Interface Utilisateur: Une interface utilisateur accessible pour le user qui affiche les données collectées et les résultats des analyses.

2.2. Composants Utilisés :

La conception du système repose sur une combinaison de composants matériels et logiciels qui assurent la collecte, le traitement, la transmission et l'affichage des données. Ces composants ont été choisis pour leur compatibilité, leur efficacité et leur pertinence par rapport aux besoins du projet.

2.2.1 Environnement hardware :

1/ Raspberry Pi 5 :

Plateforme centrale de traitement des données. Elle collecte les mesures des capteurs, traite les images des feuilles et assure la communication avec le cloud.

- Processeur Quad-core ARM Cortex-A72.
- Support de systèmes d'exploitation comme Raspberry Pi OS.
- Connectivité Wi-Fi et Bluetooth intégrée, facilitant la communication IoT.

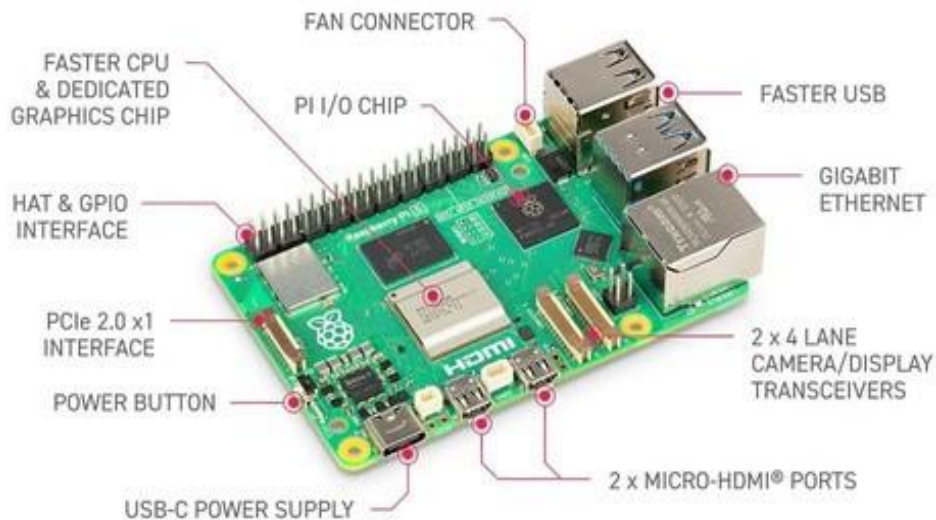


Figure 1 : Raspberry Pi 5

2/ Capteur température humidité AHT10 : Le AHT10 est un capteur numérique avancé conçu pour mesurer simultanément la température et l'humidité relative de l'air. Il intègre un capteur capacitif pour l'humidité et un capteur thermorésistif pour la température, offrant une précision élevée, une stabilité à long terme et une faible consommation d'énergie.

Caractéristiques principales :

1. Plage de mesure :

- Température : de -40°C à $+85^{\circ}\text{C}$.
- Humidité : de 0% à 100% RH (Humidité Relative).

2. Précision :

- Température : $\pm 0.3^{\circ}\text{C}$.
- Humidité : $\pm 2\%$ RH.



Figure 2 : Capteur Température Humidité AHT10

3/ Carte micro SD:

Par définition, une carte micro SD est une carte mémoire amovible utilisée pour stocker des données numériques. Elle diffère de la carte SD par sa petite taille, qui peut être utilisée dans presque tous les téléphones portables, téléphones intelligents et lecteurs de musique. La carte Micro SD est une technologie créée en 2005, sa taille est vraiment très petite (environ 11 mm x 15 mm x 1 mm), ce qui ne l'empêche pas d'avoir une très bonne qualité de stockage (16 Go).



Figure 3 : Carte micro SD

4/ Une alimentation :

Afin de fournir la puissance requise par le Raspberry Pi, il faut utiliser un cordon d'alimentation. Nous devons avoir une alimentation avec un courant de sortie maximum de 2,5 A et une tension de 5V

5/ Alimentation Électrique :

Une alimentation 5V/3A est utilisée pour assurer le fonctionnement continu de la Raspberry Pi et des capteurs.

2.2.2 Environnement software :

a. Environnement de Programmation:

- Python : Utilisé pour le traitement des données issues des capteurs et la mise en œuvre du modèle d'analyse des images.



Figure 4 : Logo Python

- VS code:

Visual Studio Code est un éditeur de code redéfini et optimisé pour la création et le débogage d'applications Web et cloud modernes.



Figure 5 : Logo VScode

b. Plateforme Firebase:

Firebase (base de données NoSQL JSON) est une base de données temps réel qui permet de stocker une liste d'objets sous forme d'arborescence.

Durant ce projet, on a utilisé le Firebase pour stocker et synchroniser les données entre la Raspberry Pi et l'application mobile.



Figure 6 : Logo Firebase

c. Real VNC :

Le logiciel VNC (Virtual Network Computing) permet de contrôler un ordinateur distant en affichant son écran et en interagissant via le clavier et la souris. Il utilise un protocole client-serveur pour transmettre les données en temps réel sur un réseau. C'est une solution pratique pour le support à distance, la gestion de serveurs et le travail à distance.



Figure 7 : Logo Real VNC

d. OS Raspbian :

Raspbian est un système d'exploitation gratuit basé sur Debian, optimisé pour le matériel Raspberry pi. Le système d'exploitation est un ensemble de programmes et d'utilitaires de base.



Figure 8 : Logo Raspbian OS

e. Node-Red:

Node-RED est un outil de programmation permettant de relier des périphériques matériels, des API et des services en ligne de manière nouvelle et intéressante.

Il fournit un éditeur basé sur un navigateur qui facilite la connexion des flux à l'aide du large éventail de nœuds de la palette qui peuvent être déployés dans son environnement d'exécution en un seul clic.



Figure 9: Logo Node-Red

e. Google Collab:

(Google Collaboratory) est un service gratuit de Google qui permet d'écrire et d'exécuter du code Python dans un environnement cloud. Il offre des ressources comme des GPU et TPU pour le calcul intensif, sans configuration nécessaire. Idéal pour l'apprentissage automatique, l'analyse de données et les projets collaboratifs.



Figure 10 : Logo Google Collab

e. Bibliothèques (Modules) utilisées :

- La bibliothèque Open CV (open source computer vision) : est une bibliothèque utilisée plusieurs langages en informatique par exemple le langage C et C++ et des autres langages comme linux etc. Cette bibliothèque est développée par langage python.
- La bibliothèque TensorFlow: Est une bibliothèque d'apprentissage automatique open source développée par Google. TensorFlow est utilisé pour créer et entraîner des modèles d'apprentissage en profondeur car il facilite la création de graphiques informatiques et une exécution efficace sur diverses plates-formes matérielles. L'article fournit un aperçu complet de Tensorflow.



Figure 11 : Logo TensorFlow

- TensorFlow Lite (TFLite): est une version allégée de TensorFlow, conçue pour exécuter des modèles de machine learning sur des appareils embarqués et mobiles. Il optimise les modèles pour être rapides, légers et adaptés aux environnements avec des ressources limitées, comme les smartphones, microcontrôleurs ou Raspberry Pi. TFLite permet d'intégrer l'intelligence artificielle directement dans des applications embarquées.

2.3 Fonctionnalités Principales :

Le système développé dans le cadre de ce mini-projet se distingue par ses fonctionnalités innovantes, conçues pour répondre aux besoins spécifiques de l'agriculture intelligente. Ces fonctionnalités permettent de surveiller les conditions environnementales, de détecter les maladies des plantes, et de faciliter la prise de décision grâce à une application mobile intuitive.

2.3.1 Surveillance des Conditions Climatiques :

Cette fonctionnalité permet de mesurer en temps réel la température et l'humidité de l'environnement agricole grâce à des capteurs. Ces données, collectées et traitées par la Raspberry Pi, sont envoyées à l'application mobile pour une surveillance en temps réel. Cela aide les utilisateurs à détecter rapidement les variations climatiques et à ajuster les systèmes d'irrigation ou de ventilation en conséquence.

2.3.2 Détection des Maladies des Feuilles :

Le système analyse des images de feuilles envoyées par les utilisateurs pour détecter la présence de maladies. Les photos prises avec un smartphone sont transférées au modèle de machine learning via l'application mobile, où elles sont classées comme saines ou malades. Cette fonctionnalité offre une détection rapide et précise, permettant une intervention efficace pour limiter les pertes de cultures.

2.3.3 Déploiement sur une Dashboard:

La troisième fonctionnalité vise à simplifier l'interaction avec le système grâce à une Dashboard développée avec l'outil Node-Red, offrant une interface utilisateur conviviale et intuitive.

Description: L'interface utilisateur affiche en temps réel les données collectées par le système, notamment :

- Les mesures de température et d'humidité.
- L'état des feuilles des plantes (classe de feuille).

3. Développement du Système

Cette partie détaille le développement du système en quatre modules principaux : la surveillance des conditions environnementales, la détection des maladies des feuilles, le développement de l'application mobile, et l'intégration des différents composants pour assurer une communication fluide.

3.1. Module de Surveillance des Conditions Environnementales

Le module de surveillance des conditions environnementales est responsable de la collecte et de l'analyse des données climatiques essentielles telles que la température et l'humidité.

3.1.1. Fonctionnement

Le fonctionnement repose sur trois étapes principales. Tout d'abord, un script Python exécuté sur le Raspberry Pi pour lire les mesures des capteurs. Ensuite, un traitement des données est effectué afin de filtrer les valeurs extrêmes ou incohérentes, assurant ainsi une précision optimale. Enfin, les données collectées sont transmises à Firebase Realtime Database, permettant leur consultation en temps réel via une interface utilisateur.

3.1.2. Schéma globale de l'application :

- Le schéma suivant montre la liaison physique de notre application qui relie le Raspberry pi 5 avec le capteur AHT10:

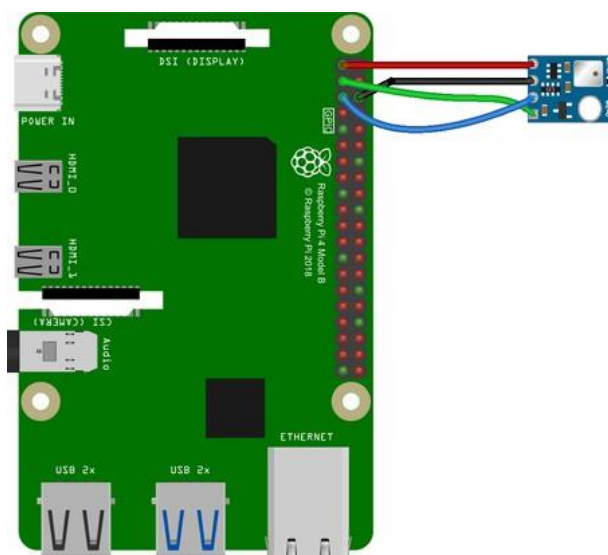


Figure 12 : Branchement de Raspberry pi 5 avec le capteur AHT10

- **Branchements du capteur AHT10 avec le Raspberry pi 5:**

AHT10	Raspberry Pi 5
VCC+	5V
GND	GND
SCL	SCL (GPIO 3)
SDA	SDA (GPIO 2)

Figure 13 : Branchements du capteur AHT10 avec le Raspberry pi 5

3.2. Module de Détection des Maladies

Ce module analyse des images de feuilles pour déterminer leur état de santé (saines ou malades) en utilisant un modèle d'intelligence artificielle.

3.2.1. Architecture globale de système:

L'architecture globale de notre système de prédiction d'état de santé des feuilles contient plusieurs étapes qui sont présentée dans la figure suivante:

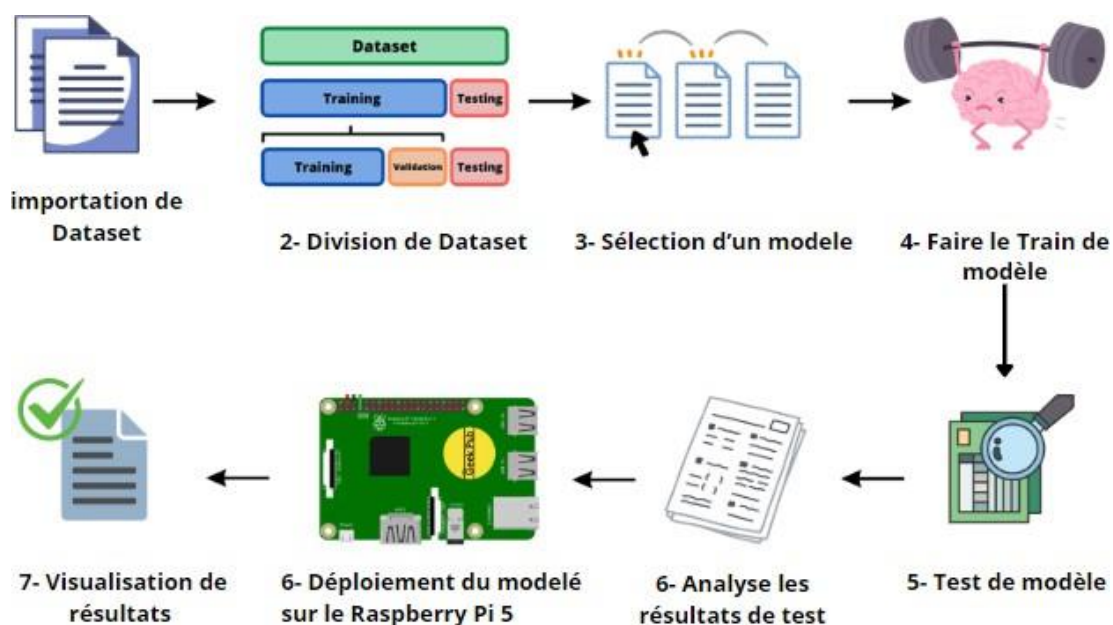


Figure 14 : Architecture globale de système de prédiction d'état de plante

Premièrement, on a fait l'importation de dataset a partie d'un lien suivant:

<https://www.kaggle.com/datasets/vipooooool/new-plant-diseases-dataset>

Deuxièmement, on a divisée notre dataset en 3 partie : Train , Test et validation

Troisièmement, Faire la sélection d'un modèle puis le test de modèle choisi.

Finalement, l'analyse des résultats de test et le déploiement de modèle sur le Raspberry Pi.

A/ Dataset utilisée:

Cet ensemble de données comprend environ 87 000 images RVB de feuilles de cultures saines et malades, classées en 38 classes différentes.

B/ Division de Dataset:

L'ensemble de données total est divisé en un rapport 80/20 d'ensemble de formation et de validation préservant la structure du répertoire. Un nouveau répertoire contenant 33 images de test est créé ultérieurement à des fins de prédiction.

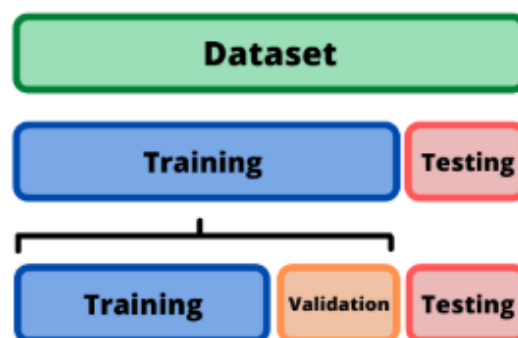


Figure 15 : Division de dataset

C/ Sélection de modèle:

On a choisi le modèle de réseau neuronal convolutif (CNN) qui a été conçu pour extraire les caractéristiques visuelles importantes et classifier les images.

- Architecture d'un CNN:

Les CNN désignent une sous-catégorie de réseaux de neurones et sont à ce jour un des modèles de classification d'images réputés être les plus performant.

Leur mode de fonctionnement est à première vue simple : l'utilisateur fournit en entrée une image sous la forme d'une matrice de pixels.

Celle-ci dispose de 3 dimensions :

- Deux dimensions pour une image en niveaux de gris.
- Une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales (Rouge, Vert, Bleu).

Contrairement à un modèle MLP (Multi Layers Perceptron) classique qui ne contient qu'une partie classification, l'architecture du Convolutional Neural Network dispose en amont d'une partie convolutive et comporte par conséquent deux parties bien distinctes :

Une partie convolutive : Son objectif final est d'extraire des caractéristiques propres à chaque image en les compressant de façon à réduire leur taille initiale. En résumé, l'image fournie en entrée passe à travers une succession de filtres, créant par la même occasion de nouvelles images appelées cartes de convolutions. Enfin, les cartes de convolutions obtenues sont concaténées dans un vecteur de caractéristiques appelé code CNN.

Une partie classification : Le code CNN obtenu en sortie de la partie convolutive est fourni en entrée dans une deuxième partie, constituée de couches entièrement connectées appelées perceptron multicouche (MLP pour Multi Layers Perceptron). Le rôle de cette partie est de combiner les caractéristiques du code CNN afin de classer l'image.

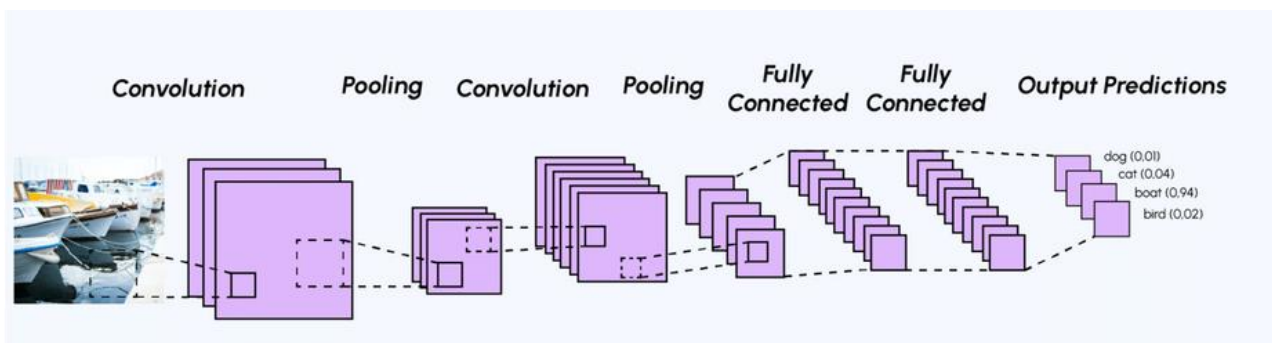


Figure 16 : Architecture d'un CNN

D/ Mesures et méthodes d'évaluation de la classification:

On a utilisé comme des métrique d'évaluation de notre modèle CNN sont comme suivant: **Un rapport de classification:** mesure l'exactitude des prédictions d'un algorithme de classification. Combien les prédictions sont vraies ou fausses. Le rapport affiche la classification principale précision des métriques, rappel et score f1 par classe. Les métriques sont calculées à l'aide de quatre méthodes pour déterminer si les prédictions sont correctes ou incorrectes :

TP(True Positive): quand le cas était positive et prédit positive

TN(True Negative): quand le cas était positive et prédit négatif

FP(False Positive): quand le cas était négatif et prédit positive

FN(False Negative): quand le cas était négatif et prédit négatif

Accuracy : Permet de mesurer l'efficacité d'un modèle à prédire correctement à la fois les individus positifs et négatifs.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Recall: est la capacité d'un classificateur à trouver toutes les instances positives. C'est la fraction de positifs qui ont été correctement identifiés. Pour chaque classe, il est défini comme le rapport des vrais positifs aux la somme des vrais positifs et des faux négatifs.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

F-score: est une moyenne harmonique pondérée de précision et de rappel telle que le meilleur score est de 1,0. et le pire est 0,0. Il est défini comme dans l'équation suivante :

$$\text{F-Score} = [2 * (\text{Recall} * \text{Precision})] / [\text{Recall} + \text{Precision}]$$

Précision: est la capacité d'un classificateur à ne pas qualifier de positive une instance qui est en réalité négative. La précision est l'exactitude des prédictions positives. Pour chaque classe, elle est définie comme le rapport des vrais positifs à la somme des vrais et des faux positifs.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Matrice de confusion:

Une matrice de confusion est une méthode de mesure des performances pour la classification du Machine Learning. Il nous aide à connaître les performances du modèle de classification sur un ensemble de données de test pour que les valeurs vraies et fausses sont connues. Il s'agit donc d'un outil très important pour évaluer la classification modèles. Comme le montre la figure suivante, il utilise les quatre méthodes d'évaluation des classificateurs nommant TP, TN, FP et FN.

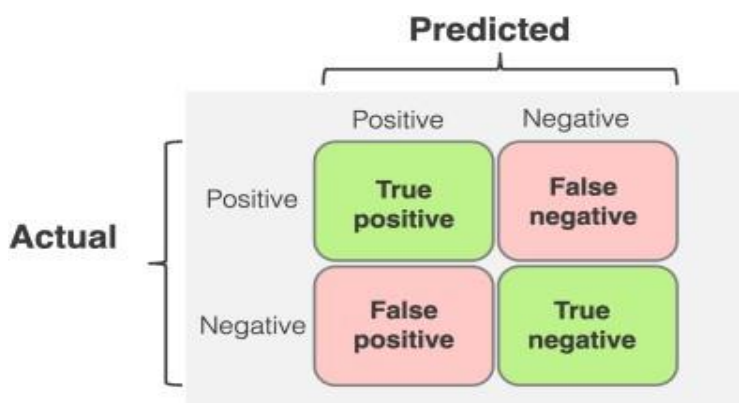


Figure 17 : Matrice de confusion

3.2.2. Déploiement de modèle CNN sur le Raspberry Pi5 :

Après les phases d'entraînement et de test de notre modèle, nous procédons au déploiement du modèle CNN sur le Raspberry Pi 5 en suivant les étapes suivantes :

A/ Préparation du modèle :

Dans cette partie, on a convertir le modèle CNN en un format optimisé pour le Raspberry Pi 5 en modèle TFLite a l'aide des lignes de code suivant:

```
# Convertir le modèle en format TFLite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

Figure 18 : Conversion du modelé en format TFLite

Ensuite, on a télécharger le modèle entraîné:

```
[ ] # Écrire le modèle TFLite dans un fichier
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)

from google.colab import files

files.download('model.tflite')
```

Figure 19 : Écriture du modèle TFLite dans un fichier

B/ Installation des dépendances :

Dans cette partie, on a Installer les bibliothèques nécessaires comme TensorFlow Lite, NumPy, et OpenCV. a l'aide des lignes de code suivant:

“sudo apt python3 install tflite-runtime numpy opencv-python”

C/ Transfert du modèle :

Ensuite, on a copier le modèle converti (.tflite) sur le Raspberry Pi via une clé USB.

D/ Développement du script de prédiction :

L'étape suivante est d'écrire un script Python pour charger le modèle, traiter les images et effectuer des prédictions.

D/ Exécution du modèle :

Finalement, on a lancer le script Python pour effectuer des prédictions sur les images capturées.

3.3. Module de Détection des Maladies

Le déploiement des données sur une Dashboard est une étape clé pour permettre une visualisation en temps réel des mesures collectées par le capteur AHT10. Pour ce faire, nous avons intégré Firebase à une interface Node-RED, un outil puissant pour la création de flux d'automatisation et de visualisation de données. Cette section détaille également la configuration de Firebase, essentielle pour l'intégration et la communication des données.

3.3.1. Configuration de Firebase:

Firebase a été configuré pour servir de pont entre le Raspberry Pi et la Dashboard Node-RED. Voici les étapes suivies :

A. Création d'un Projet Firebase: Nous avons créé un nouveau projet sur la console Firebase. Une base de données Realtime Database a été activée pour stocker les données envoyées par le Raspberry Pi. La base a été configurée avec des règles de lecture et d'écriture adaptées pour permettre un accès sécurisé.

B/Ajout des Identifiants Firebase dans le Code Python : Un fichier de configuration contenant les informations nécessaires (API Key, URL de la base de données, identifiant de projet) a été généré dans Firebase et intégré dans le script Python du Raspberry Pi.

```
# Firebase configuration
FIREBASE_CREDENTIALS_PATH = "/home/raspberry/Desktop/plantdis/new/dataset-agriculture-firebase-adminsdk-9v1wm-8s0b0449aa.json"
FIREBASE_DATABASE_URL = "https://dataset-agriculture-default-rtdb.firebaseio.com/"
```

Figure 20 : Code de configuration de Firebase

C. Envoi des Données depuis le Raspberry Pi: Les données mesurées par le capteur AHT10 (température et humidité) sont envoyées vers Firebase à l'aide du script Python suivant :

```
def update_data_in_firebase(humidity, temperature, plant_class):
    """Update data in Firebase Realtime Database."""
    ref = db.reference("/sensor_data/latest")
    data = {
        "temperature": temperature,
        "humidity": humidity,
        "plant_class": plant_class,
        "timestamp": int(time.time())
    }
    ref.set(data)
```

Figure 21 : Code de configuration du capteur AHT10

3.3.2. Configuration de Node-RED:

Dans cette partie, on a suivi les étapes suivantes pour créer une interface utilisateur avec l'outil Node-RED

A. Installation de Node-RED sur Raspberry Pi:

Pour installer le Node-RED sur le Raspberry Pi en tape la commande suivant sur le terminal de Raspberry Pi:

```
“bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)”
```

Node-RED est installé par défaut sur le système d'exploitation Raspberry Pi (32 bits). Cependant, il est recommandé d'exécuter la commande précédente pour installer les packages requis et les mettre à jour vers la version la plus récente.

Node-RED n'est pas installé par défaut sur Raspberry Pi OS (64 bits).

L'installation de Node-RED prendra quelques minutes. Au final, nous avons recevoir un message suivant dans la fenêtre du Terminal :

```

pi@raspberrypi:~$
Stop Node-RED
Remove old version of Node-RED
Remove old version of Node.js
Install Node.js 14 LTS          v14.19.3    Npm 6.14.17
Clean npm cache
Install Node-RED core           2.2.2
Move global nodes to local
Npm rebuild existing nodes
Install extra Pi nodes
Add shortcut commands
Update systemd script

Any errors will be logged to /var/log/nodered-install.log
All done.
You can now start Node-RED with the command node-red-start
or using the icon under Menu / Programming / Node-RED
Then point your browser to localhost:1880 or http://{your_pi_ip-address}:1880

Started : Mon 6 Jun 17:26:50 WEST 2022
Finished: Mon 6 Jun 17:32:27 WEST 2022

You may want to run node-red admin init
to configure your initial options and settings.

pi@raspberrypi:~$

```

Figure 22 : Installation de Node-RED sur le Raspberry pi 5

Après avoir installer et configurer le Node-RED, On a exécuter la commande suivante pour démarrer Node-RED :

“node-red-start”

```

pi@raspberrypi: Node-RED console
Settings file written to /home/pi/.node-red/settings.js
pi@raspberrypi:~$ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.1.106:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use node-red-log to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
6 Jun 17:41:23 - [info]
Welcome to Node-RED

6 Jun 17:41:23 - [info] Node-RED version: v2.2.2
6 Jun 17:41:23 - [info] Node.js version: v14.19.3
6 Jun 17:41:23 - [info] Linux 5.15.32-v8+ arm64 LE
6 Jun 17:41:23 - [info] Loading palette nodes
6 Jun 17:41:25 - [info] Settings file : /home/pi/.node-red/settings.js
6 Jun 17:41:25 - [info] Config store : 'default' [module=memory]
6 Jun 17:41:25 - [info] User directory : /home/pi/.node-red
6 Jun 17:41:25 - [warn] Projects disabled : editorTheme.projects.enabled=false
6 Jun 17:41:25 - [info] Flows file : /home/pi/.node-red/flows.json
6 Jun 17:41:25 - [info] Creating new flow file
6 Jun 17:41:25 - [info] Server now running at http://127.0.0.1:1880/
6 Jun 17:41:25 - [info] Starting flows
6 Jun 17:41:25 - [info] Started flows

```

Figure 23 : Démarrage de Node-RED

- **Démarrage automatique de Node-RED:**

Pour exécuter automatiquement Node-RED au démarrage du Pi, On a entré la commande suivante. Cela signifie que tant que notre Raspberry Pi est sous tension, Node-RED fonctionnera.

“sudo systemctl enable nodered.service”

Puis, on a redémarrer notre Raspberry Pi pour que le démarrage automatique prenne effet. Au prochain redémarrage du Raspberry Pi, Node-RED sera déjà en cours d'exécution.

“sudo reboot”

- **Access au Node-RED:**

Node-RED fonctionne sur le port 1880. Pour accéder à Node-RED, on ouvre un navigateur et on tape l'adresse IP du Raspberry Pi suivie de :1880. Comme suivant:

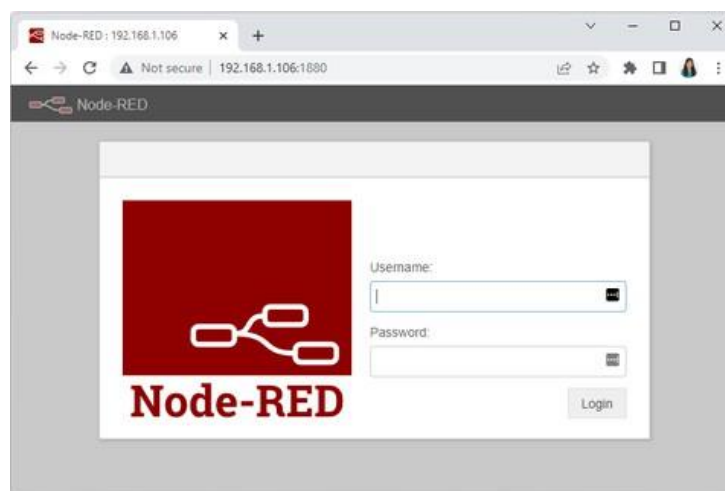


Figure 24 : Interface de Login de Node-RED

Après avoir accéder au Node-RED, on a développé notre propre interface utilisateur comme suivant :

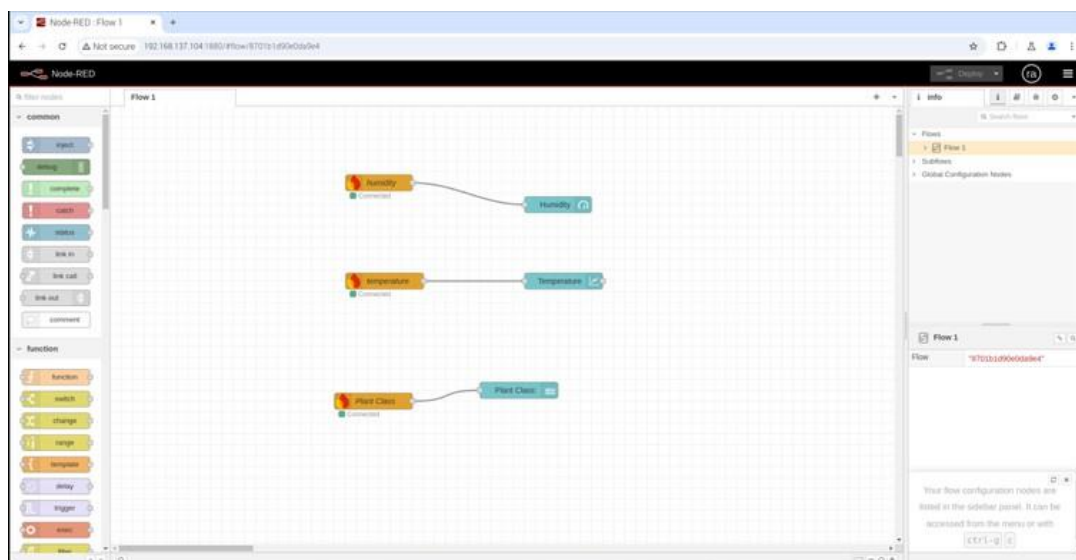


Figure 25 : conception graphique d'interface utilisateur

On a utilisé les composants nœuds Firebase (humidité, température et classe des plantes) connectés à des widgets Dashboard pour visualiser les données : jauge pour l'humidité, graphique pour la température et affichage textuel pour la classe des plantes.

- **Nœud Firebase Humidité:**

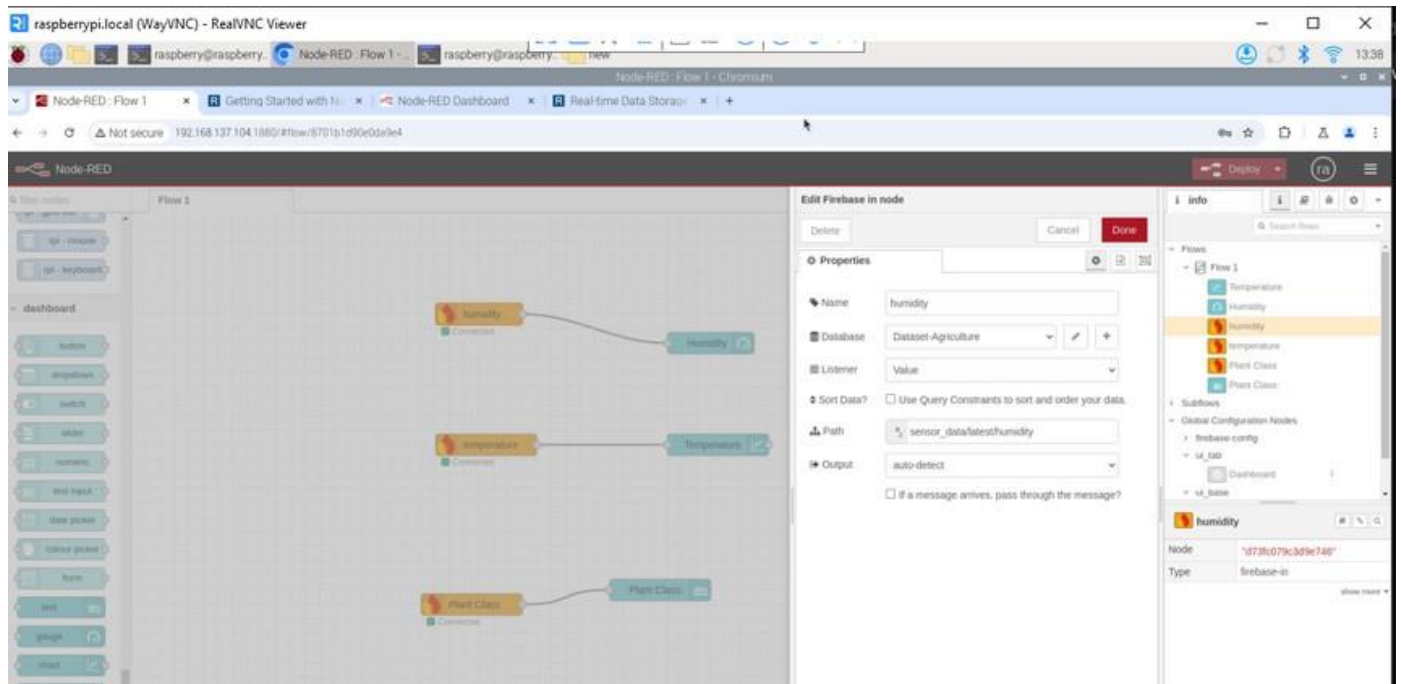


Figure 26 : Configuration d'une jauge pour l'humidité

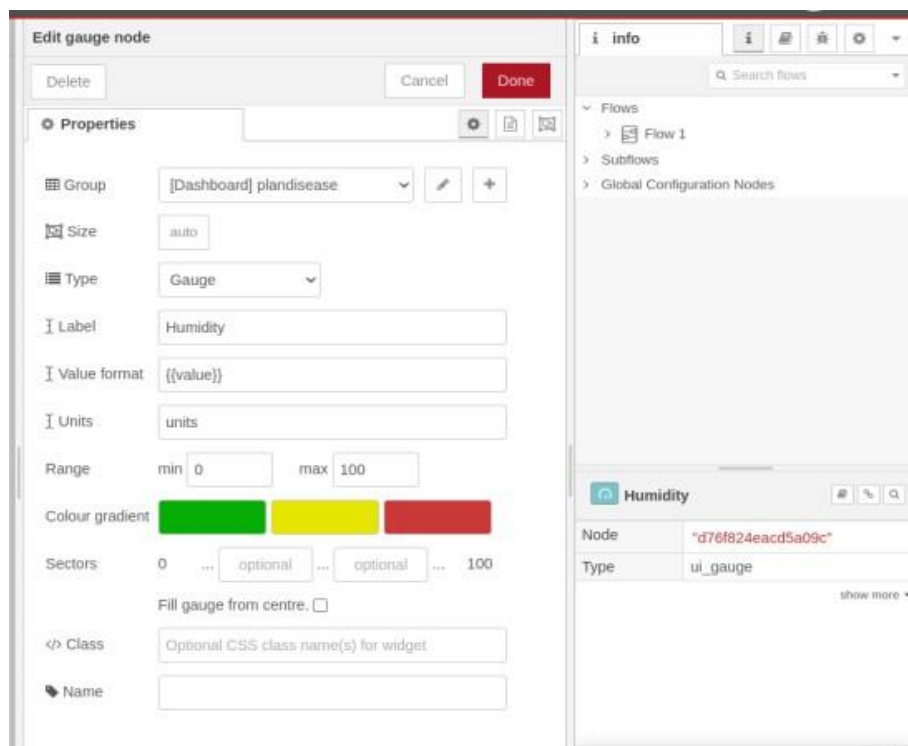


Figure 27 : Configuration d'une jauge pour l'humidité

• Nœud Firebase Temperature:

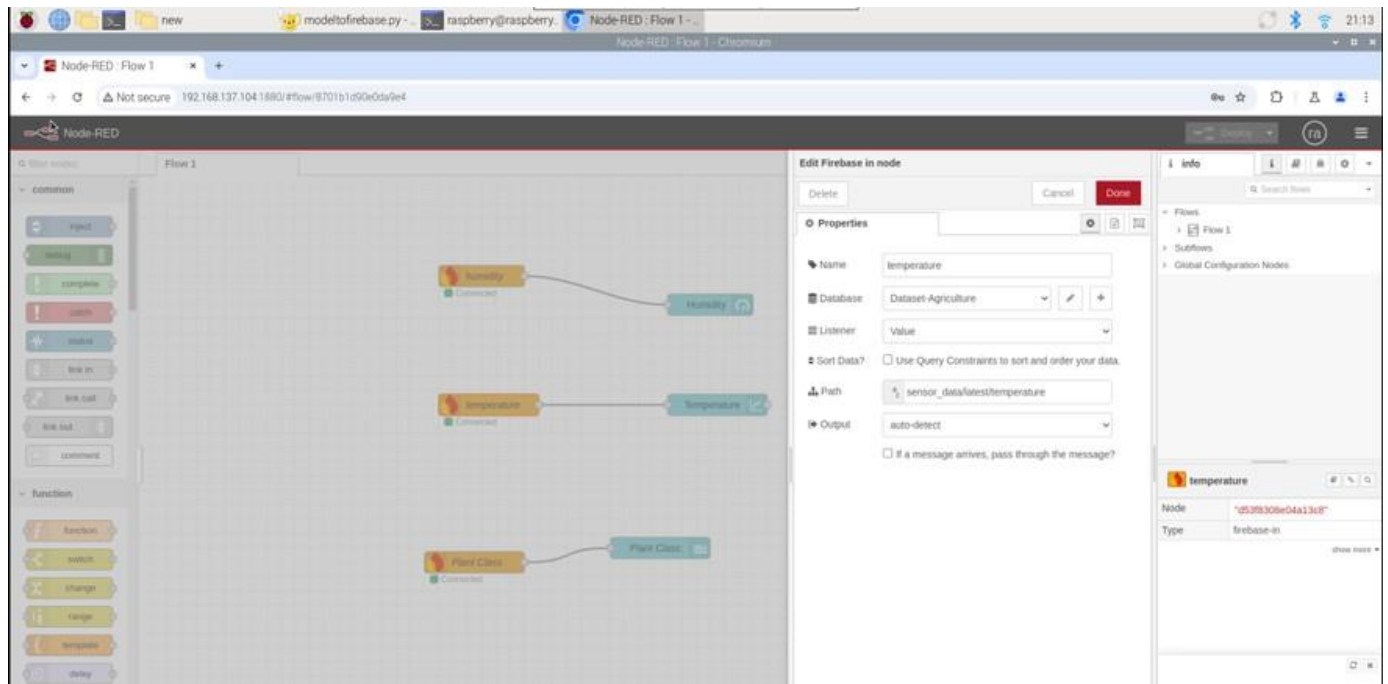


Figure 28 : Configuration d'un graphe pour la température

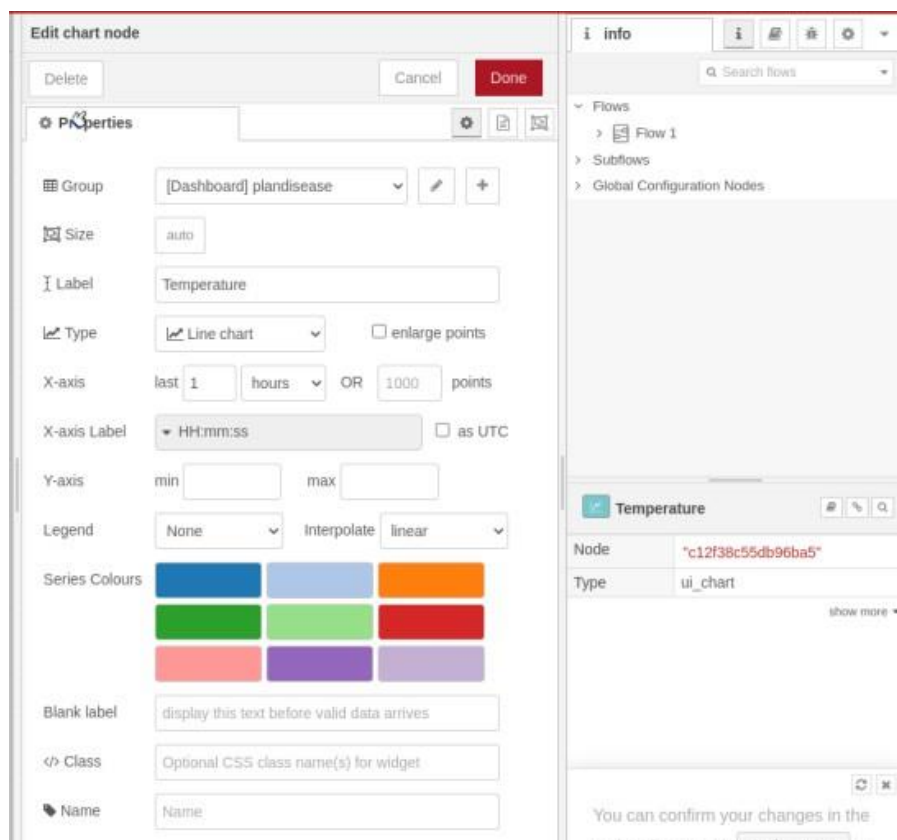


Figure 29 : Configuration d'un graphe pour la température

• **Nœud Firebase Plant Class:**

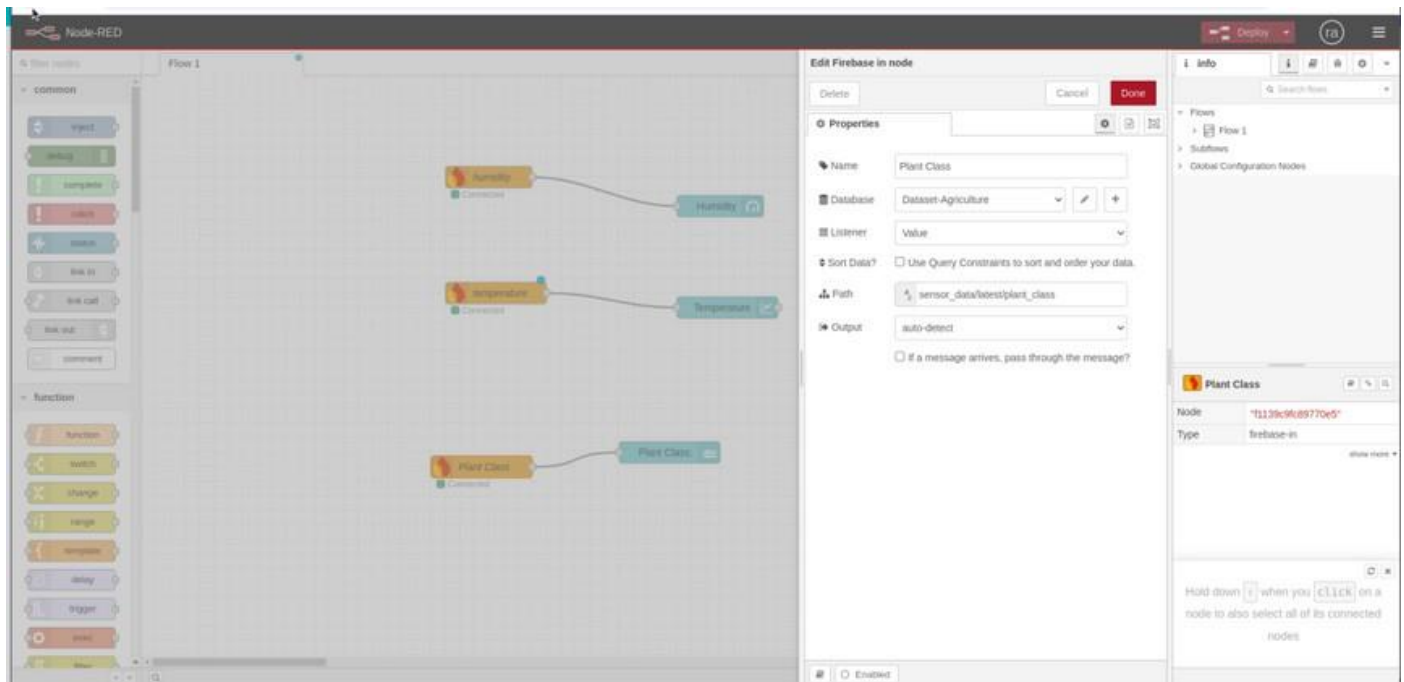


Figure 30 : Configuration d'un affichage textuel pour les classes des plantes

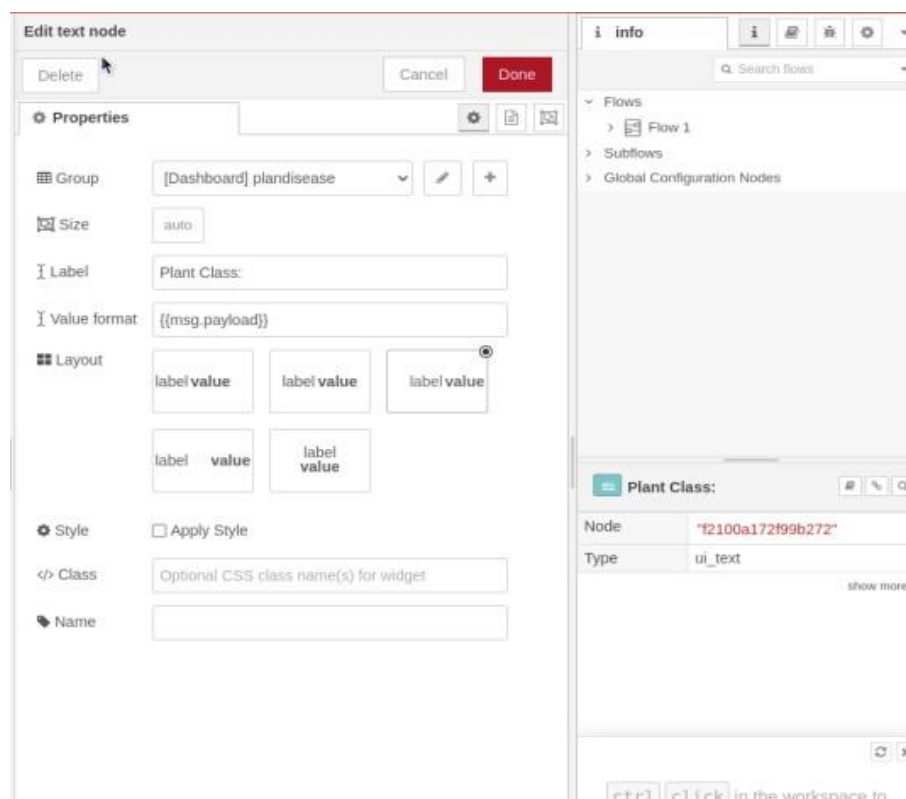


Figure 31 : Configuration d'un affichage textuel pour les classes des plantes

4. Résultat et discussion :

4.1. Résultat de Modèle CNN :

Notre modèle CNN a obtenu les valeurs suivantes comme résultats de train:

Training accuracy: 98%

test accuracy: 96%

Pour la phase de test, on a obtenu les résultats suivantes:

4.1.1. Rapport de classification:

	precision	recall	f1-score	support
Apple___Apple_scab	0.92	0.98	0.95	504
Apple___Black_rot	0.95	0.99	0.97	497
Apple___Cedar_apple_rust	0.99	0.93	0.96	440
Apple___healthy	0.94	0.98	0.96	502
Blueberry___healthy	0.97	0.99	0.98	454
Cherry_(including_sour)___Powdery_mildew	1.00	0.98	0.99	421
Cherry_(including_sour)___healthy	0.99	0.97	0.98	456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.94	0.91	0.93	410
Corn_(maize)___Common_rust	0.99	0.99	0.99	477
Corn_(maize)___Northern_Leaf_Blight	0.95	0.95	0.95	477
Corn_(maize)___healthy	1.00	0.99	0.99	465
Grape___Black_rot	0.97	0.97	0.97	472
Grape___Esca_(Black_Measles)	0.99	0.98	0.98	480
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.97	1.00	0.98	430
Grape___healthy	0.99	0.98	0.99	423
Orange___Haunglongbing_(Citrus_greening)	0.99	0.99	0.99	503
Peach___Bacterial_spot	0.95	0.96	0.96	459
Peach___healthy	0.94	1.00	0.97	432
Pepper,_bell___Bacterial_spot	0.96	0.96	0.96	478
Pepper,_bell___healthy	0.96	0.95	0.96	497
Potato___Early_blight	0.96	1.00	0.98	485
Potato___Late_blight	0.89	0.98	0.94	485
Potato___healthy	0.92	0.99	0.95	456
Raspberry___healthy	0.99	0.93	0.96	445
Soybean___healthy	1.00	0.96	0.98	505
Squash___Powdery_mildew	0.98	0.99	0.99	434
Strawberry___Leaf_scorch	0.94	0.98	0.96	444
Strawberry___healthy	0.93	1.00	0.96	456
Tomato___Bacterial_spot	0.94	0.96	0.95	425
Tomato___Early_blight	0.88	0.93	0.90	480
Tomato___Late_blight	0.96	0.86	0.91	463
Tomato___Leaf_Mold	1.00	0.91	0.95	470
Tomato___Septoria_leaf_spot	0.96	0.83	0.89	436
Tomato___Spider_mites Two-spotted_spider_mite	0.99	0.88	0.93	435
Tomato___Target_Spot	0.94	0.88	0.91	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.96	1.00	0.98	490
Tomato___Tomato_mosaic_virus	0.95	0.99	0.97	448
Tomato___healthy	0.99	0.96	0.98	481
accuracy			0.96	17572
macro avg	0.96	0.96	0.96	17572
weighted avg	0.96	0.96	0.96	17572

Figure 32 : Rapport de classification

4.1.2. Matrice de confusion :

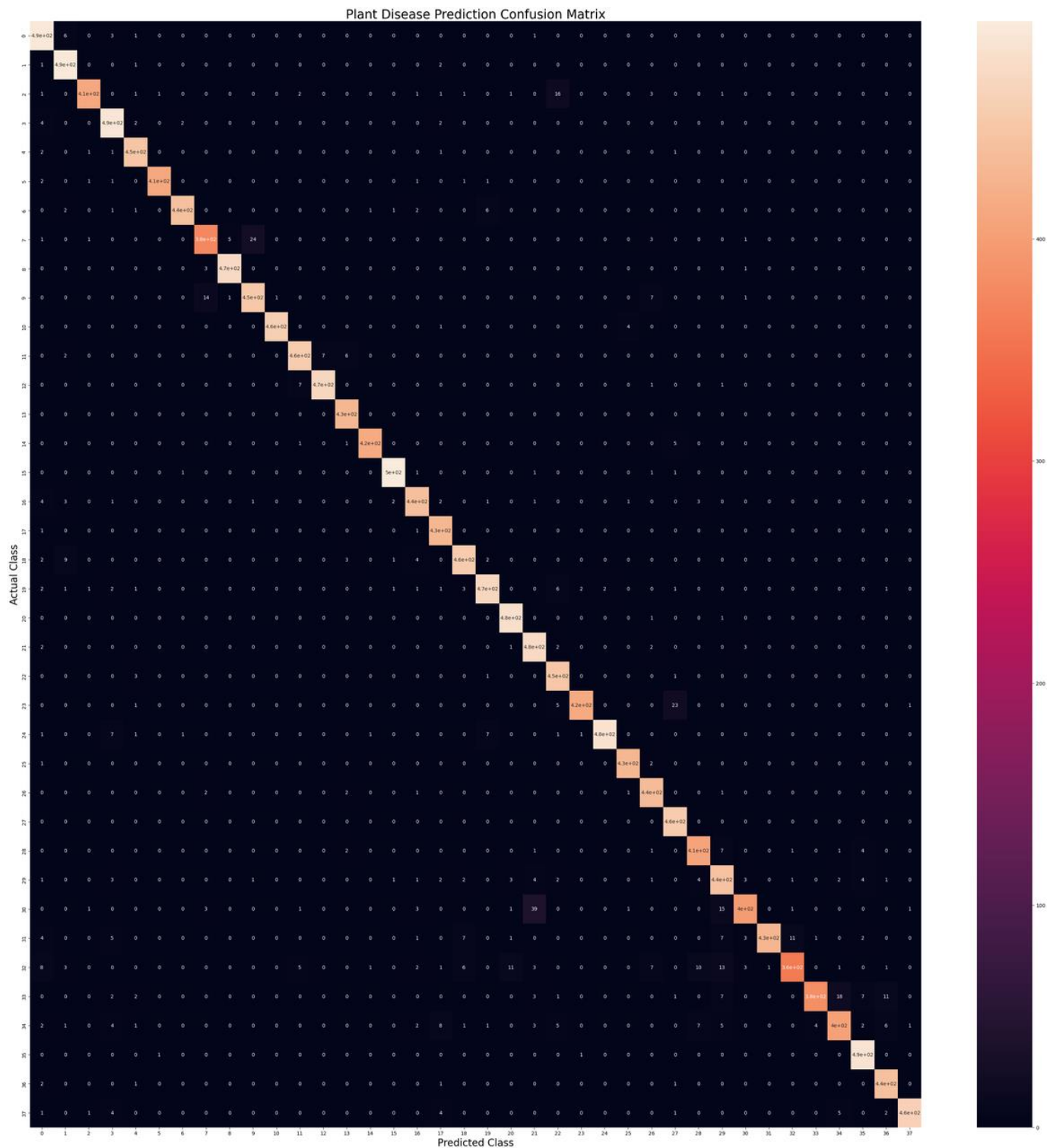
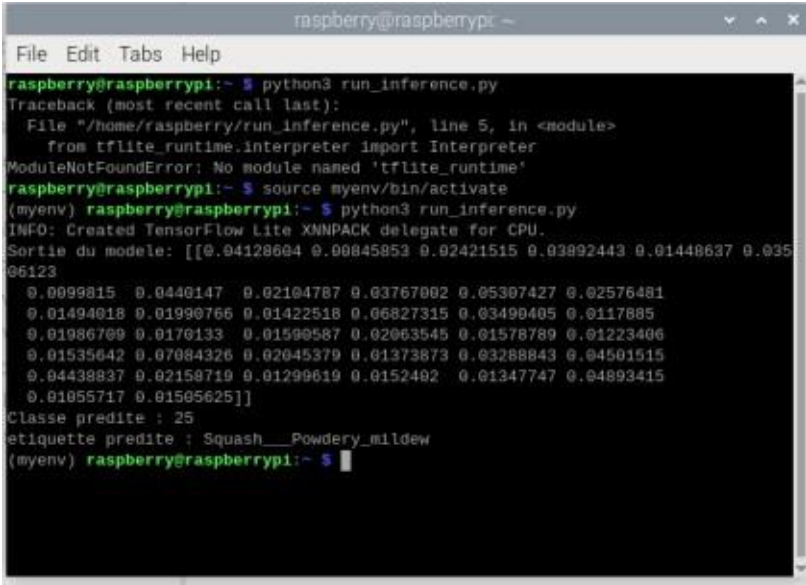


Figure 33 : Matrice de confusion

4.2. Résultat final de déploiement de projet:

Les figures suivantes illustre l'exécution réussie de script python sur le Raspberry Pi, permettant le déploiement de notre modèle de classification des feuilles et l'affichage en temps réel des mesures d'humidité et de température.

4.2.1. Déploiement de modèle TFlite sur le Raspberry Pi 5 :



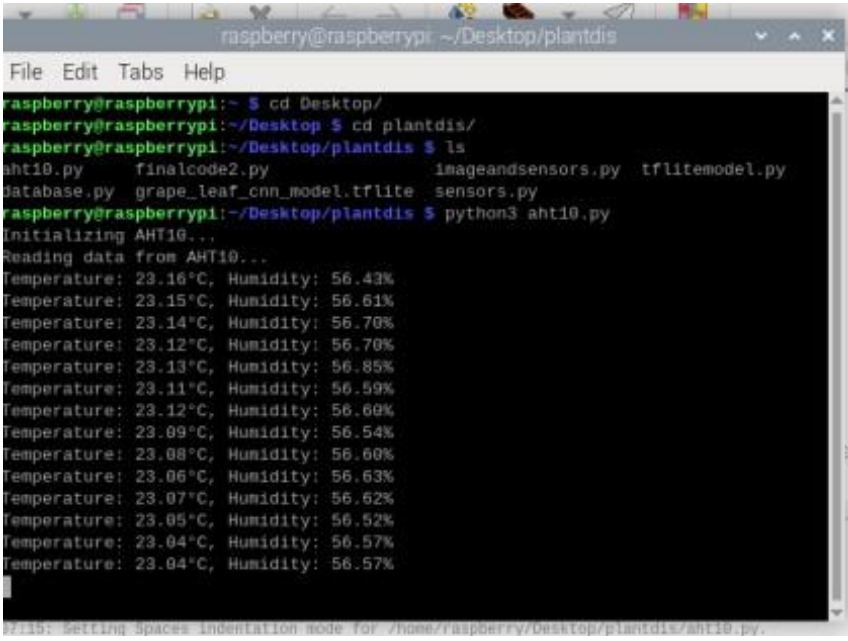
```

raspberrypi@raspberrypi ~$ python3 run_inference.py
Traceback (most recent call last):
  File "/home/raspberrypi/run_inference.py", line 5, in <module>
    from tfLite_runtime.interpreter import Interpreter
ModuleNotFoundError: No module named 'tfLite_runtime'
raspberrypi@raspberrypi ~$ source myenv/bin/activate
(myenv) raspberrypi@raspberrypi ~$ python3 run_inference.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Sortie du modele: [[0.04128604 0.00845853 0.02421515 0.03892443 0.01440637 0.03506123
0.0099815 0.0440147 0.02104787 0.03767002 0.05307427 0.02576481
0.01494018 0.01990766 0.01422518 0.06827315 0.03490405 0.0117885
0.01986709 0.0170133 0.01590587 0.02063545 0.01578789 0.01223406
0.01535642 0.07084326 0.02045379 0.01373873 0.03288843 0.04501515
0.04438837 0.02150719 0.01290619 0.0152402 0.01347747 0.04893415
0.01055717 0.01505625]]
Classe predite : 25
etiquette predite : Squash___Powdery_mildew
(myenv) raspberrypi@raspberrypi ~$

```

Figure 34 : Déploiement de modèle TFlite sur le Raspberry Pi 5

4.2.2. Résultat de mesure temps réel d'humidité et température :



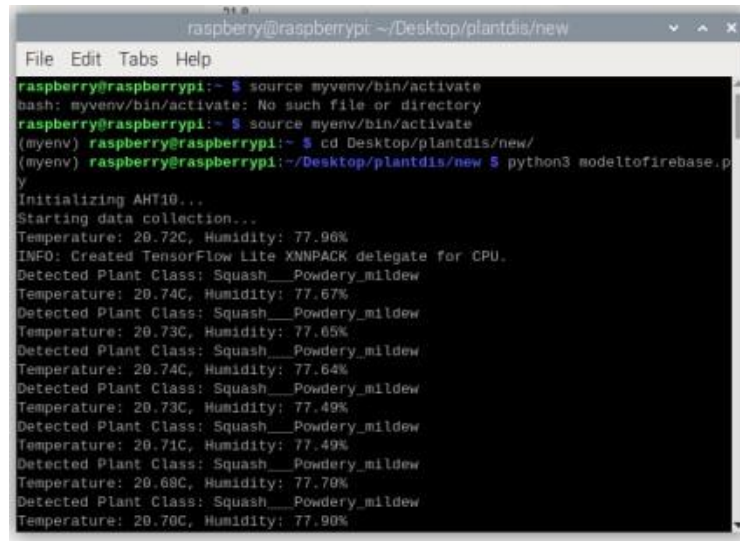
```

raspberrypi@raspberrypi ~$ cd Desktop/
raspberrypi@raspberrypi:~/Desktop$ cd plantdis/
raspberrypi@raspberrypi:~/Desktop/plantdis$ ls
aht10.py  finalcode2.py  imageandsensors.py  tflitemodel.py
database.py  grape_leaf_cnn_model.tflite  sensors.py
raspberrypi@raspberrypi:~/Desktop/plantdis$ python3 aht10.py
Initializing AHT10...
Reading data from AHT10...
Temperature: 23.16°C, Humidity: 56.43%
Temperature: 23.15°C, Humidity: 56.61%
Temperature: 23.14°C, Humidity: 56.70%
Temperature: 23.12°C, Humidity: 56.70%
Temperature: 23.13°C, Humidity: 56.85%
Temperature: 23.11°C, Humidity: 56.59%
Temperature: 23.12°C, Humidity: 56.60%
Temperature: 23.09°C, Humidity: 56.54%
Temperature: 23.08°C, Humidity: 56.60%
Temperature: 23.06°C, Humidity: 56.63%
Temperature: 23.07°C, Humidity: 56.62%
Temperature: 23.05°C, Humidity: 56.52%
Temperature: 23.04°C, Humidity: 56.57%
Temperature: 23.04°C, Humidity: 56.57%

```

Figure 35 : Mesure temps réel d'humidité et température

4.2.3. Résultat d'exécution de script final de tout le projet :



```

raspberrypi@raspberrypi: ~/Desktop/plantdis/new
File Edit Tabs Help
raspberrypi@raspberrypi:~$ source myenv/bin/activate
bash: myenv/bin/activate: No such file or directory
raspberrypi@raspberrypi:~$ source myenv/bin/activate
(myenv) raspberrypi@raspberrypi:~$ cd Desktop/plantdis/new/
(myenv) raspberrypi@raspberrypi:~/Desktop/plantdis/new$ python3 modeltofirebase.py
Initializing AHT10...
Starting data collection...
Temperature: 20.72C, Humidity: 77.96%
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.74C, Humidity: 77.67%
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.73C, Humidity: 77.65%
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.74C, Humidity: 77.64%
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.73C, Humidity: 77.49%
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.71C, Humidity: 77.49%
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.68C, Humidity: 77.70%
Detected Plant Class: Squash__Powdery_mildew
Temperature: 20.70C, Humidity: 77.98%
  
```

Figure 36 : Résultat d'exécution de script final du projet

4.2. Interface utilisateur du projet:

Notre interface Node-RED Dashboard permet d'affiche les données collectées par le capteur et les résultats du modèle de détection des maladies.

- **Graphique de température** : Visualise les valeurs de température dans le temps.
- **Plant Class** : Prédiction du modèle indiquant une maladie.
- **Jauge d'humidité** : Affiche l'humidité actuelle à 75.66%.

Cette interface permet une visualisation en temps réel des informations cruciales.

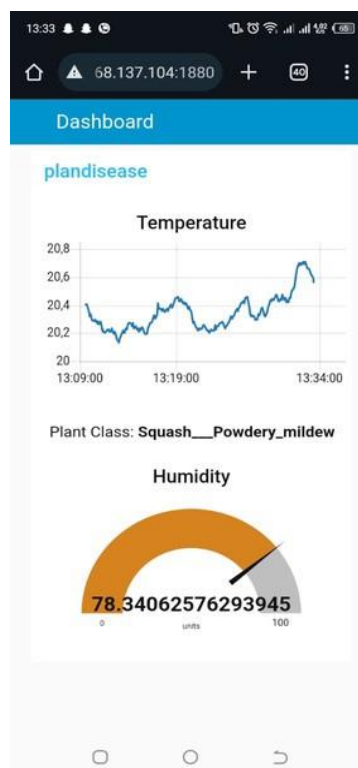


Figure 37 : Interface Utilisateur du projet

5. Conclusion :

Ce projet d'agriculture intelligente a permis de concevoir un système IoT innovant basé sur le Raspberry Pi, combinant plusieurs technologies pour répondre aux besoins de surveillance et d'optimisation dans le domaine agricole. Grâce à l'intégration de capteurs environnementaux tels que l'AHT10, nous avons pu mesurer en temps réel des paramètres essentiels comme la température et l'humidité, offrant ainsi un suivi précis des conditions climatiques.

Par ailleurs, l'utilisation d'un modèle de classification CNN a permis de détecter les maladies des feuilles avec une grande précision, facilitant une intervention rapide et efficace pour préserver la santé des cultures. Les données collectées ont été centralisées et visualisées via une interface Node-RED connectée à Firebase, assurant une accessibilité facile et une prise de décision optimale pour les agriculteurs.

En résumé, ce projet démontre l'impact positif des technologies IoT et de l'intelligence artificielle dans l'agriculture moderne. Il ouvre la voie à des solutions plus avancées, visant une gestion intelligente et durable des cultures pour répondre aux défis actuels du secteur agricole.