République Tunisienne
*****
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
*****
Université de Monastir
*****
Institut Supérieur d'Informatique et de Mathématiques de Monastir
*****
Département de Technologie

Projet Tutoré

Présenté par

RHIM Majdi

ZBIDI Chaima

La détection de fuite de gaz et la prédiction de l'incendie à partir des images en utilisant le Machine Learning

Mr MAHOUACH AYOUB **Encadrant**

Année Universitaire : 2021/2022

Remerciements

Dieu Merci !

C'est avec plaisir que nous réservons ces quelques lignes en signe de gratitude et de profonde Reconnaissance à l'égard de tous ceux qui nous ont aidés dans la réalisation de notre projet Tutoré.

Nous tenons particulièrement à exprimer en premier lieu nos profondes reconnaissances, nos Respects et nos sincères remerciements à notre encadrant Monsieur Mahouach Ayoub Pour son assistance permanente, son soutien constant dans le suivi de différentes étapes de la Réalisation du projet, ses conseils et critiques fructueux qui nous ont aimablement guidés dans Nos efforts et aussi pour sa compréhension, sa disponibilité et sa bienveillance durant toute Cette période de travail.

Nous remercions aussi tout le corps enseignant et administratif qui a contribué à notre Formation universitaire.

Table des matières

LISTE DES FIGURES

Avant-Propos

The Internet of Things :

« L'Internet of Things (IOT) décrit le réseau de terminaux physiques, les « objets », qui intègrent des capteurs, des softwares et d'autres technologies en vue de se connecter à d'autres terminaux et systèmes sur Internet et d'échanger des données avec eux. Ces terminaux peuvent aussi bien être de simples appareils domestiques que des outils industriels d'une grande complexité. Avec plus de 7 milliards de terminaux IOT connectés aujourd'hui, les experts s'attendent à ce que ce nombre passe à 10 milliards d'ici 2020 et 22 milliards d'ici 2025.», **Oracle**

The Computer Vision :

« La vision par ordinateur est un domaine scientifique et branche de l'intelligence artificielle qui traite de la façon dont les ordinateurs peuvent acquérir une compréhension de haut niveau à partir d'images ou de vidéos numériques. Du point de vue de l'ingénierie, il cherche à comprendre et à automatiser les tâches que le système visuel humain peut effectuer.», **Wikipédia**

La Limite inférieur d'explosivité (LIE) :

« Le méthane est le gaz plus communément connu sous le nom de gaz naturel. La limite inférieure d'explosivité est de 5%. Cela signifie qu'un mélange avec juste assez de gaz naturel pour s'enflammer est composé de : 5% de méthane et 95% d'air, c'est-à-dire 5% de méthane, 19,95% d'oxygène et 75,05% d'azote. La limite supérieure d'explosivité est de 15%. Les valeurs des deux limites sont arrondies. Parfois, dans la littérature, les chiffres trouvés peuvent être légèrement différents. »

Substance	LIE ou LII (Vol %)	LSE ou LSI (Vol %)
Hydrogène	4	75
Monoxyde de carbone	12,5	74
Methane	5	15
Ethane	3	12,4
Propane	2,1	9,5
Butane	1,8	8,4
Methanol	6,7	36
Ethanol	3,3	19
Acetone	2,6	13

Chapitre 1 : Etude et spécification des besoins

I. INTRODUCTION :

Dans ce chapitre on va attaquer l'ensemble des problèmes qu'elle nous a poussé à penser à une solution, ainsi que la présentation générale et l'architecture de cette dernière.

II. PROBLEMATIQUE :

« Une explosion au gaz a eu lieu, tôt ce matin, dans une maison du quartier d'Ibn Sina à Tunis, ce qui a entraîné la mort d'un homme sur place et d'un bébé par une projection de débris alors qu'il se trouvait dans sa poussette dans la rue. L'accident a également fait deux autres victimes grièvement blessés, dont un jeune homme et une femme ».

Cet article est un échantillon de plusieurs cas que ce soit dans les maisons ou bien dans les industries dans la Tunisie et dans le monde entier. L'ensemble des solutions présentes surtout en Tunisie est à la base de la détection de fuite et une alarme. L'absence de la portabilité, le suivie en temps réel et la notification automatique de la protection civile en cas d'urgence sont des problèmes majeurs de cette solution

III. CAHIER DE CHARGE :

On a pensé à réaliser un projet pour trouver une solution optimale qui peut détecter le gaz (LPG) ainsi que la surveillance en temps réel de la part de l'utilisateur et la protection civile avec la détection de l'incendie si elle est présente. Notre solution est caractérisée par sa portabilité, l'alerte instantanée en cas d'urgence et la détection automatique de la fuite et l'incendie à l'aide de l'intelligence artificielle.

IV. ARCHITECTURE DE LA SOLUTION PROPOSEE :

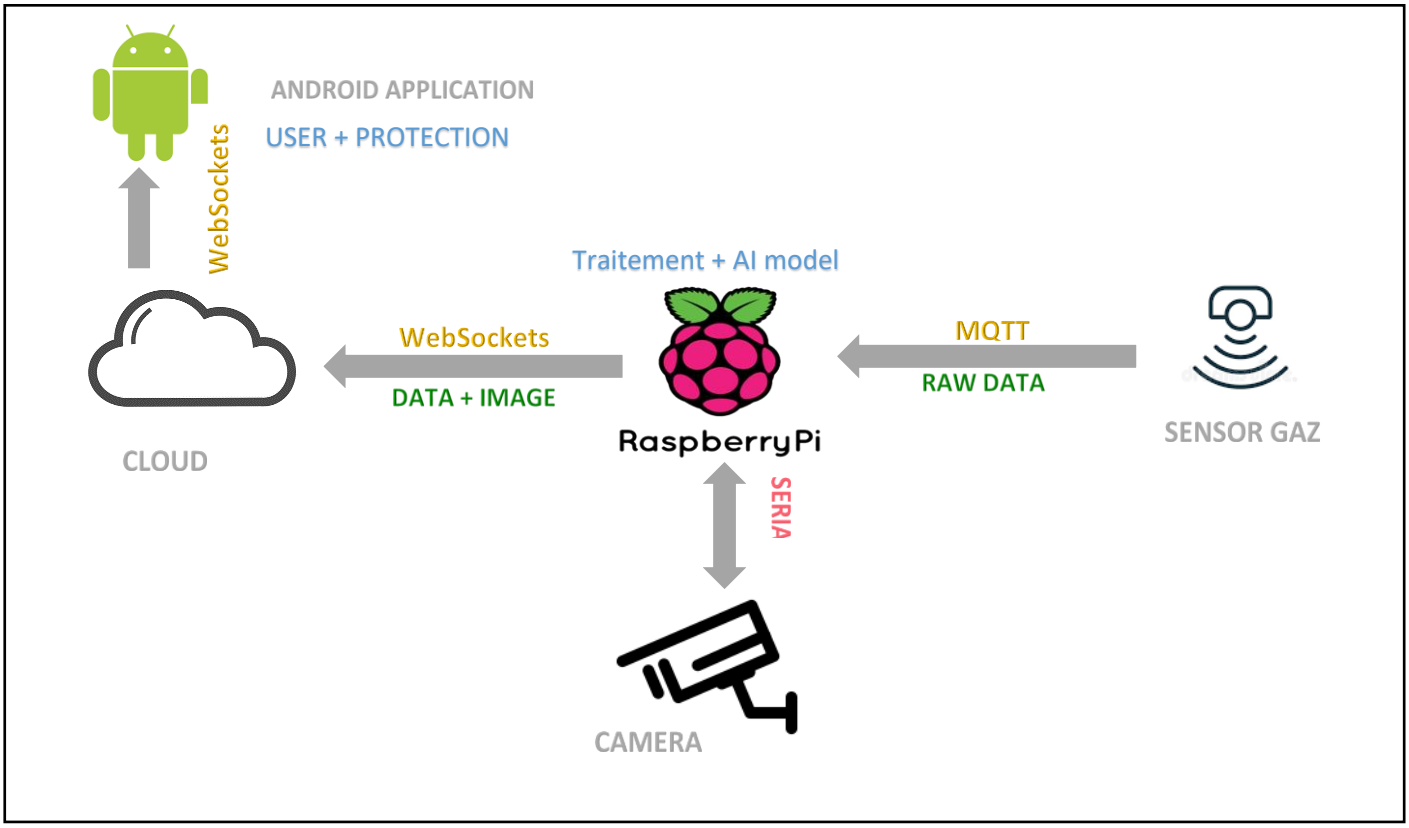


Figure 1: Architecture de Solution proposée

On peut diviser l'architecture en deux parties comme ceux-ci :

- Sensor data :

Notre capteur MQ9 branchés avec un microcontrôleur « ESP32 WROOM32 » publie les données de la concentration du gaz LPG via le protocole MQTT sur un topic, le Raspberrypi « subsicriber » écoute sur le même topic où le traitement des données et l'adaptation pour l'envoyer au cloud « Firebase » via HTTP .Une application Android est développée pour récupérer et afficher la concentration de gaz en temps réel.

- Image data :

La caméra est activée si une condition est validée sur la concentration de LPG pour capturer des images et les envoyer vers un modèle de classification d'image « Tensorflow ». Le modèle computer vision va décider la présence d'un incendie ou non. Si oui, le système envoie l'image capturée de l'incendie vers le cloud puis l'application Android la récupère.

V. CONCLUSION :

Dans ce chapitre on a présenté le problème des fuites du gaz et ses conséquences dans le démotique et les industries, ainsi que les solutions actuelles et ses défis. Aussi on a proposé notre projet et son architecture globale.

Chapitre 2 : Conception et développement

I. INTRODUCTION

Dans ce chapitre nous abordons les composants utilisé, les protocoles et les outils de développement que nous a aidé à réaliser une solution portable, stable et performante.

II. LOGICIELS ET OUTILS DE DEVELOPPEMENT UTILISE :

Partie Hardware :

- **CAPTEUR DE GAZ :**

La plupart des fuites sont provenant du gaz naturel (LPG), alors on a décidé de sélectionner le capteur de gaz MQ9, à travers la variation de sa résistance interne le capteur est capable de détecter plusieurs gaz combustible comme méthane (CH4), monoxyde de carbone (CO) et gaz naturel (LPG).

Figure 2: Capteur du gaz MQ9

Ce capteur admet 4 pins dont 2 pour l'alimentation (VCC, GND)
Tension d'alimentation : 5 ± 0,1 V.
Et 2 pins pour de sorties (analogique et numérique).
Plage de détection : 10 à 1 000 ppm de CO, 100 à 10 000 ppm de gaz combustible.
Température et hygrométrie optimales de fonctionnement : 18 à 22 °C, 65 ± 5 % HR.

- **LE CARTE ESP32 WROOM32** *:*

L'ESP32 est un microcontrôleur fabriqué par la multinationale chinoise Expressif system créée en 2008, L'ESP32 est le successeur de l'ESP8266 avec

beaucoup d'améliorations faites au niveau du hardware et du software, Il ajoute un processeur, Wi-Fi plus rapide, plus GPIO et prend en charge Bluetooth 4.0 et Bluetooth basses énergies. Alors pour cela nous avons choisi le ESP32

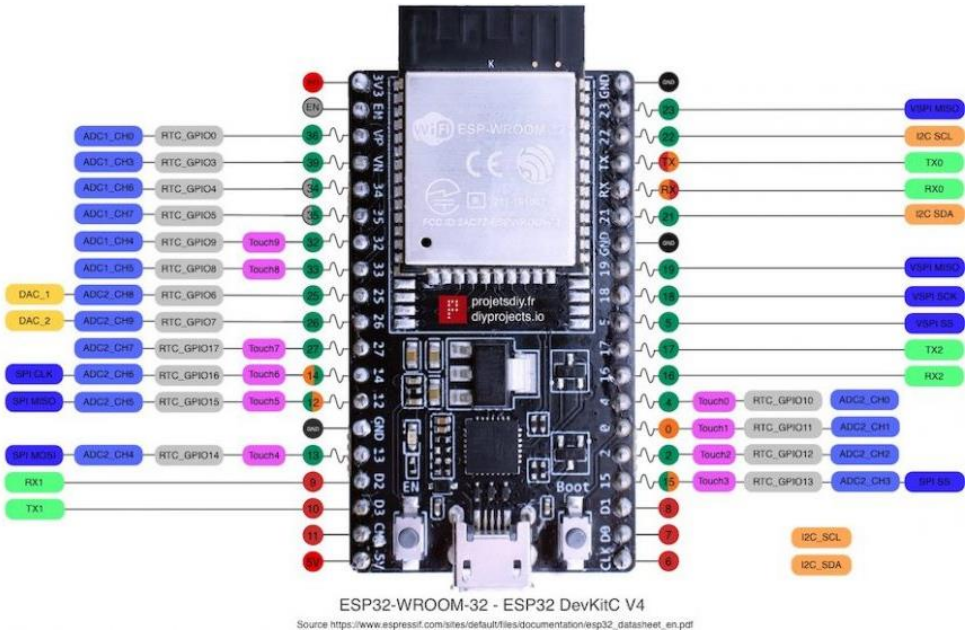


Figure 3 : datasheet de la carte ESP32

Cette carte a une caractéristique que l'on veut représenter ici :
Microprocesseur : Tensilica Xtensa Dual-Core 32-bit LX6
Fréquence d'horloge : 80 à 240 MHz (réglable)
Mémoire Flash : 4MB
ROM : 448 KB / SRAM : 520 KB
Bluetooth 4.0 (BLE/Bluetooth Smart) and Bluetooth Classic (BT)
Alimentation : 5 à 9V
Entrées/sorties digitales : 48 GPIO pins
Courant par broche E/S : 40 mA
Courant par broche 3.3V : 50 mA

- **RASPBERRY PI 3 :**

Le Raspberry Pi3 est un ordinateur mono-carte pouvant se connecter à un téléviseur, à un clavier et disposant d'une connectivité Wi-Fi et Bluetooth. Il se boote depuis la carte micro-SD et fonctionne sous OS Linux ou Windows 10 IOT. Il est fourni sans boîtier, alimentation, clavier, écran et souris dans le but de diminuer le coût et de favoriser l'utilisation de matériel de récupération.

La version 3 est basée sur un processeur ARM Cortex-A53 64 bits quatre cœurs à 1,2 G. Hz (environ 10x plus rapide que le Pi1 et 50% plus performante que le modèle Pi2) et possède 1 GB de mémoire RAM, pour cela nous avons choisi la version 3.

Figure 4 : Carte Raspberry PI 3

Cette carte a une caractéristique que l'on veut représenter ici :

Alimentation à prévoir : 5 Vcc/maxi 2,5 A* via prise micro-USB (* intensité maxi si toutes les fonctions sont utilisées)

CPU : ARM Cortex-A53 quatre cœurs 1,4 GHz

Wi-Fi : Dual-band 2,4 et 5 GHz, 802.11b/g/n/AC (Broadcom BCM43438)

Bluetooth 4.2 (Broadcom BCM43438)

Mémoire : 1 GB LPDDR2

Ethernet 10/100/1000 : jusqu'à 300 Mbps

4 ports USB 2.0

Port Ethernet 10/100 base T : RJ45

Bus : SPI, I2C, série

Support pour cartes micro-SD

Sorties audio :
- HDMI avec gestion du 5.1
- Jack 3,5 mm en stéréo

Sorties vidéo : HDMI

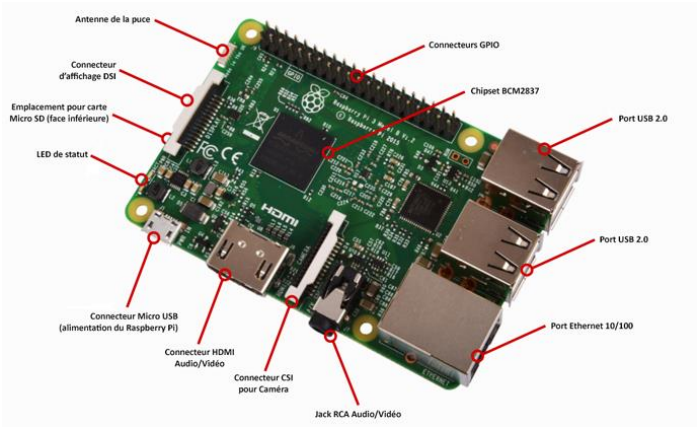


Figure 5 : composants de la carte Raspbberry PI 3

- **CAMERA RASPBERRY PI NOIR V2 :**

Module caméra miniature 8 MP pour Raspberry Pi3 ou Pi4 basé sur un capteur Sony IMX219 de 8 mégapixels. Ce module se raccorde facilement, via une nappe incluse, sur le connecteur CSI d'une carte Raspberry Pi.
L'interface CSI utilisée est conçue spécialement pour les caméras et permet d'atteindre des débits importants.
Cette caméra est compatible avec les versions Raspberry Pi 3B+ et 4B.

Figure 6 : caméra Raspberry V2

Caractéristique :
Petite carte : 25mm x 23mm x 9mm

Optique : 1/4"
Senseur 8 Mégapixels (3280×2464 pixels) Sony IMX219 avec un module à Focus fixé
Support 1080p30, 720p60 et enregistrement vidéo 640x480p60/90
1.4 µm X 1.4 µm par pixel avec technologie OmniBSI pour haute performance (grande sensibilité, faible bruit, faible interférence)

Partie Software :

- **Node RED :**

Node-RED est un outil de développement basé sur les flux pour la programmation visuelle développé à l'origine par IBM pour connecter des périphériques matériels, des API et des services en ligne dans le cadre de l'Internet des objets

Figure 7: Node-RED

- **MOSQUITTO BROCKER :**

Mosquitto est un serveur MQTT Open Source (Broker) que l'on peut installer sur un Raspberry Pi mais aussi sur presque tous les systèmes d'exploitation (MacOs, Windows, Linux…).

Figure 8 : Mosquitto

MQTT est un protocole de communication très rapide et léger particulièrement bien adapté à la domotique et aux objets connectés. Il facilite la communication entre objets connectés (M2M) tout en économisant la batterie.

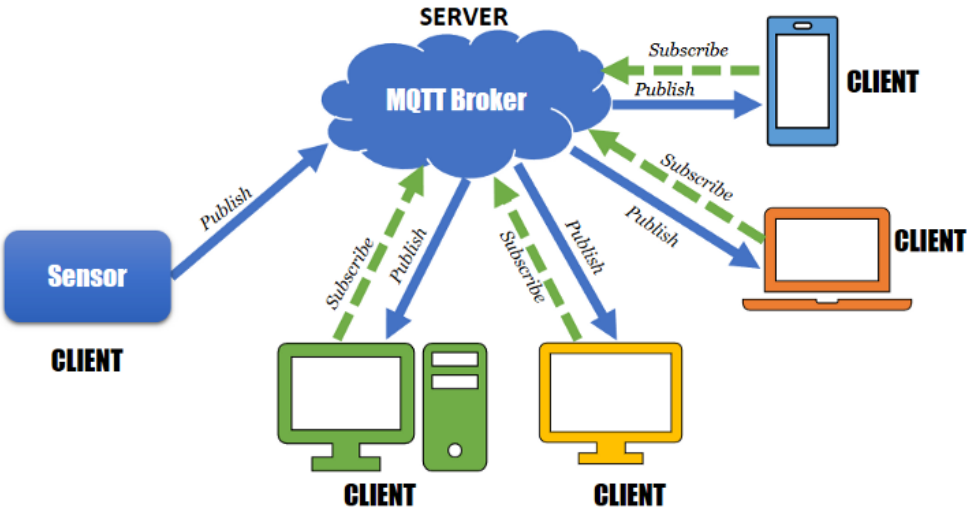


Figure 9 : Serveur MQTT broker

- TENSORFLOW :

TensorFlow est une plate-forme Open Source de bout en bout dédiée aux machines Learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine Learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie. Ci-dessous vous pouvez voir le diagramme actuel de la plateforme TensorFlow :

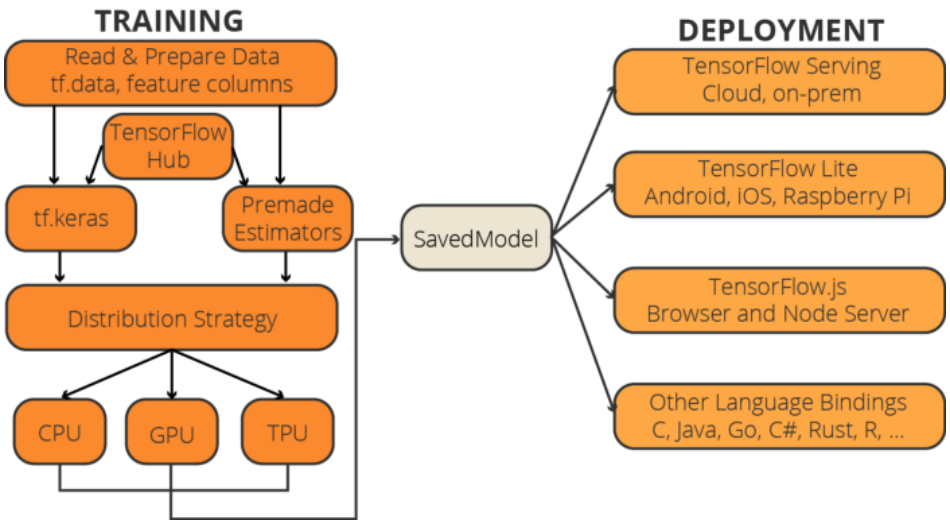


Figure 10 : Diagramme actuel de la plateforme Tensorflow

- **FIREBASE :**

Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application (Android, IOS, JavaScript, Node-JS, JAVA, Unity, PHP, C++ ...). Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de l'authentification sociale (Google, Facebook, Twitter et Github), et des notifications, ou encore des services, tel que par exemple un serveur de communication temps réel.

Lancé en 2011 sous le nom d'Envolve, par Andrew Lee et par James Templin, le service est racheté par Google en octobre 2014. Il appartient aujourd'hui à la maison mère de Google : Alphabet.

Figure 11 : Firebase

- **ANDROID STUDIO :**

Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, MacOs, Chrom OS et Linux

Figure 12 : Environnement Android Studio

Android Studio permet principalement d'éditer les fichiers Java/Kolín et les fichiers de configuration XML d'une application Android.

Il propose entre autres des outils pour gérer le développement d'applications multilingues et permet de visualiser rapidement la mise en page des écrans sur des écrans de résolutions variées simultanément. Il intègre par ailleurs un émulateur permettant de faire tourner un système Android virtuel sur un ordinateur.

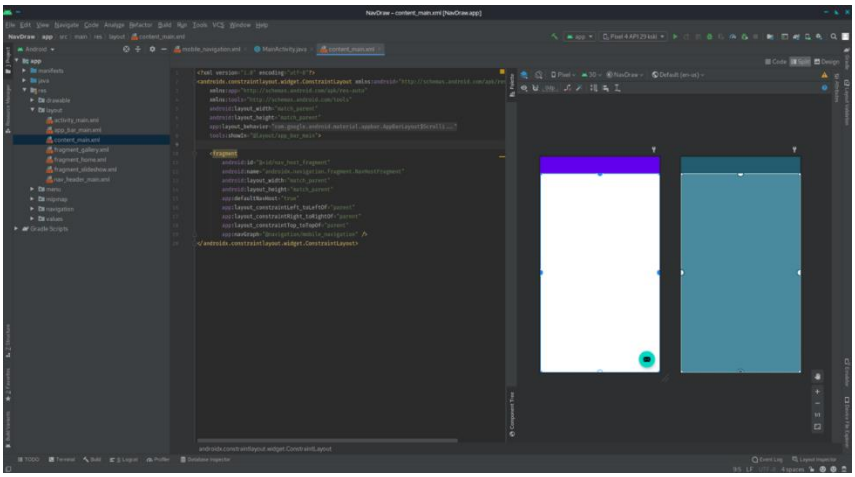


Figure 13 : l'interface de l'environnement Android Studio

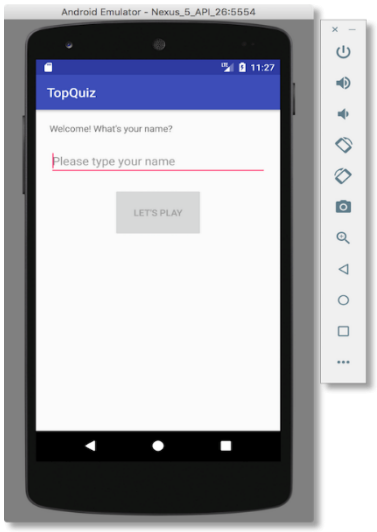


Figure 14 : Emulateur

- **ARDUINO SOFTWARE (IDE) :**

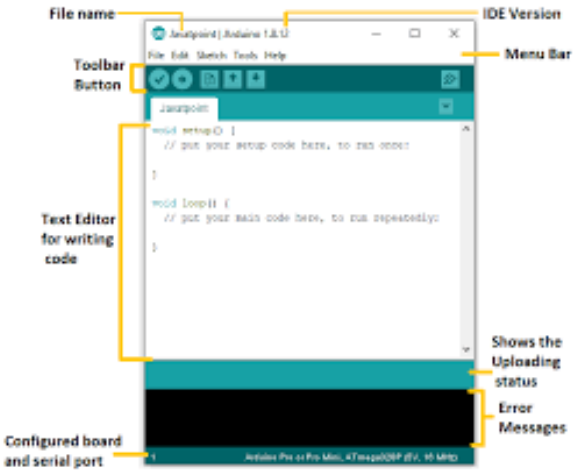
Les créateurs d'Arduino ont développé un logiciel pour que la programmation des cartes Arduino soit visuelle, simple et complète à la fois. C'est ce que l'on appelle une IDE, qui signifie *Integrated Development Environnent* ou Environnement de Développement « Intégré » en français (donc EDI). Le langage de programmation utilisée est le C++ langage Standard, ce qui rend aisé le développement des programmes sur les plateformes Arduino. Mais dans notre projet nous avons utilisé l'IDE pour programmer notre carte ESP32.

L'IDE affiche une fenêtre graphique qui contient un éditeur de texte et tous les outils Nécessaires à l'activité de programmation.

L'IDE Arduino permet :

▪D'éditer un programme : des croquis (sketch en Anglais).
▪ De compiler ce programme dans le langage « machine » de l'Arduino.
▪ De téléverser le programme dans la mémoire de l'Arduino.
▪ De communiquer avec la carte Arduino grâce au termina.

Figure 15 : Logiciel Arduino IDE



III. CONCLUSION :

Suite aux validations effectuées dans ce chapitre, nous considérons que les choix Technologiques et dimensionnels sont satisfaisants. Dans le chapitre qui suit, nous allons nous intéresser à la réalisation du système.

Chapitre 3 : Réalisation & Validation

I. INTRODUCTION :

Ce chapitre représente l'ensemble des techniques utilisé pour la réalisation et la validation de la solution, ainsi nous présentons les flows des données qui englobent notre projet.

II. Image Classification Using Tensorflow

On a utilisé la méthode de Transfer Learning ou apprentissage par transfert, selon Wikipédia « L'apprentissage par transfert est l'un des champs de recherche de l'apprentissage automatique qui vise à transférer des connaissances d'une ou plusieurs tâches sources vers une ou plusieurs tâches cibles. »

Alors on a utilisé le model pre-trained «MobileNet» créer par Google puis on a ajouté notre dataset à classifier qui est composée de deux classe la première est un ensemble d'image des maisons sans incendies et l'autre classe avec présence d'incendies. Le model est maintenant qualifié à prédire si une image est appartient au classe 1 ou 2 avec une précision de 98%.

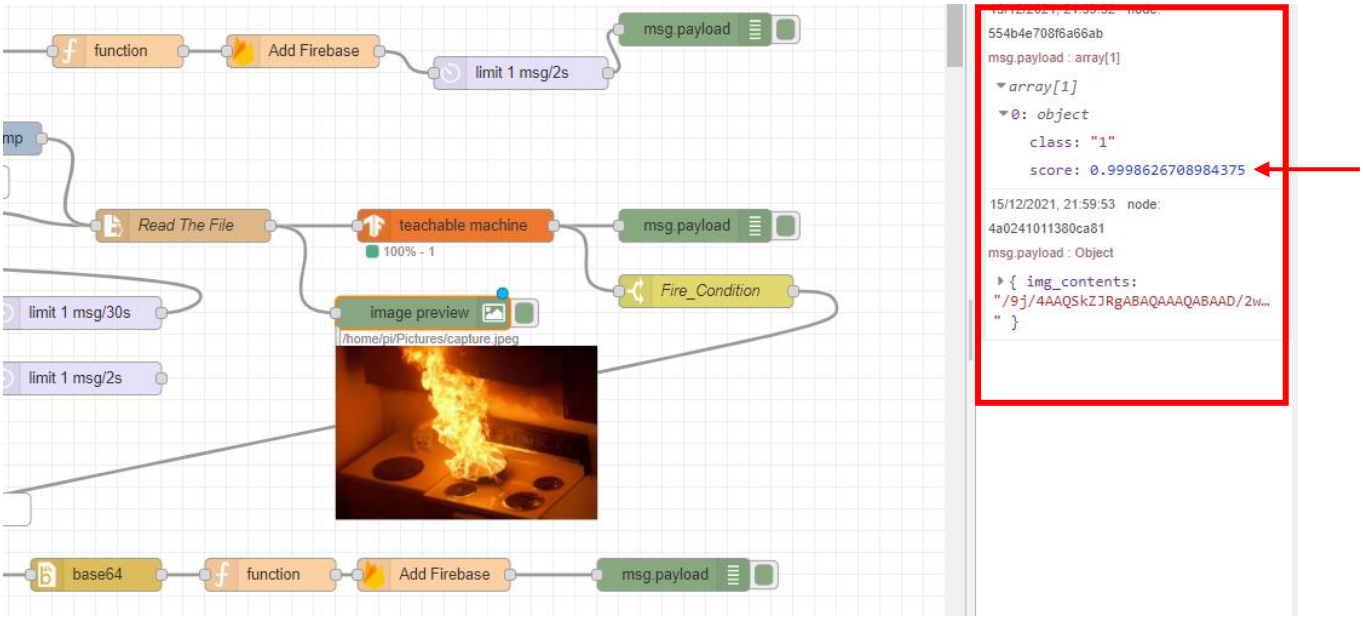


Figure 16 : model Tensoflow

III. Câblage et test de capteur du gaz MQ9 avec la carte ESP32

Tout d'abord on a commencé à faire l'étalonnage du capteur MQ9 pour mesurer le gaz naturel

Selon le diagramme donné dans la Datasheet on a calculé l'équation de la droite linéaire « ax+b » de la concentration du gaz LPG en fonction de RS/R0 :

$$X = -(y-1.3)/0.0003$$

Avec

X : ppm (concentration du gaz LPG)
Y : RS/R0 (R0 une constante, RS résistance variable sensible de l'ensemble des gaz)

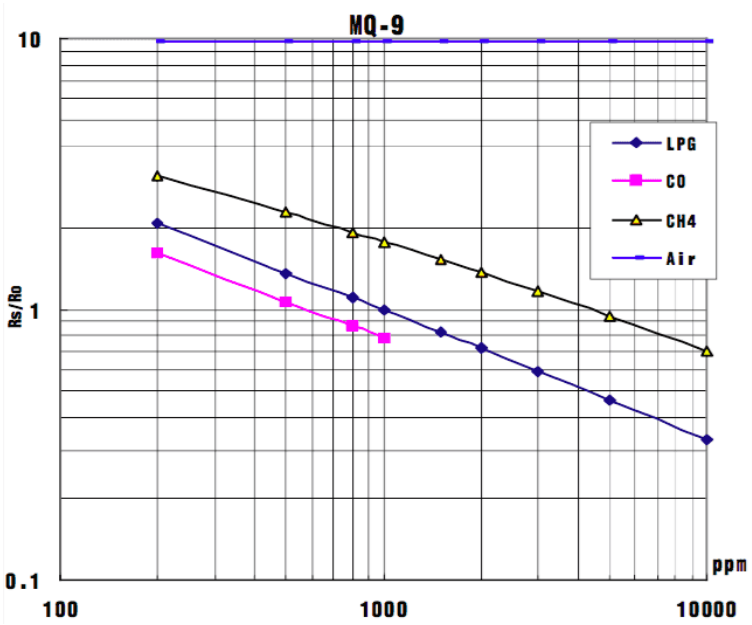


Figure 17 : E. Sensitivity characteristic curve

Etant donné que l'ESP32 admet 18 convertisseurs analogiques numériques avec une résolution de 12 bits, on a utilisé pin numéro 32 (ADC1 CH4) pour la lecture des valeurs analogiques. Après conversion les valeurs

mesurées en PPM à chaque instant sont publiées au broker (l'adresse IP de la carte Raspberry) à travers le protocole MQTT où le topic est gaz/ppm.

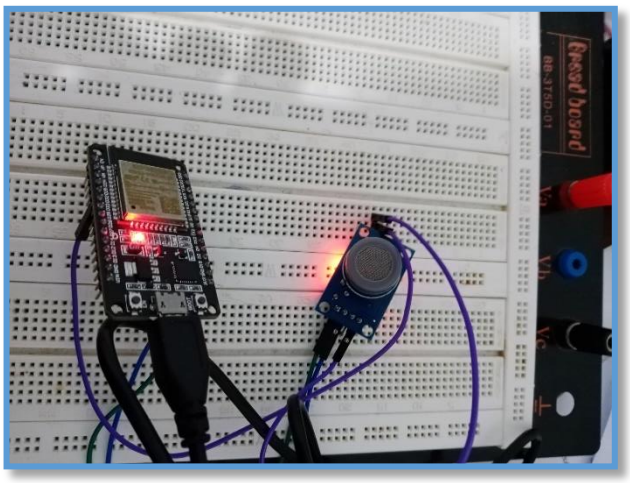


Figure 18 : câblage de capteur MQ9 avec ESP32

IV. Raspberry Pi as a service

On a nommé cette partie comme ceux-ci car le PI offre dans notre projet plusieurs services dont il représente le broker, le subscriber ainsi que la camera avec le model Tensorflow pour prédire les incendies sans oublier le service d'envoyer les données au cloud sous format JSON.

Tout cela est organisé par l'environnement Node-RED :

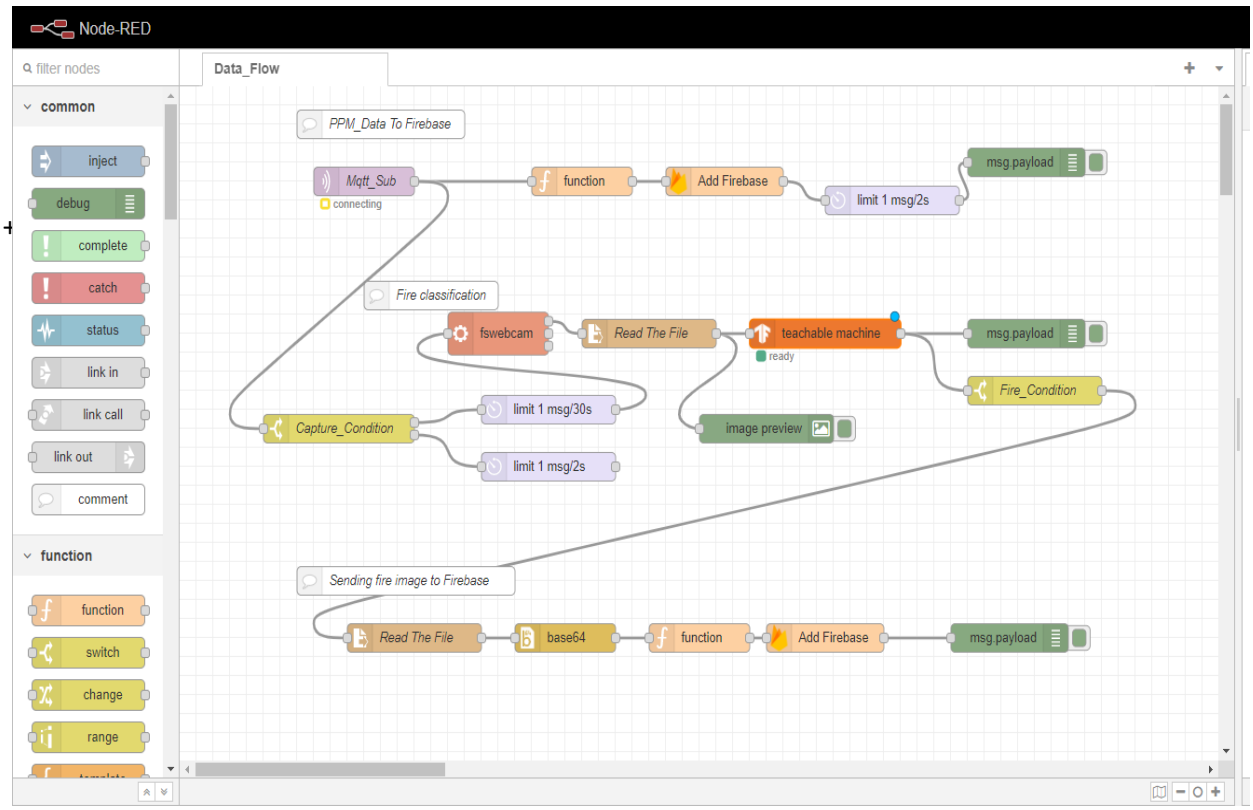


Figure 19 : Node-RED

Les flows des données commencent par l'écoute du subscriber au topic gaz/ppm pour recevoir les données envoyer par le capteur, ces données sont envoyées en temps réels au cloud (Firebase).

Ces données suivent aussi une condition sur la concentration du gaz (5000ppm) pour activer le camera et prendre une photo instantanément puis l'envoyer au model Tensorflow.

Si le model a détecté une incendie, l'image est encodée en base64 pour la compressée et l'envoyée vers le Cloud.

Figure 20 : Firebase Realtime Database

V. Application Android

Partie graphique :

On a créé la première interface qui nous amène vers deux autres interfaces à l'aide de deux boutons successivement « user » et « protection »

Figure 21 : interface d'accueil de l'application Android

La première est spécifique pour l'inscription ainsi que pour la connexion pour un utilisateur

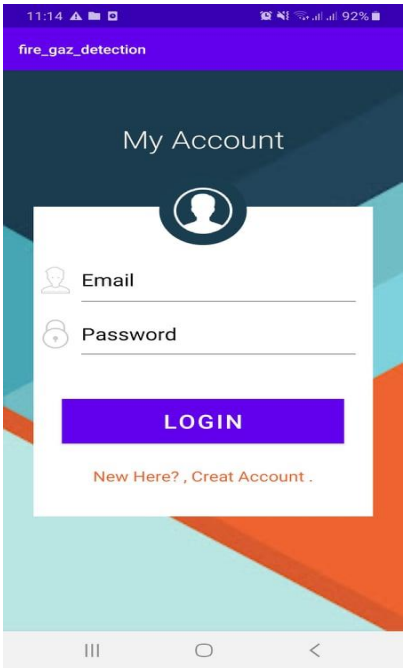


Figure 23 : interface login

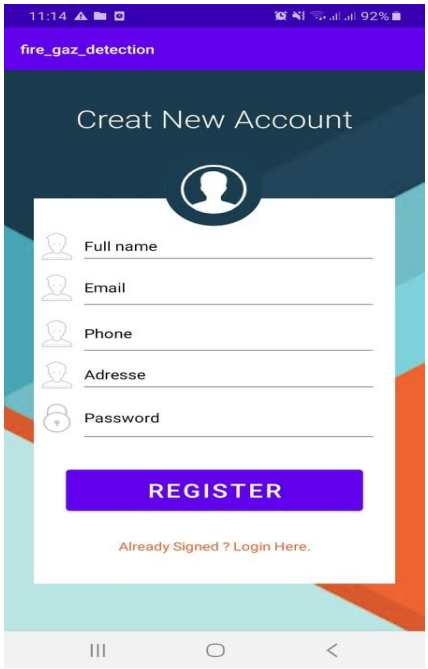


Figure 22 : interface Creat Account

La deuxième permet la connexion pour les agents de pompiers (sécurité).

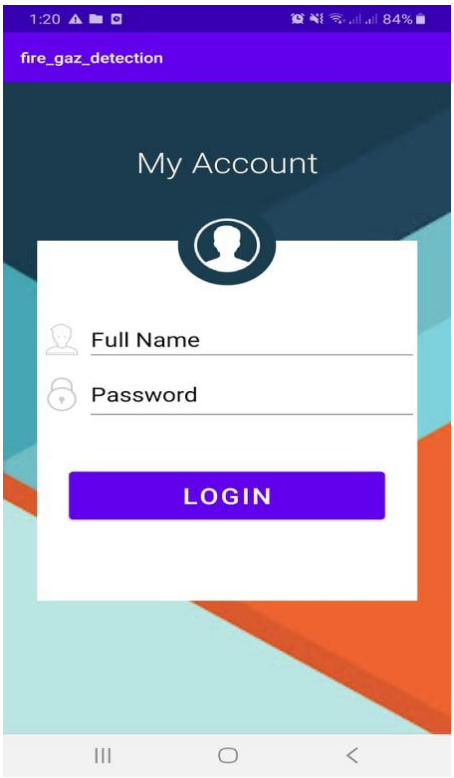


Figure 24 : interface login

La clique sur le bouton « LOGIN » dans l'interface de l'utilisateur nous amène vers une autre interface où se trouve la valeur de gaz LPG.

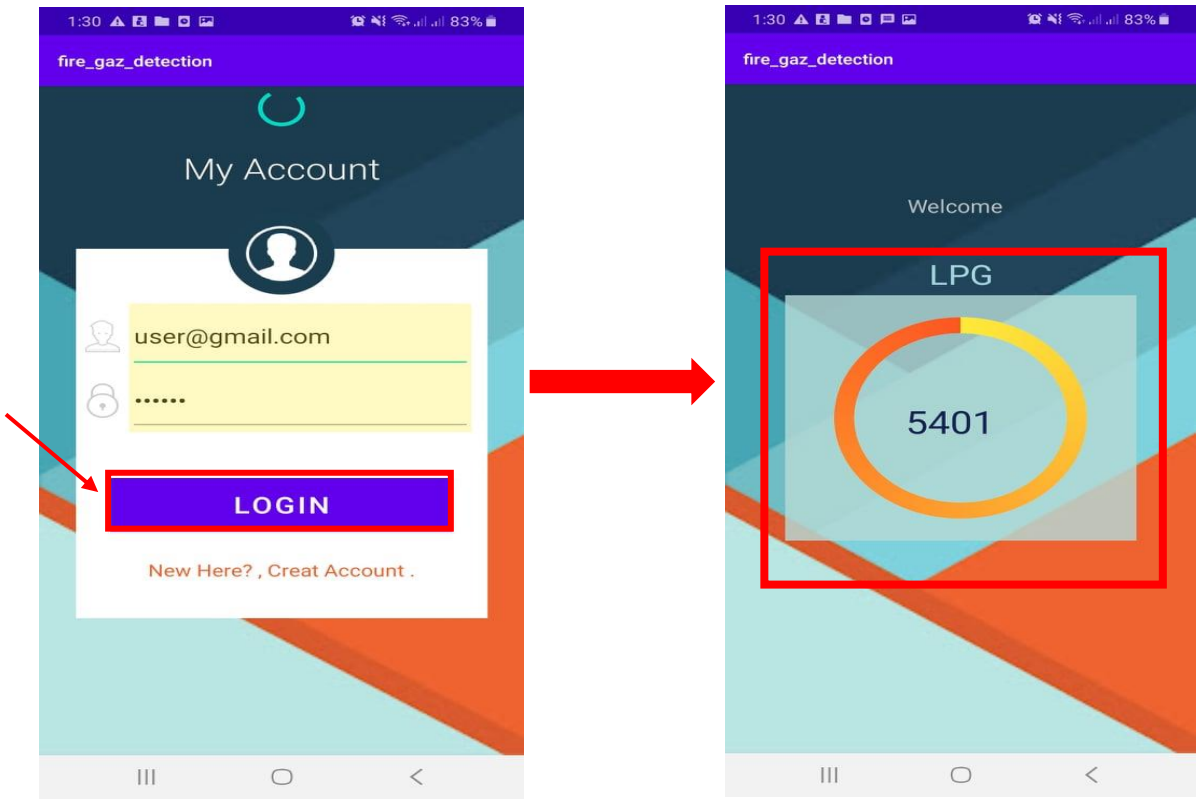


Figure 25 : récupération du gaz

Concernant l'interface « protection » maintenant, une clique sur le bouton « LOGIN » permet d'amener vers une interface où nous avons trouvé l'image de l'incident avec le nom et la localisation de l'utilisateur

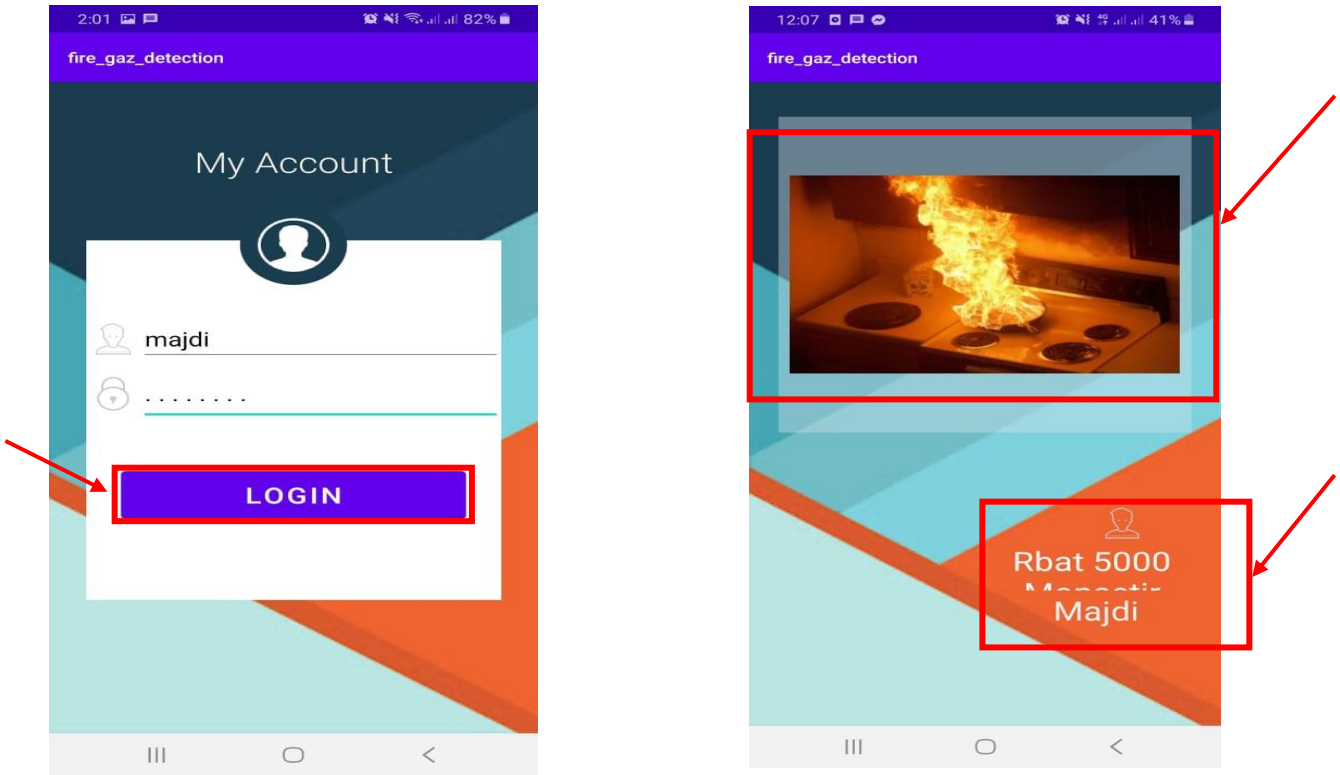


Figure 26 : récupération d'image

Partie traitement :

L'authentification de l'utilisateur est assurée par Firebase Authentification

« Cet outil fournit des SDK faciles à exploiter, des services back-end ou encore des bibliothèques d'interface utilisateur. Ces bibliothèques vous permettent d'authentifier cos utilisateurs. »

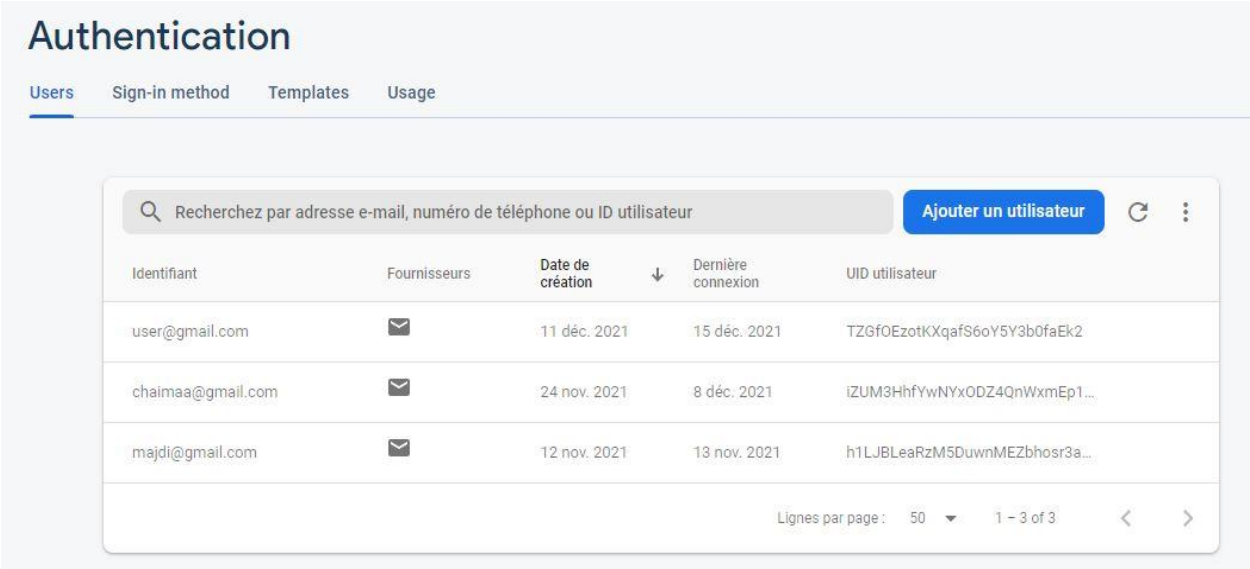


Figure 27 : Authentification Firebase

L'authentification de la protection est assurée par un mot de passe commun entre les agents de pompiers.

La récupération de la concentration du gaz LPG est assurée par Firebase Realtime Database « cet outils permettant le stockage et la synchronisation de données de vos utilisateurs, les développeurs peuvent gérer cette base de données en temps réel ». Nous avons utilisé la fonction «**OnDataChange**» qui va écouter chaque changement de valeur dans la base et le récupérer.

Figure 28 : Firebase Realtime Database

Si la concentration de gaz LPG est supérieure à 5000 ppm (c à d 10% de la limite inférieur d'explosivité LEL) une notification était déclenché pour notifier l'utilisateur qu'il y a un gaz a été détectée.

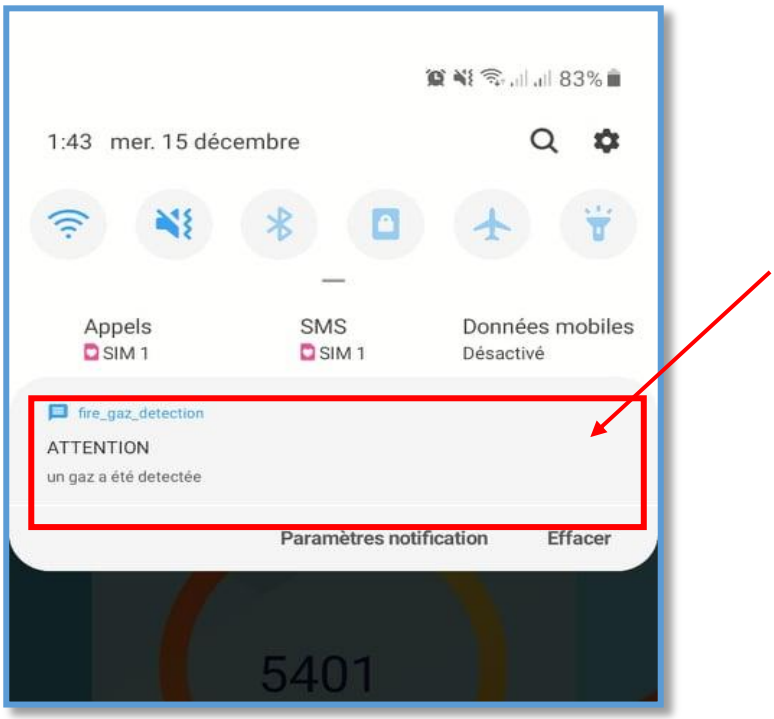
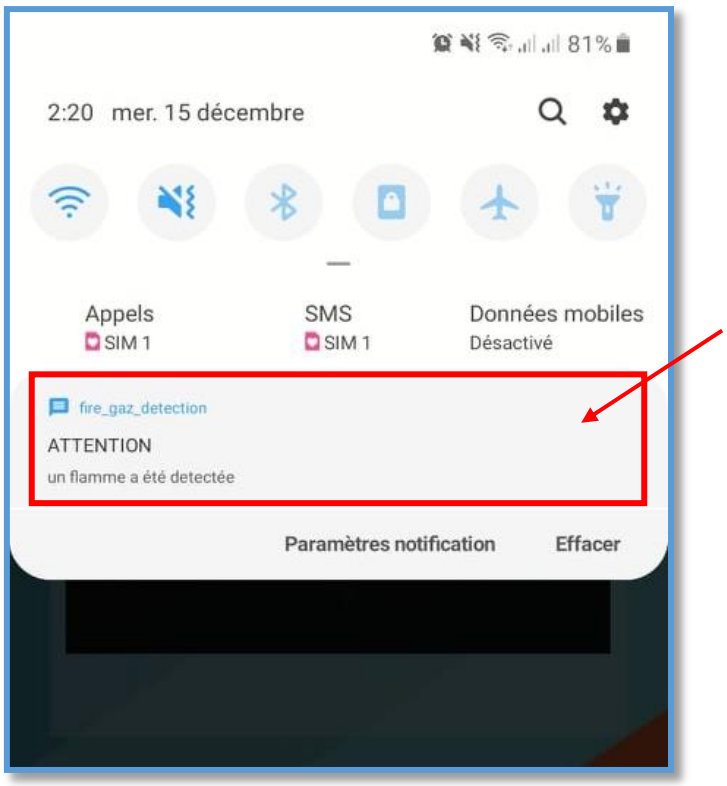


Figure 29 : notification

Concernant l'autre interface, nous avons utilisé la même fonction pour récupérer l'image vers l'application Android.

Comme nous avons précisé précédemment, l'image est stockée sous forme base64. Pour régénérer le format JPEG on a décodé la donnée reçus en un ensemble de bytes puis on a la convertit en Bitmap (image pixélisé).

Si on reçoit une image, une notification est déclenchée pour notifier les agents de pompiers qu'il y a une flamme a été détectée.



VI. CONCLUSION :

On a abordé dans ce chapitre comment les données sont mesurées, transmises, traitées et stockées.

Chapitre n° 4 : Expectation

I. Evolution du réseau en Tunisie :

Pour vraiment réaliser la notion d'alerte en temps réels il faut évoluer l'infrastructure de notre réseaux cellulaire, évidement l'installation du réseau 5G va aider à appliquer un service fiable étant donné que le débit est énormément plus rapide.

Ainsi que la notion du MEC (Mobile Edge Computing) si elle est présente va influencer de façon remarquable sur le temps de latence.

« La norme MEC (Multi-Access Edge Computing) permet de déplacer le trafic informatique et les services depuis un cloud centralisé vers un réseau de périphérie, plus proche du client. Au lieu d'envoyer toutes les données à traiter dans un cloud, le réseau de périphérie analyse, traite et stocke les données. Le fait de collecter et de traiter les données à proximité du client réduit la latence et apporte aux applications à large bande passante une performance en temps réel. **», Juniper Networks**

II. Utiliser la solution dans l'industrie

Evidement les dangers de fuite des gaz dans l'industrie sont plus risqués, on peut alors appliquer notre solution mais il faut changer le protocole de communication entre le capteur et la carte Raspberry. On peut dans ce cas utiliser une communication LoRa vu que la distance est plus large que les domotiques.

III. Evoluer la caméra

Dans la plupart des cas les maisons et les industries sont équipées d'une caméra de sécurité, on peut alors l'utilisée au lieu du module Raspberry pi. De cette façon la communication entre la carte et la caméra devient sans fils et la qualité d'image s'évolue.

On peut ainsi au lieu de prendre des images successives on utilise un « live Stream » pour la détection des incendies.

IV. L'authentification de la protection civil

Dans le future on peut créer une base de donnée local dont elle dispose le code série de chaque agent de protection civil, l'authentification se base alors sur ce code unique pour assurer la confidentialité.