REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département d'Electronique

Rapport

Projet de Fin de Cycle (PFC)

MCIL 3

FILIERE : **Electronique**

Spécialité : Industries Electroniques

Par

- **ABED OUSSAMA**
- **AMIRI SAMIR**

Intitulé

Etude et Réalisation d'un système de reconnaissance faciale basé sur une carte ESP32-cam et la librairie OpenCV pour le langage Python

Présenté le : ……26/06/2022……

Devant le Jury composé de :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
M. Boussahoul Abdelkarim	*MAA*	*Président*	*Univ-BBA*
M. Bentouhami. L	*MCB*	*Examinateur*	*Univ-BBA*
M. Abed Tarek	*MAA*	*Encadreur*	*Univ-BBA*

Année Universitaire 2021/2022

Remerciements

Nous remerciant ALLAH qui nous aide et nous donne
la patience et le courage durant ces
longes années d'étude.
Nous tiens à remercier grandement nos Encadreur
M. Abed Tarek
pour sa grande disponibilité et ses précieux conseils.
Nous remerciant également tous les enseignants du
département d'électronique d'université de BBA.
plus spécialement les ***membres de jury*** *de notre travail.*
Enfin, nous adressons nos plus sincères remerciements à tous
nos amis et collègues qui nous ont toujours
soutenue et encouragée au cours de
la réalisation de ce mémoire.
Merci à tous.

Résumé

Dans ce projet, nous avons effectué une étude et une réalisation d'un système de reconnaissance faciale autour d'une carte ESP32-CAM ainsi que les librairies destinées à cet environnement de développement. Comme seconde étape, nous nous sommes intéressés à implémenter un programme en langage python afin de tester les fonctions de reconnaissance faciale de la librairie OpenCV.

Mots-clés : reconnaissance faciale, ESP32-CAM, environnement de développement, Python, bibliothèque OpenCV.

ملخص

في هذا المشروع، أجرينا دراسة وإنجاز لنظام التعرف على الوجه القائم أساسا حول بطاقة ESP32-CAM وكذلك المكتبات المخصصة لبيئة التطوير هذه. كخطوة ثانية، كنا مهتمين بتنفيذ برنامج بلغة Python من أجل اختبار وظائف التعرف على الوجه في مكتبةOpenCV .

كلمات مفتاحية: التعرف على الوجه، ESP32-CAM، بيئة التطوير، Python، مكتبةOpenCV .

Abstract

In this project, we carried out a study and a realization of a facial recognition system around an ESP32-CAM card as well as the libraries intended for this development environment. As a second step, we were interested in implementing a program in python language in order to test the facial recognition functions of the OpenCV library.

Keywords: facial recognition, ESP32-CAM, development environment, Python, library OpenCV.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

Introduction générale

La recherche sur la reconnaissance faciale a commencé dans les années 1960 et avec le développement de la technologie d'imagerie informatique et optique, la phase d'application réelle a commencé à la fin des années 1990 dans certains pays développés du monde et la technologie de reconnaissance faciale est connue comme une technologie sophistiquée qui aide à identifier et à reconnaître les visages humains grâce à des images ou à un système de streaming vidéo et fonctionne en comparant les caractéristiques faciales de l'image en question avec des images de visages stockées dans la base de données.

À l'heure actuelle, la vidéosurveillance et la reconnaissance faciale sont répandues partout et sont utilisées dans de nombreux espaces (banques, transports, magasins...) c'est-à-dire que nous les trouvons dans tous les lieux publics et les espaces à risque. Nous trouvons également cette technologie dans les téléphones, elle remplace l'utilisation des mots de passe ou des empreintes digitales. Elle est ainsi un moyen puissant de protéger les données personnelles et d'assurer la protection du téléphone en cas de vol. Elle trouve également son application dans la sécurité publique et les enquêtes médico-légales. Cette technique est même récemment utilisée en Chine pour s'assurer que les étudiants ne sont pas absents de la salle de classe, L'objectif de la reconnaissance faciale est d'accroître la sécurité en protégeant les biens et les personnes grâce au système de streaming vidéo.

C'est dans cette thématique que s'inscrit notre projet de fin d'études. Notre but est de concevoir un système de vidéosurveillance intelligent qui intègre la reconnaissance faciale et qui permet donc d'identifier les personnes non enregistrées et d'identifier les visages. L'objectif principal de notre étude, est de mettre à profit une carte esp32-cam ainsi que les bibliothèques esp32-cam et Opencv pour la mise au point d'un système de vision par ordinateur et de reconnaissance faciale.

Le mémoire est divisé en deux grands chapitres :

- Le chapitre1 : présente les concepts de base de la reconnaissance faciale et donne un aperçu sur les ressources de la carte esp32-cam et les bibliothèques de fonction associées à cette carte.
- Le chapitre2 **:** présente la mise en œuvre des méthodes proposées de détection et de reconnaissance faciale, ainsi que les résultats des tests du système conçu.

Et à la fin, nous terminons notre mémoire avec une conclusion qui résume notre travail et les perspectives futures pour compléter le travail que nous commencé.

Chapitre I
La reconnaissance faciale et ESP32-CAM

I. 1 Introduction

La reconnaissance faciale a été traitée par de nombreux chercheurs et des développements ont émergé chaque année dans ce domaine, La reconnaissance faciale en tant qu'une des techniques biométriques de base, a pris une part de plus en plus importante dans le domaine de la recherche, ceci étant dû aux avances rapides dans les techniques existantes. Parmi les derniers systèmes utilisés dans ce domaine sont ceux basés sur la carte ESP32-CAM pour adapter cette carte pour ce type de système complexe ainsi que ses propriétés distinctives.

Dans ce chapitre, nous essayons d'introduire la technique de reconnaissance faciale et de comprendre ses caractéristiques générales ainsi qu'une connaissance approfondie de la carte ESP32-CAM et de ses outils de programmation.

I. 2 La reconnaissance faciale

La reconnaissance faciale est un moyen d'identifier ou de confirmer l'identité d'un individu grâce à son visage. Les systèmes de reconnaissance faciale peuvent servir à l'identification de personnes sur des photos, dans des vidéos ou en temps réel. La reconnaissance faciale est une catégorie de sécurité biométrique. D'autres logiciels biométriques incluent la reconnaissance de la voix, des empreintes digitales, de la rétine ou de l'iris. Cette technologie est surtout utilisée pour la sécurité et l'application de la loi, bien que d'autres domaines s'y intéressent de plus en plus.

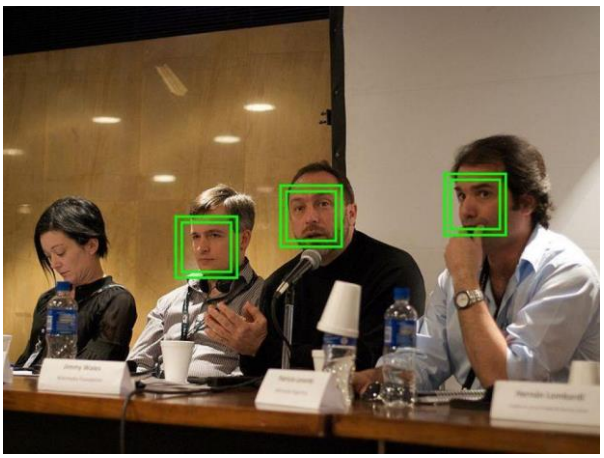


Figure I.1. Détection de visage

I. 2. 1 Généralités sur la reconnaissance faciale

La reconnaissance faciale permet la vérification biométrique en toute situation, ce qui la rend très efficace dans de nombreuses applications liées à la sécurité. La recherche automatique de la reconnaissance faciale remonte au moins aux années 1960, mais les techniques les plus courantes sont uniquement dues au travail basé sur l'apparence à la fin des années 1980 et 1990 **[1].**

Les performances du système de reconnaissance dépendent de plusieurs facteurs qui interfèrent avec plusieurs niveaux et peuvent limiter la précision.

Au cours des vingt dernières années, la reconnaissance automatique des visages est devenue un enjeu primordial, notamment dans les domaines de l'indexation de documents multimédias et

surtout dans la sécurité, ceci est dû aux besoins du monde actuel et aussi à des caractéristiques avantageuses. Plusieurs méthodes de reconnaissance faciale ont été proposées sur deux axes principaux : la reconnaissance d'images statiques et l'identification de séquences d'images (vidéo). Sans aucun doute, les investissements dans la reconnaissance faciale conduisent à une diversité dans les domaines d'application (sécurité, télésurveillance, contrôle d'accès ...). **[2]**.

I. 2. 2 Etapes de la reconnaissance faciale

La reconnaissance faciale est un système permettant d'identifier et de confirmer les personnes en contrôlant si celles-ci appartiennent à la base de données du système. L'image suit un processus de reconnaissance faciale spécifique contenant plusieurs étapes, Les systèmes de reconnaissance faciale peuvent varier, mais ils fonctionnent généralement de la manière suivante :

Figure I.2. Les étapes de reconnaissance Faciale

La détection du visage : le système détecte la présence d'un visage humain dans une image ou un vidéo, qui peut s'effectuer en temps réel ou à partir de banque de données déjà enregistrées. **[3]**

L'analyse du visage : le système extrait les traits et peut aussi détecter les caractéristiques génériques (ex. âge, genre, couleur de peau), tout comme les caractéristiques émotionnelles du visage. **[3]**

L'association du visage : le système compare les données reçues avec celles de banques de données comportant d'autres visages et traits, permettant de confirmer ou d'infirmer l'identité de l'individu ou encore d'interpréter ses émotions. **[3]**

Confirmation de l'identité : Le système rend un résultat sur la correspondance. **[3]**

I. 2. 3 Domaines d'applications de la reconnaissance faciale

Les systèmes de reconnaissance faciale sont plus applicables à la vie quotidienne. Des exemples d'applications de reconnaissance faciale comprennent :

- Les réseaux sociaux sont utilisés sur Internet pour identifier une personne à travers l'image de son visage et pour vérifier l'identité des achats en ligne, sur le smart phone

pour déverrouiller, ou par des services de sécurité pour reconnaître les individus recherchés.

- Réduire la fraude par la vérification d'identité dans les banques, les élections, les soins de santé, les paiements de prestations, etc.
- La technologie de reconnaissance faciale peut être utilisée pour identifier les personnes qui ont un casier judiciaire et localiser les criminels et les trouver. **[2]**

I. 3 Les microcontrôleurs ESP32

I. 3. 1 Description de ESP 32

ESP32 est une série de microcontrôleurs à faible coût et à faible consommation d'énergie sur puce avec Wi-Fi intégré et Bluetooth bi-mode. La série ESP32 utilise un microprocesseur Tensilica Xtensa LX6 dans des variantes à double cœur et à cœur unique et comprend des commutateurs d'antenne intégrés, un balun RF, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation. ESP32 est créé et développé par Espressif Systems, et est fabriqué par TSMC (Taiwan Semiconductor Manufacturing Company) en utilisant leur processus de 40 nm. **[4]**

L'ESP 32 a été annoncé pour la première fois en septembre 2016 par Espressif Systems, et il coûte environ 5 $.

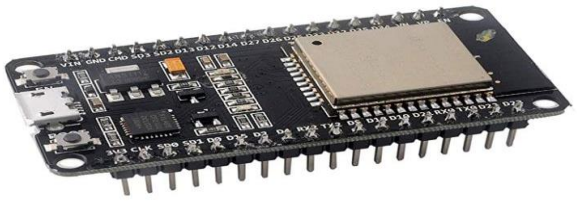


Figure I.3. Un carte ESP 32

I. 3. 2 Les caractéristiques techniques

- **Processeurs :**
  - CPU : microprocesseur Xtensa dual-core (ou single-core) 32 bits LX6, fonctionnant à 80 ou 240 MHz et fonctionnant jusqu'à 600 DMIPS.
  - Coprocesseur ultra basse consommation (ULP).
- **Mémoire :** 520 KB SRAM
- **Connectivité sans fil :**
  - Wi-Fi : 802.11 b / g / n, Le module Wi-Fi fonctionne dans une plage de 2,4 GHz à 2,5 GHz.
  - Bluetooth : v4.2 BR / EDR et BLE (partage la radio avec Wi-Fi).
- Trois modes de fonctionnement : 1. Point d'accès. 2. Client. 3. Point d'accès + station
- Microprocesseur à double cœur de 32 bits
- La tension de fonctionnement est de 3,3 V
- Fréquence d'horloge de 80 MHz et va jusqu'à 240 MHz
- La mémoire ROM est de 448 Ko
- La mémoire flash externe est prise en charge et peut aller jusqu'à 32 Mo, c.-à-d. 4 Mo
- L'intensité maximale dans chaque broche est de 12 mA, mais il est recommandé d'utiliser 6 mA
- Il dispose de 36 broches d'entrée/sortie à usage général
- Les broches d'entrée et de sortie à usage général sont dotées des fonctionnalités PWM/I2C et SPI
- Tension de fonctionnement de 2 à 3,6 V
- Flux de sommeil profond de 2,5 µA
- Support tactile capacitif à 10 électrodes
- La température de fonctionnement est comprise entre -40 °C et +125 °C

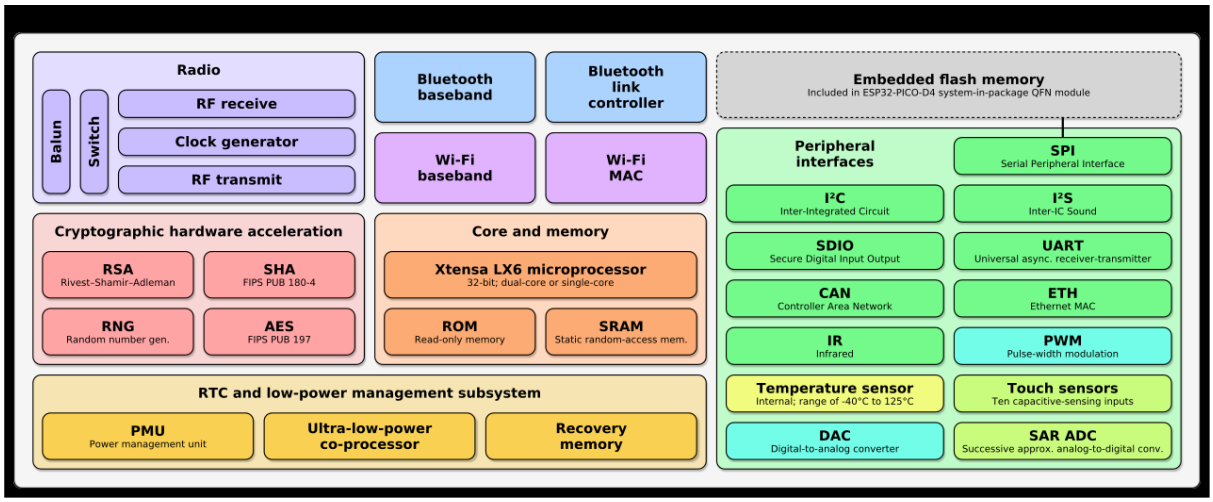


Figure I.4. Schéma fonctionnel ESP32

I. 3. 3 Brochage de la carte de développement ESP32 :

La carte de développement ESP32 possède au total 30 broches qui la connectent au monde extérieur. Les connexions sont les suivantes :

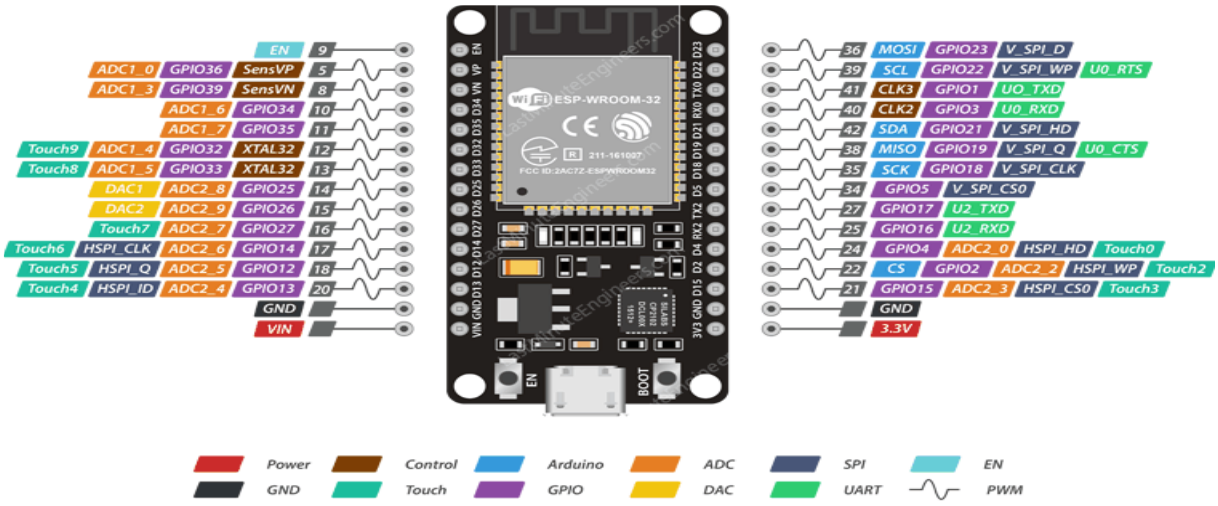


Figure I.5. Brochage de la carte de développement ESP32

- Broches d'alimentation : deux broches d'alimentation sont disponibles à savoir : Broche VIN et broche 3,3 V. La broche VIN peut être utilisée pour alimenter directement l'ESP32 et ses périphériques, qui doit être connectée à une source de tension 5V régulée. La broche 3,3 V est la sortie d'un régulateur de tension intégré. Cette broche peut être utilisée pour alimenter des composants externes.

- GND : est une broche de masse de la carte de développement ESP32.
- Broches de connexions séries : broches matérielles I2C et SPI d'ESP32 pour brancher toutes sortes de capteurs et de périphériques dans votre projet.
- Broches GPIO : La carte de développement ESP32 possède 25 broches GPIO qui peuvent être affectées à diverses fonctions par programmation. Chaque GPIO activé numérique peut être configuré en pull-up ou pull-down interne, ou réglé en haute impédance. Lorsqu'il est configuré comme entrée, il peut également être défini sur front-trigger ou level-trigger pour générer des interruptions CPU.
- Pins UART : La carte de développement ESP32 possède 2 interfaces UART, UART0 et UART2, qui permettent une communication asynchrone (RS232 et RS485) et un support IrDA, et communiquent jusqu'à 5 Mbps. UART fournit également la gestion matérielle des signaux CTS et RTS et le contrôle de flux logiciel (XON et XOFF).
- Capteur à effet Hall : L'ESP32 intègre un capteur Hall basé sur une résistance à porteuse N. Lorsque la puce est dans le champ magnétique, le capteur Hall développe une petite tension latéralement sur la résistance, qui peut être directement mesurée par l'ADC.
- Capteurs tactiles : L'ESP32 dispose de 10 GPIO à détection capacitive, qui détectent les variations induites par le contact ou l'approche des GPIO avec un doigt ou d'autres objets. La nature à faible bruit de la conception et la haute sensibilité du circuit permet d'utiliser des plots relativement petits. Des tableaux de tampons peuvent également être utilisés, de sorte qu'une plus grande zone ou plusieurs points peuvent être détectés **[5].**

I. 3. 4 Carte de développement :

L'ESP32 est une version mise à jour de l'ESP8266, qui était une puce qui a pris les expérimentateurs du monde occidental par "surprise" en 2014. L'ESP8266 d'origine a été introduit sur un module appelé ESP-01, qui avait très peu de documentation en anglais donc ses capacités étaient inconnues à l'époque. Une fois la documentation traduite en anglais, de nombreux expérimentateurs ont rapidement pris conscience de la puissance de l'ESP8266, ce qui l'a rendu rapidement très populaire.

L'ESP32 a amélioré la conception de l'ESP8266 de plusieurs façons. Il offre à la fois Bluetooth et BLE (Bluetooth Low Energy), alors que l'ESP8266 n'a que le WiFi (ce que, bien sûr, l'ESP32 a aussi). Il est plus rapide et est disponible dans une version avec un processeur à double cœur.

Il est également capable de fonctionner en mode ultra-basse consommation, idéal pour les applications alimentées par batterie.

Figure I.6. Un carte ESP 8266

I. 3. 5 Cartes de développement ESP32 populaires

ESP32 CAM

La carte de développement ESP32-CAM est un peu différente des autres cartes de développement de cette liste. Elle dispose d'une caméra intégrée et d'un connecteur pour carte micro SD.

L'ESP32-CAM est basé sur le module ESP32-S, il partage donc les mêmes spécifications. Cela comprend les interfaces UART, SPI, I2C et PWM, le téléchargement d'images Wi-Fi, des vitesses d'horloge allant jusqu'à 160 MHz et 9 ports GPIO.

Il comprend un module OV2640 – doté d'un capteur de 2 mégapixels – et prend également en charge les caméras OV7670.

ESP32-DevKitC

L'ESP-DevKitC est une carte de développement relativement petite d'Espressif. Les broches d'E/S sont détachées sur les en-têtes des deux côtés pour faciliter l'interfaçage.

Les développeurs peuvent connecter des périphériques avec des fils de cavalier ou monter l'ESP32-DevKitC V4 sur un breadboard.

Un avantage clé de cette carte de développement n'est pas seulement sa petite taille, mais aussi sa faible consommation d'énergie.

Node-MCU-32S

La carte de développement NodeMCU dispose d'une connectivité Wi-Fi+Bluetooth, d'un CP2102 embarqué et de clés.

Une caractéristique clé de cette carte de développement proposée par la firme AI Thinker est que les broches E/S du module ESP-WROOM-32 sont accessibles via les en-têtes d'extension. De plus, il est open-source et supporte plusieurs codes sources différents.

I. 4 Description de la carte ESP32-CAM :

ESP32-CAM est une carte de développement à faible coût basée sur l'ESP32 avec une caméra compacte. C'est une solution idéale pour les projets d'application, de prototypage de systèmes de domotique ou de IoT (Internet of Things).

La carte intègre le Wi-Fi, le Bluetooth traditionnel et le BLE (Bluetooth Low Energy), avec deux processeurs LX6 32 bits hautes performances. Il adopte une architecture pipeline à 7 phases, un capteur sur puce, un capteur Hall, un capteur de température, etc., et les plages de modulation de fréquence principales de 80 MHz à 240 MHz, Entièrement compatible avec les normes Wi-Fi 802.11b/g/n/e/i et Bluetooth 4.2, il peut être utilisé en mode maître pour construire un contrôleur de réseau autonome, ou comme esclave relié à d'autres MCU hôtes pour augmenter les performances.

Il convient aux appareils électroménagers intelligents, au contrôle sans fil industriel, à la surveillance sans fil, à l'identification QR sans fil, à la reconnaissance faciale, aux signaux GPS sans fil et à d'autres applications IoT. C'est une solution idéale pour les applications IoT.

Figure I.7. Un carte ESP 32-CAM

I. 4. 1 Caractéristiques techniques de l'ESP32-CAM

Le module ESP32-CAM dispose quelques caractéristiques techniques très intéressantes disponibles sur la fiche technique du fabricant. Les caractéristiques les plus importantes sont :

Connectivité : Wi-Fi 802.11b / g / n + Bluetooth 4.2 avec BLE. Prend en charge le téléchargement d'images via Wi-Fi.

Ports d'E/S : UART, SPI, I2Cet PWM. Il a 9 broches GPIO.

Fréquence d'horloge : jusqu'à 160Mhz.

Puissance de calcul du microcontrôleur : jusqu'à 600 DMIPS.

Mémoire : 520 Ko de SRAM + 4 Mo de PSRAM + fente pour carte SD

Appareil photo : Prend en charge les caméras OV2640, ces types de caméras ont :

- Un capteur de 2 MP
- Résolution de l'image UXGA 1622 × 1200 px
- Format de sortie YUV422, YUV420, RGB565, RGB555 avec compression de données 8 bits.
- Transfert de l'une image entre 15 et 60 FPS.

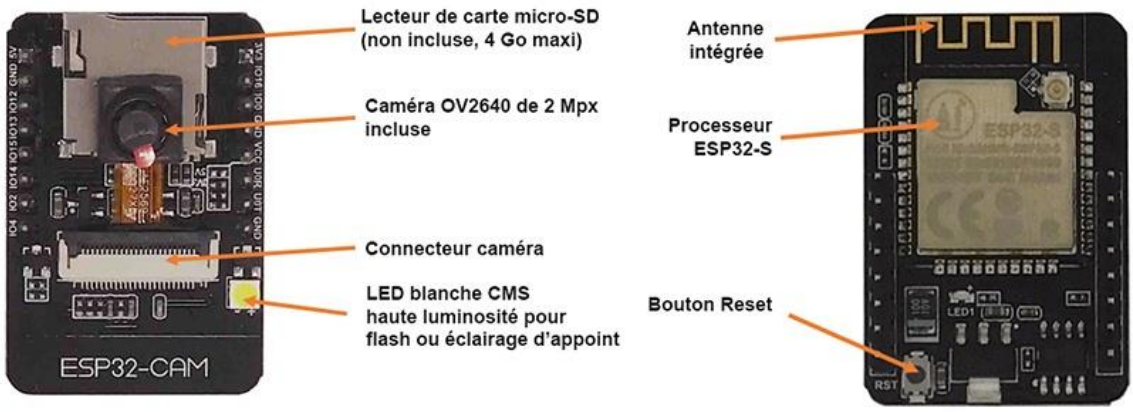


Figure I.8. ESP32-CAM composantes

I. 4. 2 Brochage de ESP32-CAM

La carte de développement ESP32-CAM possède au total 16 broches qui la connectent au monde extérieur. Les connexions sont les suivantes :

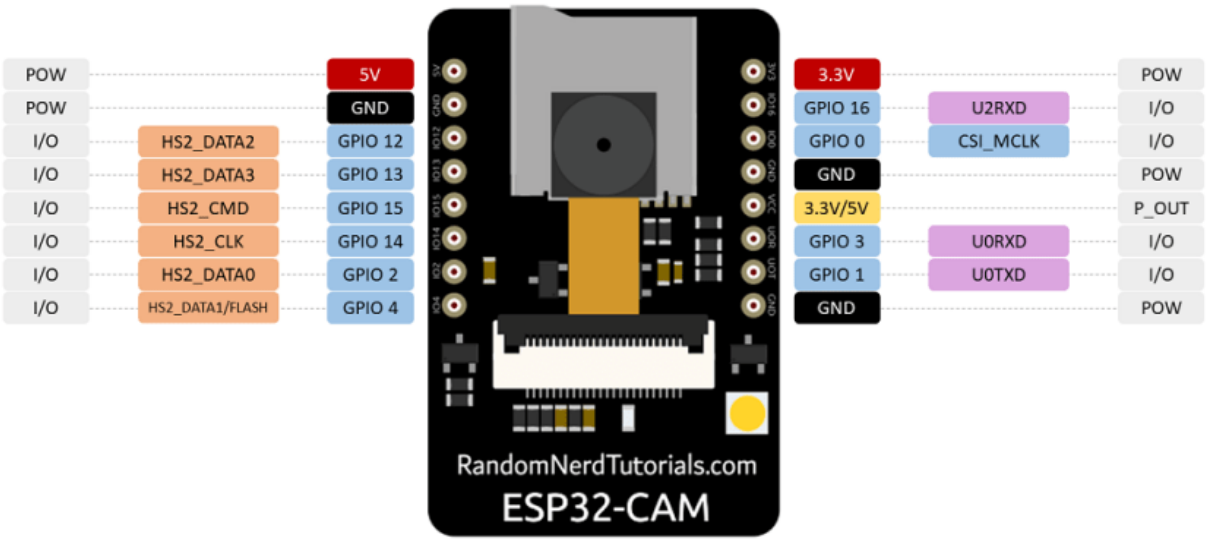


Figure I.9. Pin out d'ESP32-CAM

Broches d'alimentation : L'ESP32-CAM est livrée avec trois GND broches (colorées en noir) et deux broches d'alimentation (colorées en rouge) : 3.3V et 5V.

I. 4. 3 Applications de la carte ESP32-CAM :

La gamme d'applications des modules ESP32-CAM est très large. En particulier, ils sont bien adaptés aux appareils domotiques, à l'électronique portable et à l'Internet des objets (IoT). Un processeur efficace et de nombreux périphériques permettent de réaliser des projets très complexes et avancés. Pour les utilisateurs débutants, cela peut poser problème, car la multitude de fonctions, d'options disponibles et même la documentation de base (ESP32 Technical Reference Manual) comprenant presque 700 pages peuvent déborder.

Heureusement, l'utilisation d'ESP32-CAM est assez simple grâce à de nombreuses solutions toutes faites. Tous les logiciels nécessaires sont gratuits et, ce qui est le plus important, il est possible de trouver sur Internet de nombreuses bibliothèques et beaucoup de projets qui peuvent constituer une inspiration ou une base pour vos propres études. **[6]**

A titre d'exemples, des applications de la carte ESP32-CAM peuvent être proposées dans les domaines de robotique, réseaux locaux, automatisation des usines, systèmes de reconnaissance faciale, domotique (smart home applications) ...

I. 4. 4 Outils de développements :

Il existe une variété de plates-formes de développement qui peuvent être équipées pour programmer l'ESP32-CAM :

- **Arduino IDE** avec le module ESP32 Arduino Core.
- Espruino.
- FAUST, langage de programmation de traitement de données audio, utilisant son DSP.
- Lua RTOS pour ESP32.
- MicroPython, une variante pour les systèmes embarqués du langage Python.
- Mruby une variante pour les systèmes embarqués du langage Ruby ;

Dans notre travail nous avons optés pour l'environnement de développement intégré Arduino (Arduino IDE) qui est une application multiplateforme (pour Windows, macOS, Linux) écrite dans des fonctions de C et C ++. Il est utilisé pour écrire et télécharger des programmes sur des

cartes compatibles Arduino et autres. Des modules supplémentaires doivent être ajoutés à l'IDE Arduino, afin de programmer l'ESP32-CAM. **[4]**

I. 5 Etude du système de développement Arduino IDE

I. 5. 1 Description de l'IDE

L'IDE est un logiciel de programmation qui permet d'écrire, de modifier un programme et de le convertir en une série d'instructions compréhensibles pour la carte. Il programme par code, contenant une cinquantaine de commandes différentes. Le langage de l'IDE Arduino est un mélange entre le C et le C++, il possède un jeu d'instruction très riche. A l'ouverture, L'interfdu logiciel Arduino se présente de la façon suivante : **[7]**

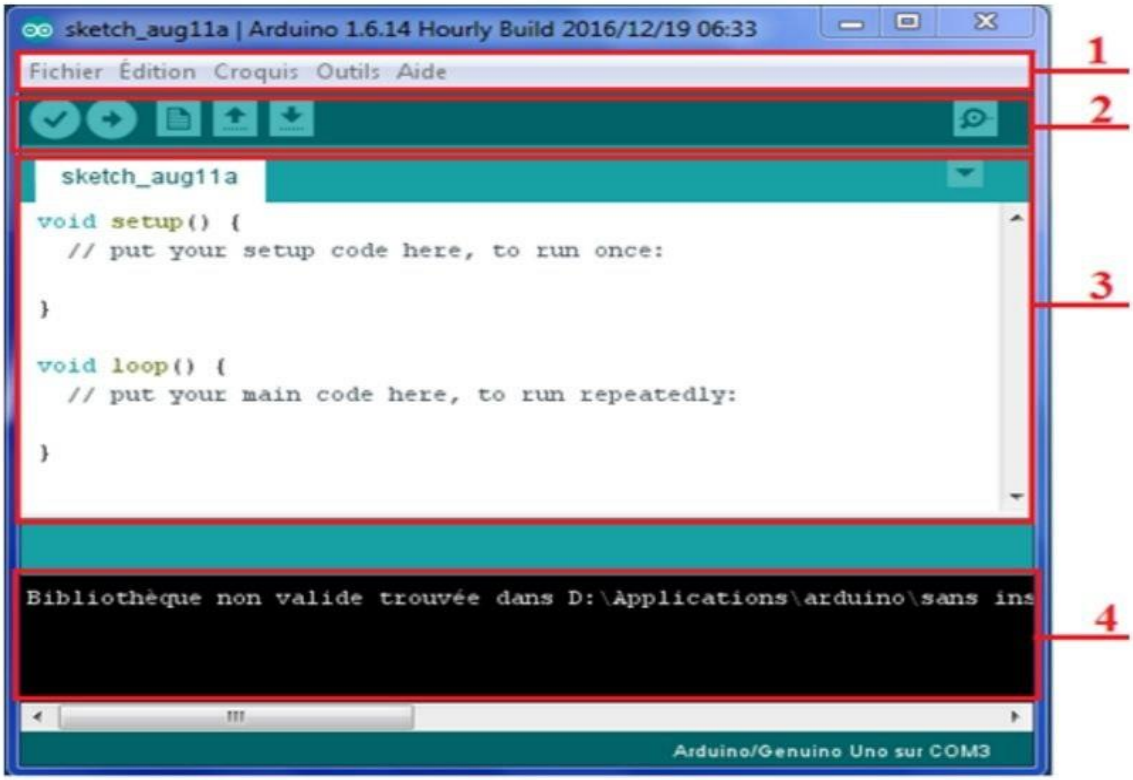


Figure I.10. Interface de logiciel IDE Arduino

➢ **Barre1 :** Options de configuration du programme tel que :

- Créer un nouveau programme ou ouvrir un programme existant.
- Enregistre/demande où enregistrer le document en cours.
- Toute une liste se déroule pour afficher les noms d'exemples de programmes Existant...etc.

➢ **Barre2 :** boutons pout la programmation des cartes :

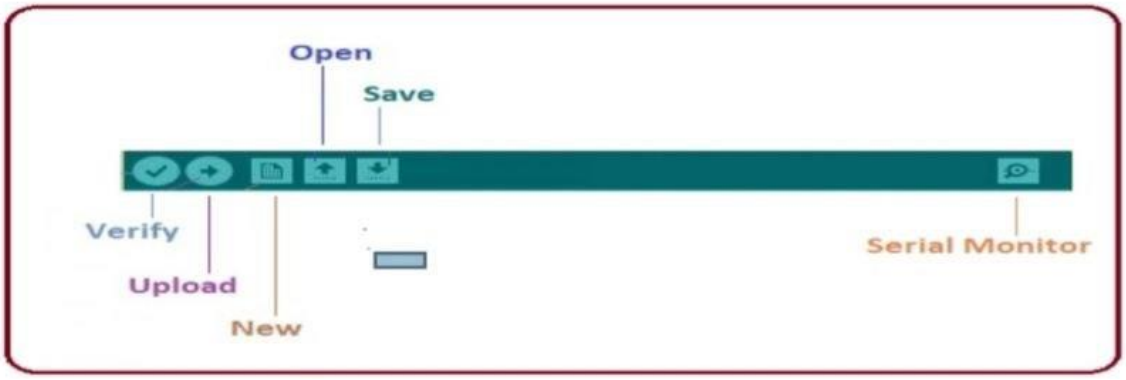


Figure I.11. Présentation des boutons

- Vérifier (Verify) : vérifier les erreurs dans le code.
- Charge (Upload) : compiler le code et charge le programme sur la carte Arduino.
- Nouveau (New) : créer un nouveau sketch.
- Ouvrir (Open) : ouvrir un des sketchs déjà présents.
- Sauvegarder (Save) : sauvegarder le sketch.
- Serial Monitor : permet d'accéder au port série (en RX/TX).

➢ **Barre3 :** programme à créer.

➢ **Barre4 :** débogueur (affichage des erreurs de programmation). **[7]**

I. 5. 2 Structure d'un programme Arduino

Un programme Arduino est constitué de trois parties :

- La partie où l'on déclare les variables, affecte les broches, appelle les bibliothèques.
- L'initialisation (Setup) où l'on configure les broches en entrée ou en sortie, initialise les différents objets (comme le moniteur série, les composants ayant leur propre bibliothèque. Cette fonction est exécutée une seule fois après la mise sous tension de la carte ou après un reset.
- Le programme principal (Loop). Cette fonction est répétée indéfiniment.

- **Coloration syntaxique :** Lorsqu'un code est écrit dans l'interface de programmation,

Certains mots apparaissent en différentes couleurs qui clarifient le statut des différents éléments :

- **En orange**, apparaissent les mots-clés reconnus par le langage Arduino comme des

Fonctions existantes. Lorsqu'on sélectionne un mot coloré en orange et qu'on effectue un

Clic avec le bouton droit de la souris, on a la possibilité de choisir « Find in reference » : cette commande ouvre directement la documentation de la fonction sélectionnée.

- **En bleu**, apparaissent les mots-clés reconnus par le langage Arduino comme des

Constantes.

- **En gris**, apparaissent les commentaires qui ne seront pas exécutés dans le programme. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes. **[7]**

I. 5. 3 Etude des fonctions de la librairie ESP32

I. 5. 3. 1 Fonctions de la bibliothèque Wi-Fi ESP32 (IDE Arduino)

Les bibliothèques Wi-Fi prennent en charge la configuration et la surveillance des fonctionnalités Wi-Fi ESP32. Cela inclut une configuration pour :

- ✓ Mode station (également appelé mode STA ou mode client Wi-Fi). ESP32 se connecte à un point d'accès.
- ✓ Mode AP (également appelé mode Soft-AP ou mode point d'accès). Les stations se connectent à l'ESP32.
- ✓ Mode de cohabitation Station/AP (ESP32 est à la fois un point d'accès et une station connectée à un autre point d'accès).
- ✓ Différents modes de sécurité pour ce qui précède (WPA, WPA2, WEP, etc.) **[8]**

- ➢ **La bibliothèque Wi-Fi**

La première chose à faire pour utiliser les fonctions Wi-Fi ESP32 est d'inclure la bibliothèque WiFi.h dans le code spécial, comme suit :

```
#include <WiFi.h>
```

- **Point d'accès**

Lorsque vous définissez votre carte ESP32 comme point d'accès, vous pouvez être connecté à l'aide de n'importe quel appareil doté de capacités Wi-Fi sans vous connecter à votre routeur. Lorsque vous définissez l'ESP32 comme point d'accès, vous créez son propre réseau Wi-Fi et les périphériques Wi-Fi (stations) à proximité peuvent s'y connecter, comme votre smartphone ou votre ordinateur. Ainsi, vous n'avez pas besoin d'être connecté à un routeur pour le contrôler.

Cela peut également être utile si vous souhaitez que plusieurs périphériques ESP32 communiquent entre eux sans avoir besoin d'un routeur**. [8]**

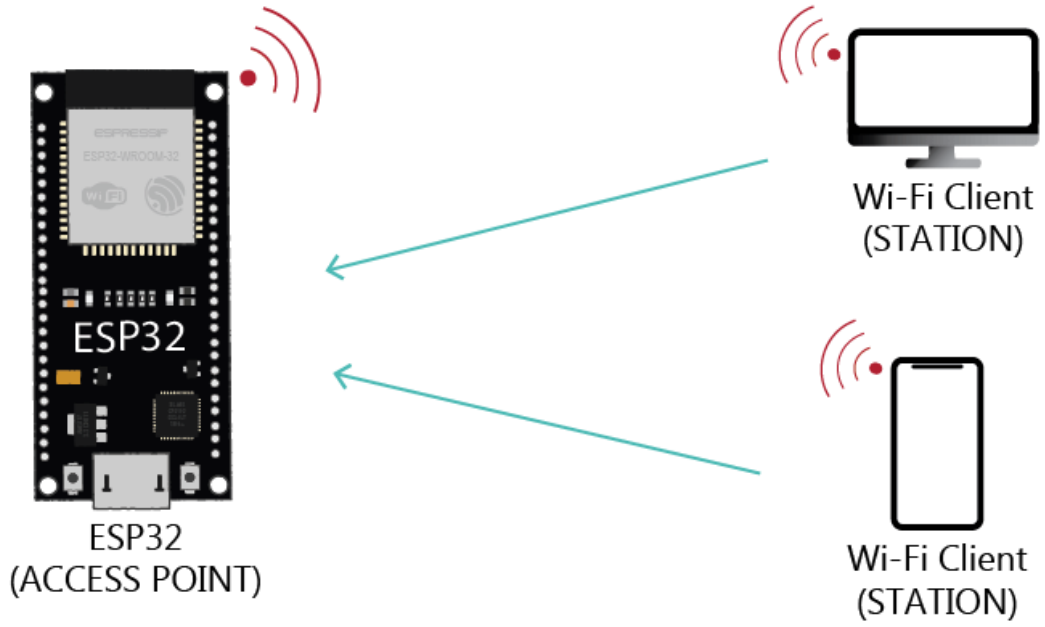


Figure I.12. Connectez-vous au WI-FI

- **Définir ESP32 comme point d'accès**

Pour définir ESP32 comme point d'accès, réglez le mode Wi-Fi sur le point d'accès :

WiFi.mode(WIFI_AP)

Et puis, utilisez la méthode softAP comme suit

WiFi.softAP(ssid, password);

Ssid est le nom donné au point d'accès esp32 et la variable **password** est le mot de passe du point d'accès. Si nous ne définissons pas de mot de passe, nous le définissons sur NULL.

Il existe également d'autres paramètres facultatifs que vous pouvez transmettre à softAP. Voici tous les paramètres :

WiFi.softAP(const char* ssid, const char* password, int channel, int ssid_hidden, int max_connection)

- **ssid :** nom du point d'accès – maximum de 63 caractères;
- **Password :** minimum de 8 caractères ; set to NULL si vous souhaitez que le point d'accès soit ouvert ;
- **Channel :** Wi-Fi numéro de canal (1-13)
- **ssid_hidden:** (0 = broadcast SSID, 1 = hide SSID)
- **Max_connection** : nombre maximal de clients connectés simultanément (1-4) **[8]**

I. 5. 3. 2 Fonctions de la bibliothèque Bluetooth ESP32 (IDE Arduino)

- **Esp32 BLE Serveur et client (Bluetooth Low Energy)**

Découvrez comment établir une connexion BLE (Bluetooth Low Energy) entre deux cartes ESP32. Un ESP32 sera le serveur, et l'autre ESP32 sera le client.

Le serveur BLE annonce des caractéristiques qui contiennent des lectures de capteur que le client peut lire. Le client ESP32 BLE lit les valeurs de ces caractéristiques (température et humidité) et les affiche sur un écran OLED.

- **Qu'est-ce que la technologie Bluetooth basse consommation**

Avant de passer directement au projet, il est important de jeter un coup d'œil rapide à certains des concepts de base de ble afin de mieux comprendre le projet plus tard. Si vous êtes déjà familiarisé avec BLE, vous pouvez passer à la section Vue d'ensemble du projet.

Le Bluetooth basse consommation, BLE en bref, est une alternative économe en énergie au Bluetooth. L'application de base de BLE est la transmission à courte distance de petites quantités de données (faible bande passante). Contrairement au Bluetooth, qui fonctionne toujours, BLE reste immobile sauf lorsque vous commencez à vous connecter. **[9]**

- **Serveur et client BLE**

Avec Bluetooth Low Energy, il existe deux types d'appareils : le serveur et le client. L'ESP32 peut agir en tant que client ou serveur.

Le serveur annonce son existence, de sorte qu'il peut être trouvé par d'autres périphériques et contient des données que le client peut lire. Le client analyse les périphériques à proximité et, lorsqu'il trouve le serveur qu'il recherche, il établit une connexion et écoute les données entrantes. C'est ce qu'on appelle la communication point à point.

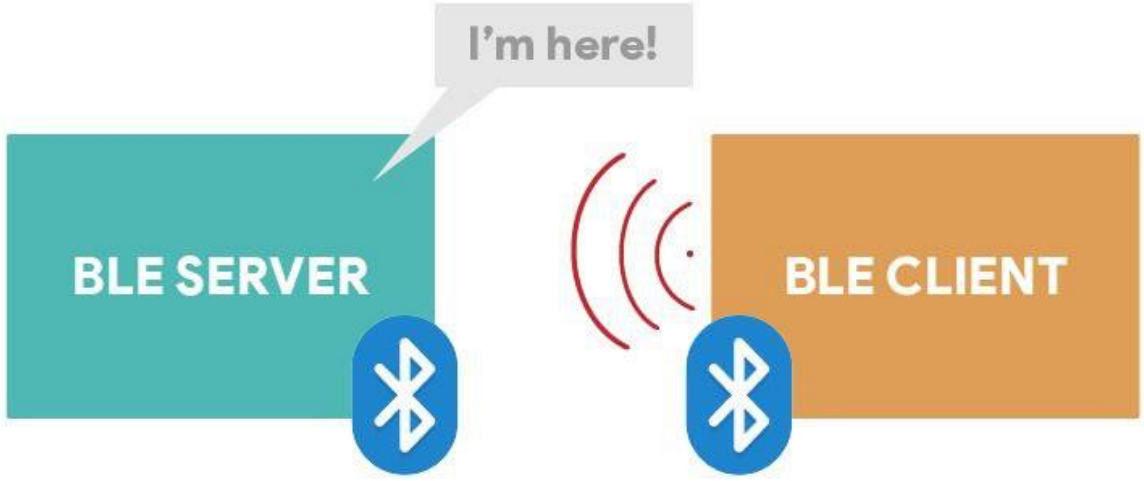


Figure I.13. Communication serveur avec un client BLE

Comme mentionné précédemment, BLE prend également en charge le mode de diffusion et le réseau maillé :

Mode de diffusion : le serveur transmet des données à de nombreux clients connectés ;

Réseau maillé : tous les appareils sont connectés, il s'agit d'une connexion plusieurs à plusieurs.

Même si les configurations de réseau de diffusion et de maillage sont possibles à mettre en œuvre, elles ont été développées très récemment, il n'y a donc pas beaucoup d'exemples implémentés pour l'ESP32 en ce moment.

- **GATT**

GATT signifie Generic Attributes et définit une structure de données hiérarchique exposée aux périphériques BLE connectés. Cela signifie que le GATT définit la façon dont deux périphériques BLE envoient et reçoivent des messages standard. Il est important de comprendre cette hiérarchie, car il sera plus facile de comprendre comment utiliser le BLE et écrire vos applications.

- **Fonctionnement du code**

Jetons un coup d'œil rapide au fonctionnement de l'exemple de code du serveur BLE. Il commence par importer les bibliothèques nécessaires pour les fonctionnalités BLE. **[9]**

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
```

I. 5. 3. 3 Fonctions de la bibliothèque Les entrées/sorties ESP32 (IDE Arduino)

Le système de développement Arduino vient avec un nombre important de fonctions de base permettant d'interagir avec son environnement. Les fonctions les plus utilisée sont les fonctions d'entrée/sorties. Ce sont elles qui permettent d'envoyer ou de mesurer une tension sur une des broches de la carte. Dans un premier temps, avant d'effectuer une mesure ou d'envoyer une commande. Il est nécessaire de définir la direction des broches utilisées. Pour cela on fait appel à la fonction pin Mode en lui donnant d'une part, la broche concernée, et d'autre part, la direction : **[10]**

```
Void setup () {
pinMode (1, OUTPUT) ; // Broche 1 en sortie
PinMode (2, INPUT) ; // Broche 2 en entrée
 }
```

Une fois cette configuration faite, on peut procéder à l'utilisation des broches. Toutes les broches sont capables d'écrire et de lire des données numériques (c'est-à-dire des 0 (0V) ou

des 1 (5V)). Mais, certaines disposent de fonctionnalités supplémentaires. Tout d'abord, toutes les cartes Arduino possèdent des entrées analogiques. Ce sont les broches A0-A1-A2 etc. Elles permettent de lire des tensions analogiques (Comprise entre0 et 5V) et de le convertir en entier (compris entre 0 et 1023) proportionnellement à la tension mesurée. Certaines cartes Arduino possèdent des sorties analogiques faisant l'opération inverse (met une tension sur la broche proportionnellement à l'entier donné), mais ce n'est pas le cas pour l'Arduino UNO.

Pour pouvoir tout de même contrôler des composants autrement qu'en « tout où rien » il est possible d'utiliser des broches PWM. Ce sont les broches annotées par un tilde ~ sur la carte. La PWM (Pulse Width Modulation) ou MLI (Modulation à Largeur d'Impulsion) est utilisée pour synthétiser des signaux analogiques en modulant le temps passé à l'état 1 (5V). Le signal obtenu est représenté (figure II.4). En utilisant une fréquence relativement élevée, la PWM permet de commander certains composants comme s'il recevait une tension analogique. Cela provient du fait que les composants utilisés dans l'électronique analogique, ne changent pas d'états instantanément. Par exemple, une ampoule à incandescence éclaire un court instant après avoir été éteinte. Ce phénomène est généralement invisible à l'œil nu. Grâce à la PWM, on pourra par exemple faire varier l'intensité d'une LED.

- **DigitalRead(pin) :** mesure une donnée numérique sur une des broches, la broche en question doit être réglée en entrée.
- **DigitalWrite (pin, value) :** écrit une donnée numérique sur une des broches, la broche concernée doit être réglée en sortie. Le paramètre valu doit être égal à HIGH (état 1 soit 5V) ou LOW (état 0 soit 0V).
- **AnalogRead(pin) :** mesure une donnée analogique sur une des broches (compatible seulement), la broche doit être réglée sur entrée.
- **analogWrite (pin, value):** écrit une donnée sous forme de PWM sur une des broches (compatible uniquement), la broche doit être réglée en sortie. Le paramètre valu doit être compris dans l'intervalle [0 ;255] [**10]**

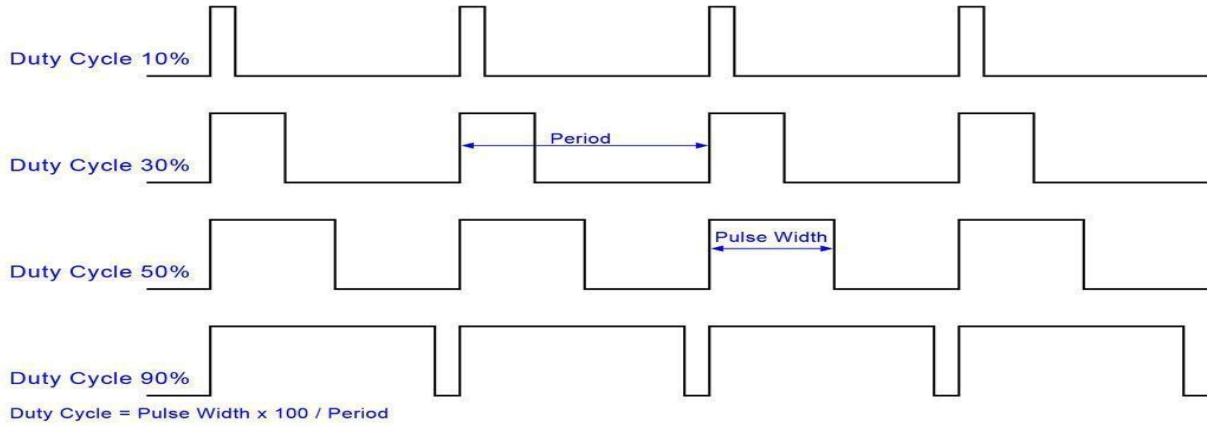


Figure I.14. Signal PWM

I. 6 Présentation Python et OpenCV

I. 6. 1 Python

Python est un langage de script de haut niveau, structuré et open source, multi paradigme et multi - usage. Il a été développé à l'origine par Guido van Rossum en 1989.

Le langage Python est un langage de programmation interprété, ce qui veut dire que les instructions qu'on lui envoie sont « transcrites » en langage machine au fur et à mesure de leur lecture. D'autres langages (comme le C / C ++) sont appelés « langages compilés » car, avant de pouvoir les exécute, un logiciel spécialisé se charge de transformer le code du programme en langage machine. On appelle cette étape la « compilation ». À chaque modification du code, il faut rappeler une étape de compilation. Les avantages d'un langage interprété sont la simplicité (on ne passe pas par une étape de compilation avant d'exécuter son programme) et la portabilité**.** **[11]**

Python est placé sous une licence libre proche de la licence BSD7 et fonctionne sur la plupart des plates - formes informatiques, des smartphones aux ordinateurs centraux de Windows à Unix avec notamment GNU / Linux en passant par macOS, encore Android, iOS, et peut aussi être traduit en Java ou .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

I. 6. 1. 1 Python IDEL

Python IDLE Autrement dit environnement de développement et d'apprentissage intégré est un environnement de développement intégré (IDE) pour Python. Le programme d'installation Python pour Windows contient le module IDLE par défaut. IDLE n'est pas disponible par défaut dans les distributions Python pour Linux. Il doit être installé. IDLE fournit un éditeur de texte complet pour créer des scripts Python incluant des fonctionnalités telles que la coloration syntaxique, la complétion automatique et l'indentation intelligente. Il possède également un débogueur avec des fonctionnalités à pas et de points d'arrêt. **[11]**

I. 6. 2 Opencv

OpenCV (Open - source Computer Vision library) est la librairie de référence pour le traitement d'image et d'apprentissage automatique. OpenCV a été conçu pour fournir une infrastructure commune aux applications de vision par ordinateur et pour accélérer l'utilisation de la perception de la machine dans les produits commerciaux. OpenCV étant un produit sous licence BSD, il est facile pour les entreprises d'utiliser et de modifier le code.

La bibliothèque contient plus de 2500 algorithmes optimisés, qui inclut un ensemble complet d'algorithmes classiques et avancés de vision par ordinateur et d'apprentissage automatique. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, reconnaître des chiffres et des caractères, identifier des objets, classer les actions humaines dans des vidéos, suivre des objets en mouvement. OpenCV compte plus de 47 000 utilisateurs et le nombre de téléchargements atteint plus de 18 millions. La bibliothèque est largement utilisée dans les entreprises, les groupes de recherche et les organismes gouvernementaux. Aux côtés de sociétés bien définies telles que Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda et Toyota qui utilisent la bibliothèque, ainsi que de nombreuses startups telles qu'Applied Minds, VideoSurf et Zeitera Corporation. OpenCV associe des images de vues de rue, détecte des intrusions dans une vidéosurveillance, surveille des équipements, aide les robots à naviguer et à ramasser des objets, détecte les accidents de noyade dans les piscines.

La bibliothèque possède des interfaces C++, Python, Java et MATLAB et prend en charge Windows, Linux, Android et Mac OS. OpenCV s'appuie principalement sur les applications de vision en temps réel. Des interfaces CUDA et OpenCL complètes sont développées en ce moment. Il existe plus de 500 algorithmes et environ 10 fois plus de fonctions qui composent ou supportent ces algorithmes. OpenCV est écrit natif en C++ et possède une interface basée sur un modèle qui fonctionne de manière transparente avec les conteneurs STL. **[11]**

OpenCV a une structure modulaire, ce qui signifie que le package comprend plusieurs bibliothèques partagées ou statiques. Les modules principaux accessibles au travers de cette API sont :

- **Core :** un module compact définissant les structures de données de base, y compris la matrice multidimensionnelle dense mat et les fonctions de base utilisées par tous les autres modules.
- **Imgproc :** un module de traitement d'image comprenant un filtrage d'image linéaire et non linéaire, des transformations d'image géométriques conversion (redimensionnement, affinage et déformation en perspective, remappage générique basé sur, une space coule d) des histogrammes, etc.
- **Vidéo :** un module d'analyse vidéo qui inclut des algorithmes d'estimation de mouvement, de soustraction d'arrière-plan et de suivi d'objets.
- **Calib3d :** algorithmes géométriques de base à vues multiples, calibrage de caméras simples et stéréoscopiques, estimation de la pose d'objets, algorithmes de correspondance stéréoscopique et éléments de reconstruction 3D.
- **Features2d :** détecteurs, descripteurs et appariеurs de descripteurs d'éléments saillants.
- **Object :** détection d'objets et d'instances des classes prédéfinies (par exemple, les visages, les yeux, les tasses, les personnes, les voitures, etc.).
- **Highgui :** une interface facile à utiliser pour des fonctionnalités d'interface utilisateur simples.
- **Videoio :** une interface facile à utiliser pour la capture vidéo et les codecs vidéo.
- **GPU :** algorithmes accélérés par le GPU de différents modules OpenCV.

Certains autres modules auxiliaires, tels que les wrappers de test FLANN et Google, les liaisons Python, etc.

I. 7 Conclusion

Dans ce chapitre, les concepts nécessaires à la compréhension de la reconnaissance faciale ont été présentés. Par la suite une étude générale est fournie sur la carte ESP32. Après avoir défini ESP32 et déterminé ses propriétés les plus importantes, une étude détaillée de la carte ESP32-CAM est présentée, ainsi que les outils de programmation et les langages utilisés dans sa programmation et, en plus des caractéristiques les plus importantes qui en ont fait un composant requis du système de reconnaissance faciale.

Chapitre II
Partie réalisation

II. 1 Introduction

Dans le précédent chapitre nous avons vu la partie théorique et tout ce qui est essentiel concernant les systèmes de reconnaissance faciale.

Ce dernier chapitre est consacré à la conception et à la programmation du système qui permettra d'identifier les personnes à partir du streaming vidéo pour la reconnaissance faciale.

Pour implémenter ce système, nous utilisons deux méthodes différentes, la première méthode dans laquelle nous utilisons le logiciel Arduino Ide pour programmer esp32-cam, la deuxième méthode en utilisant le logiciel Visual Studio Code et la bibliothèque OpenCV par webcam, à travers une implémentation par le langage de programmation Python.

II. 2 Description du projet

Le but de notre projet est d'identifier et de reconnaître le visage à travers la diffusion anonyme de la vidéo via la caméra en définissant un cadre spécifique qui apparaît sur le visage en cas de personne devant la caméra, Pour ce faire, nous avons utilisé deux méthodes, dont la première consiste à déterminer la présence de la personne devant la caméra à l'aide d'un cadre vert. Il devient ensuite rouge avec un message d'avertissement s'il n'est pas identifié car il n'est pas dans la base de données. Soit il est reconnu il apparaît sur le visage un cadre jaune avec un message particulier, pour effectuer ce qui précède nous avons utilisé une carte esp32-cam et nous avons créé un réseau sans fil reliant l'ordinateur (interface utilisateur) et la carte via un routeur Wi-Fi. (Figure II.1)

La deuxième méthode est basée sur l'exactitude dans l'identification du visage même en plaçant de petites images devant la caméra et en montrant cela à travers un cadre vert montrant où le visage est en face de la caméra, qui nous a appelés à utiliser la bibliothèque Opencv et le langage de programmation python tout en utilisant la caméra d'ordinateur pour recevoir des entrées.

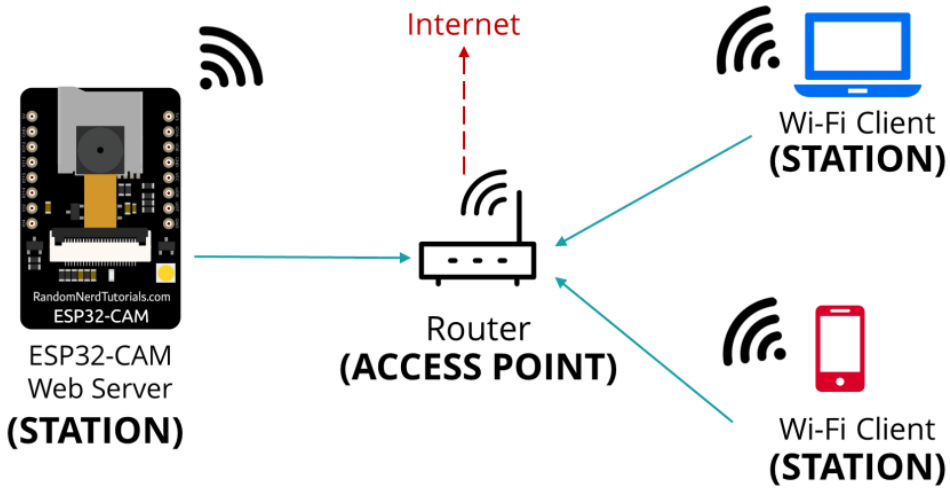


Figure II.1. Réseau sans fil du système de RF utilisant ESP32-CAM

II. 3 La première Méthode :

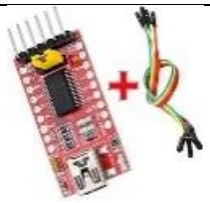
Streaming vidéo et reconnaissance faciale avec Esp32-cam et Arduino IDE

II. 3. 1 Outils de développement

II. 3. 1. 1 Le Hardware

Voici une présentation générale des composants que nous avons utilisés :

- Carte esp32-cam
- Cable FTDI
- Cable USB
- Pc
- Modem

Nom de Outils	**Photo de Outils**
Carte esp32-cam	
Cable FTDI	

Cable USB	
Pc(ordinateur)	
Modem	

Tableau IIII.1. Matériels utilisés dans premier partie

II. 3. 1. 2 Le Software:

- Logiciel Arduino ide.

II. 3. 2 LES ETAPES D'IMPLEMENTATION

II. 3. 2. 1 Installation et configuration ESP32 avec Arduino IDE

Nous installons le panneau esp32 sur Arduino IDE, Logiciel Arduino ide n'ayant pas le package ESP32 par défaut, il faut le télécharger depuis le site officiel de la carte via le lien suivant : https://dl.espressif.com/dl/package_esp32_index.json.

Après avoir installé le package ESP32, nous ouvrons le code du programme en suivant les étapes suivantes dans le logiciel Arduino IDE : File> Exemples> ESP32> Caméra et ouvrez l'exemple **cameraWebServer**.

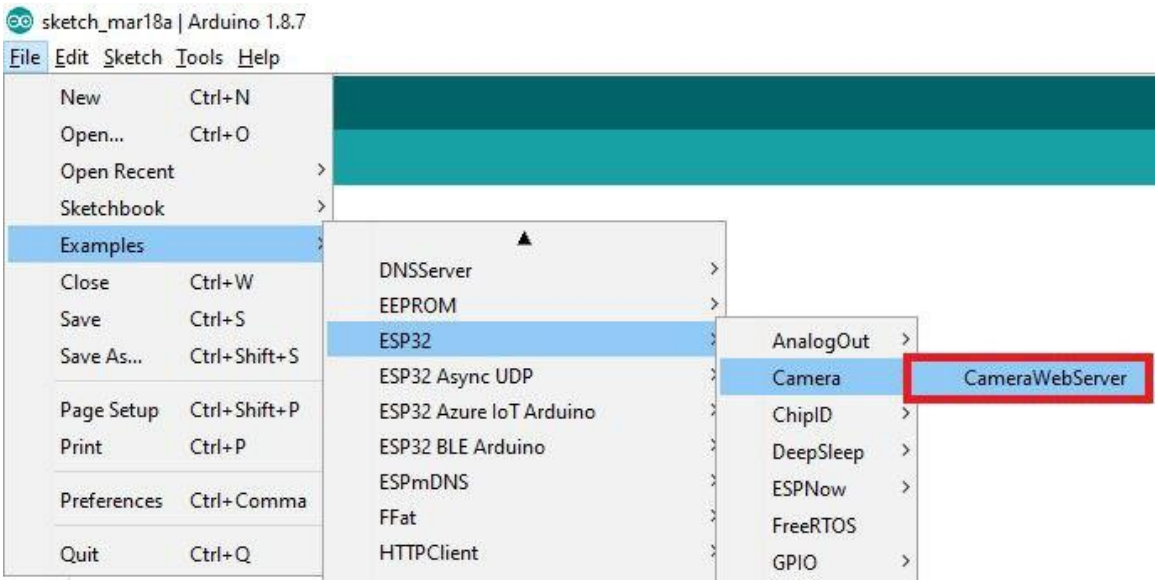


Figure II.2. Fichier de programme dans Arduino IDE

II. 3. 2. 2 Connexion ESP32-CAM au module FTDI

La programmation de la carte ESP32-CAM est similaire à celle des modules ESP32, à une différence majeure près. La carte ESP32-CAM n'a pas de port USB, elle ne peut donc pas être connectée directement à un ordinateur. L'utilisation d'un convertisseur USB-série est donc nécessaire.

Le module de conversion USB-TTL (ou module FTDI) est un module interface qui permet d'adapter des signaux TTL d'une liaison série aux signaux USB. Il existe beaucoup de modules FTDI basés sur des puces CP2102 ou CP2104.

La connexion entre la carte ESP32-CAM et le module FTDI se fait de la manière résumée dans le tableau (Tableau II.2) et comme montrée dans la figure (Figure II.3).

ESP32-CAM	FTDI PROGRAMMER
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

Tableau II.2. Connexion ESP32-CAM et module FTDI

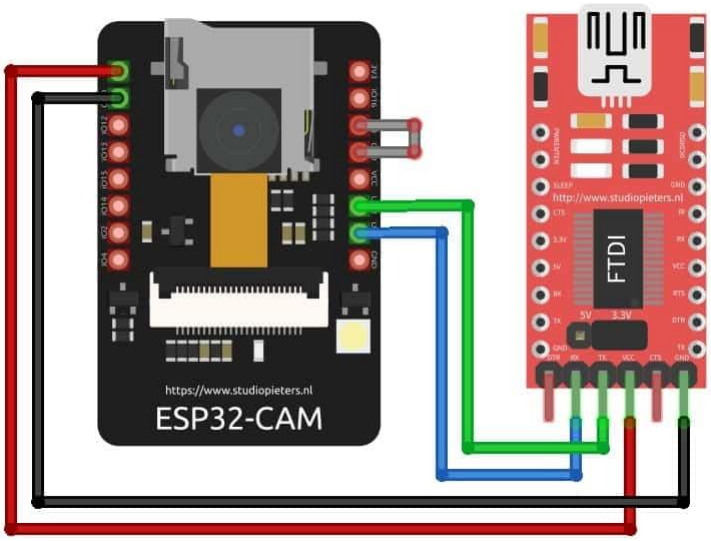


Figure II.3. Connexion ESP32-CAM FTDI

L'utilisation du module FTDI peut être compensée par l'utilisation de la carte Micro USB ESP32-CAM-MB

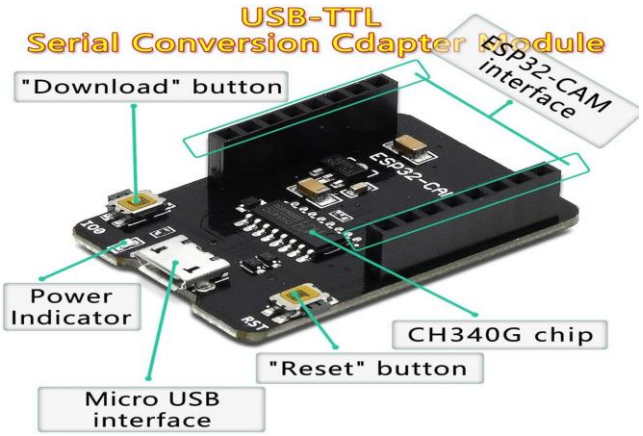


Figure II.4. Un carte Micro USB ESP32-CAM-MB

II. 3. 2. 3 La réalisation finale de notre projet

Nous connectons le panneau ESP32-CAM à votre ordinateur à l'aide du programmeur FTDI comme dans l'image suivante.

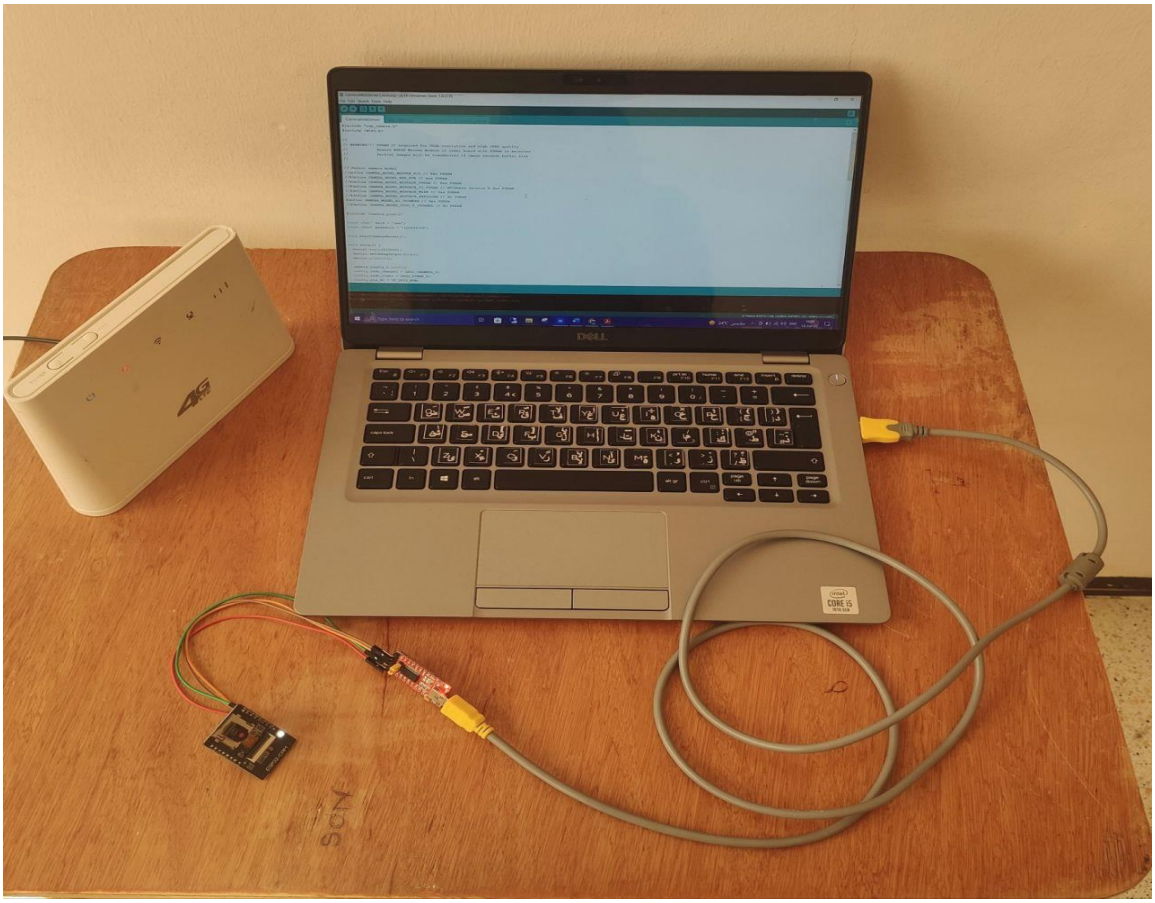


Figure II.5. La réalisation finale

II. 3. 2. 4 Programmation de la carte ESP32-cam

Avant de télécharger le code sur le panneau esp32-cam, nous devons inclure les informations d'identification réseau dans les variables suivantes :

```
const char* ssid = " Nom du réseau Wi-Fi ";
const char* password = " Mot de passe Wi-Fi ";
```

Ensuite, nous identifions la bonne unité de caméra, auquel cas nous utilisons un modèle AI-THINKER :

```
Select camera model

//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
//#define CAMERA_MODEL_M5STACK_ESP32CAM
 #define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_TTGO_T_JOURNAL
```

Par la suite, on va à **Outils** sur Arduino IDO > nous sélectionnons **AI_THINKER ESP32-CAM**, puis nous sélectionnons le port **COM** qui est connecté à esp32 via un câble USB.

- Nous appuyons sur le bouton de téléchargement  Pour vérifier et télécharger le code sur la carte ESP32-CAM.
- Lorsque vous commencez à voir ces points dans la fenêtre de correction d'erreur comme indiqué ci-dessous, nous appuyons sur le bouton RST sur ESP32-CAM. Ensuite, nous attendons que le code termine le téléchargement.

- Nous ouvrons Serial Monitor sur Arduino IDE à un taux baud de 115200 en appuyant sur le bouton de réinitialisation esp32-cam intégré au bouton RST ESP32-CAM ; l'état de la connexion Wi-Fi, l'unité de caméra et l'adresse IP esp32 du serveur Web seront imprimés sur Serial Monitor comme indiqué dans l'image suivante.

Figure II.6. Adresse IP sur serial monitor

II. 3. 2. 5 Les résultats du système de détection et de reconnaissance faciale

Nous pouvons accéder au serveur de streaming de caméra sur un réseau, ouvrir un navigateur et écrire l'adresse IP ESP32CAM.Nous appuyons sur le bouton de démarrage pour commencer à diffuser la vidéo.

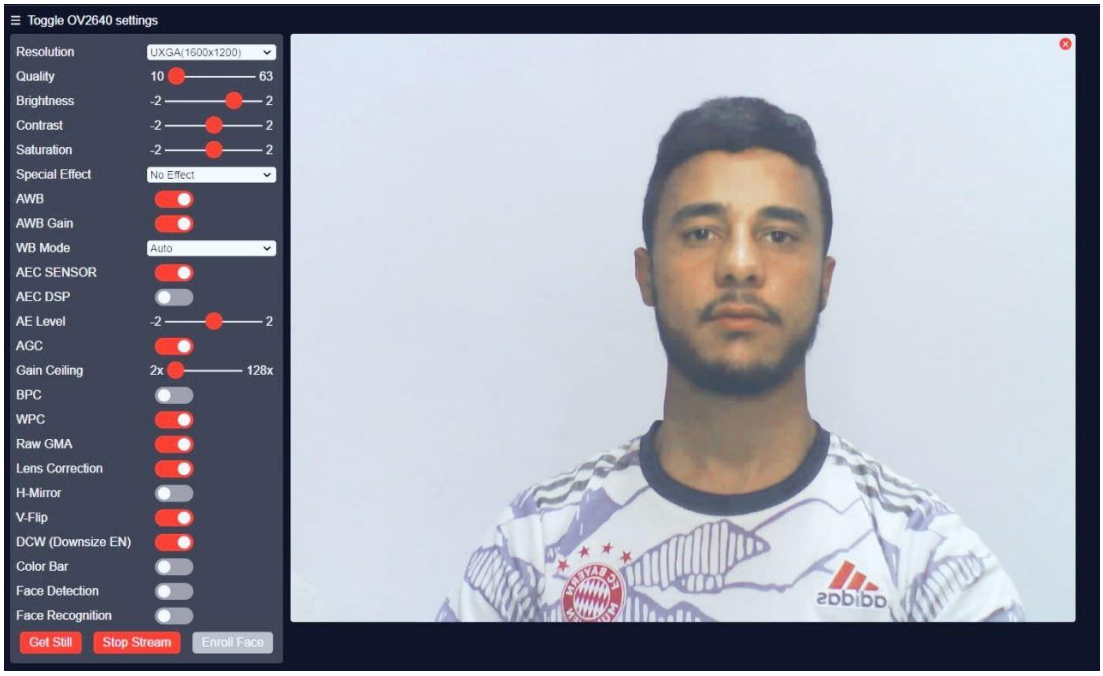


Figure II.7. Extraire le lien de connexion via le navigateur

Détectez et reconnaissance faciale en activant les primes de découverte faciale identifiez le visage dans le coin gauche du côté inférieur, puis cliquez sur le bouton : **Enroll face.**

Figure II.8. Détection de visage (1)

Ensuite, l'utilisateur reste devant la caméra pendant quelques secondes jusqu'à ce que le processus de sauvegarde du visage dans la base de données soit terminé, Et un cadre jaune apparaît autour du visage avec le message -Hello sir -.

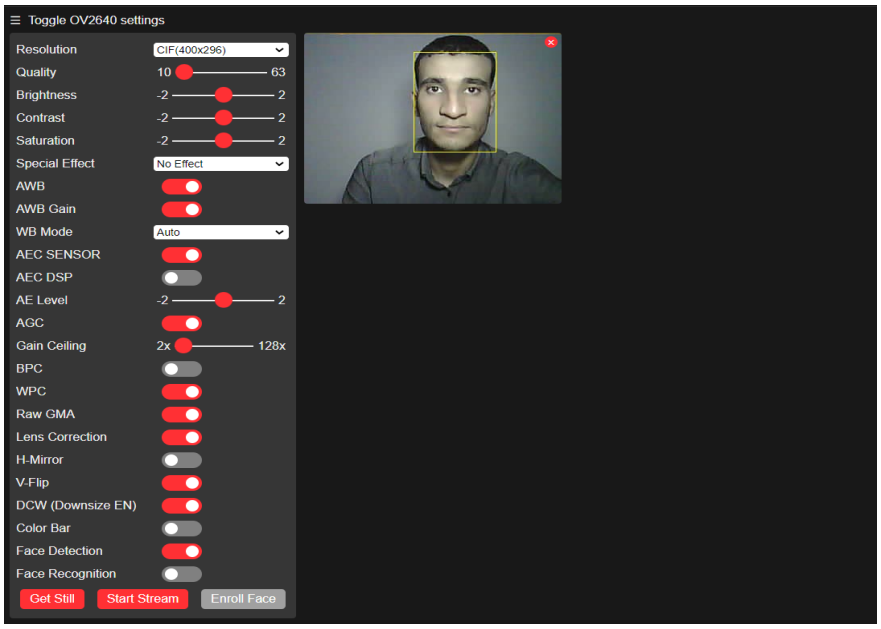


Figure II.9. Détection de visage (2)

- Lorsqu'une personne non enregistrée fait face à la caméra, vous la reconnaîtrez comme un intrus et afficherez une alerte de cadre rouge comme dans l'image (Figure II.9)

Le résultat final :

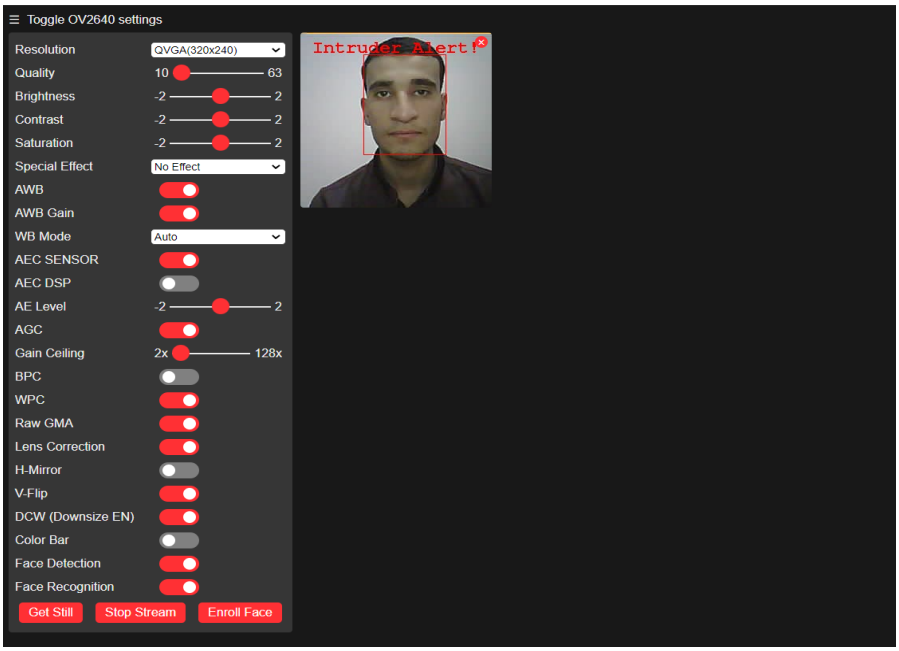


Figure II.10. Le résultat final (pas autorisé)

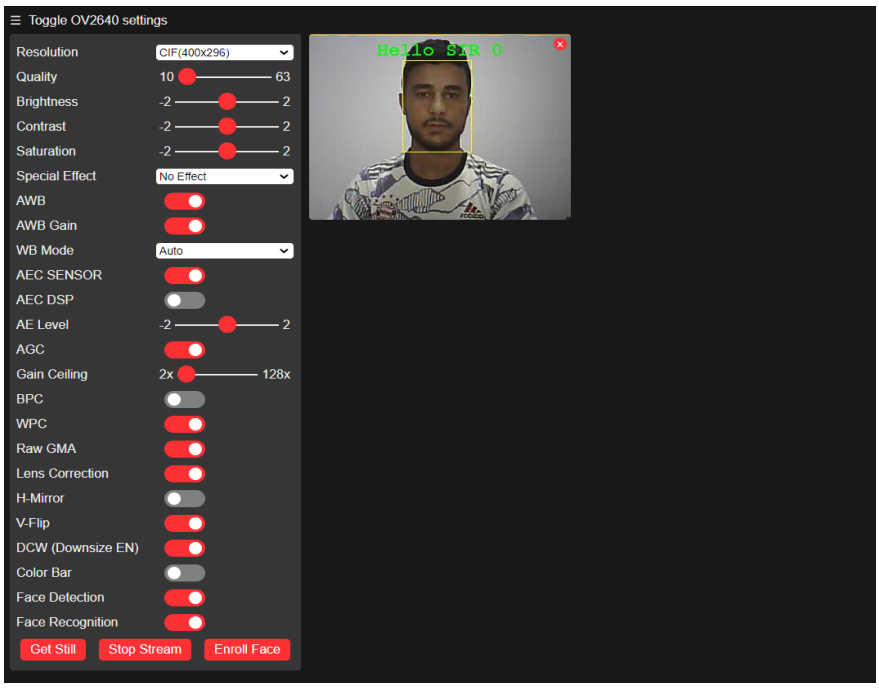


Figure II.11. Le résultat final (autorisé)

II. 4 La deuxième Méthode :

Streaming vidéo et reconnaissance faciale avec webcam et Visual studio Code

Dans la seconde partie, on fera un système de reconnaissance faciale et de streaming vidéo basé sur le langage de développement Python et la librairie OpenCV via webcam.

II. 4. 1 Outils de développement

II. 4. 1. 1 Le Hardware :

Device name: DESKTOP-52SM0IQ.

Processor: Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz.

Installed RAM: 8.00 GB (7.73 GB usable).

System type: 64-bit operating system, x64-based processor.

Camera: webcam 2MP.

II. 4. 1. 2 Le Software:

Système d'exploitation : Windows 10 Pro Version : 21H2.

Logiciel : Visual studio Code.

La bibliothèque : OpenCV 4.5.5.

II. 4. 2 LES ETAPES D'IMPLEMENTATION

II. 4. 2. 1 Installation et configuration OpenCV-Python avec Visual studio Code

Premièrement on doit télécharger **Visual studio Code** depuis son site officiel : https://code.visualstudio.com/download, le logiciel doit être compatible à notre système d'exploitation (Windows).

Afin de vérifier que VS Code est bien installé et fonctionne correctement, nous allons lancer **Visual studio Code.**

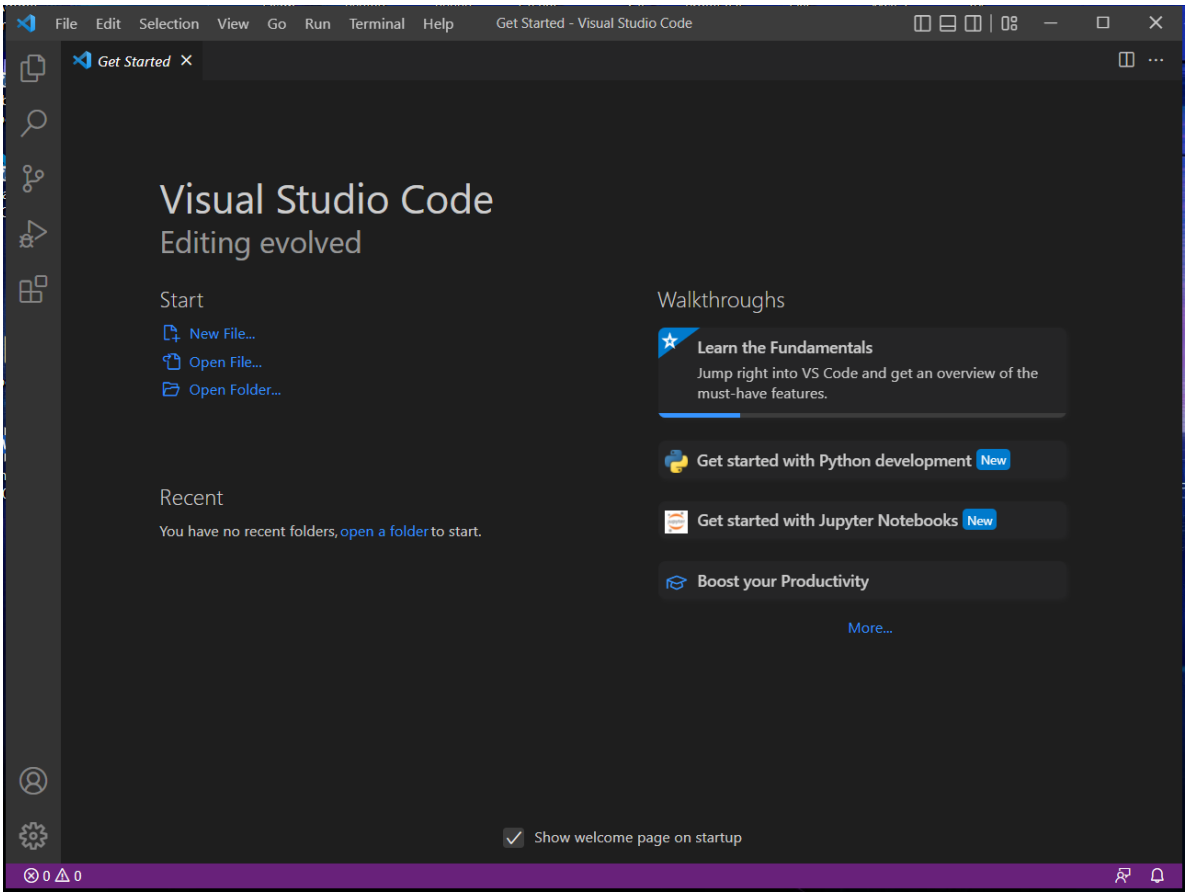


Figure II.12. Ouverture de VS Code sur Windows 10

II. 4. 2. 2 Téléchargement et installation Python3.10 :

Apres l'installation de VSCode nous devons installer Python 3.10, pour ce faire :

a) Cliquer sur l'onglet: EXTENSIONS
b) Cherchez dans la boîte de recherche de mot : Python, Puis on appuie sur le premier résultat.
c) Puis on l'installe en appuyant sur : install.
d) Lorsque l'installation est terminée, **Python** est prêt pour l'utilisation.

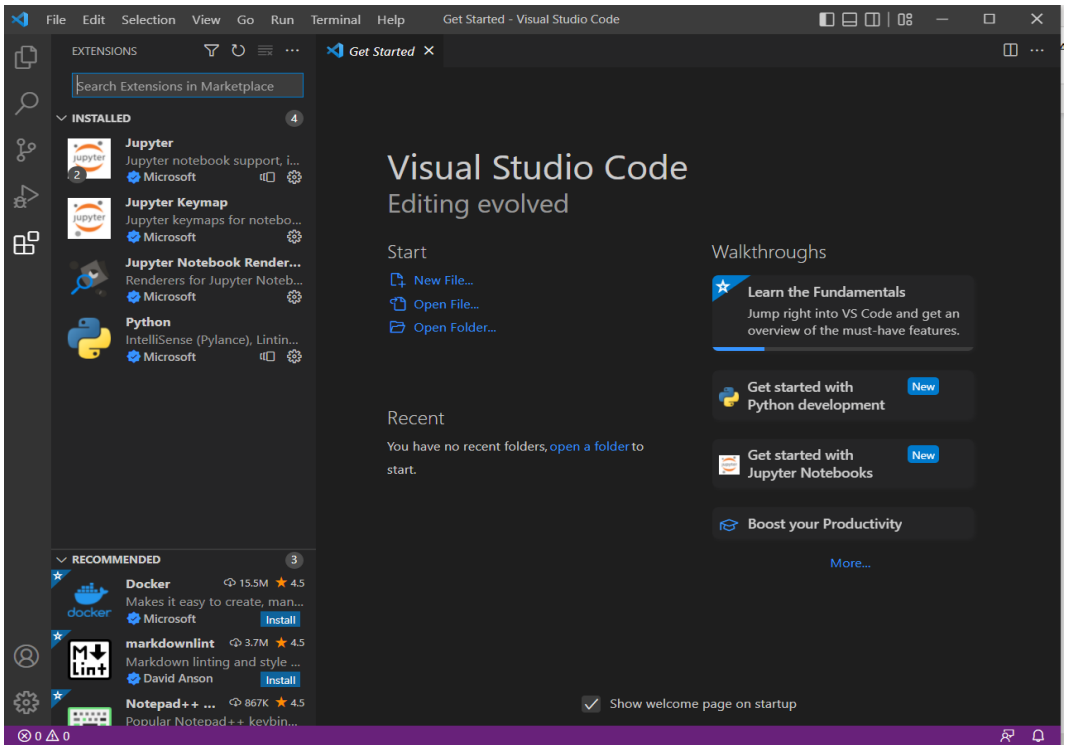


Figure II.13. Installation Python (1)

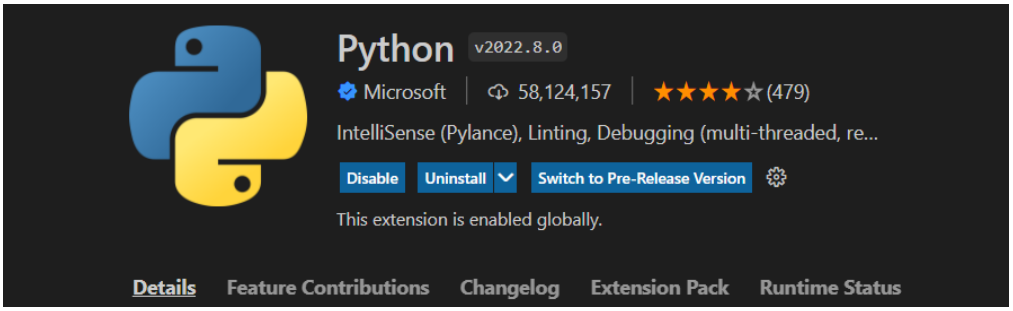


Figure II.14. Installation Python (2)

II. 4. 2. 3 Téléchargement et installation d'OpenCV :

OpenCV peut être directement téléchargé et installé à l'aide de **pip** (gestionnaire de paquets). Pour installer OpenCV, il suffit d'aller dans la ligne de commande et de taper la commande suivante :

```
pip install opencv-python
```

En commençant par l'installation :

- Tapez la commande dans le Terminal et continuez :

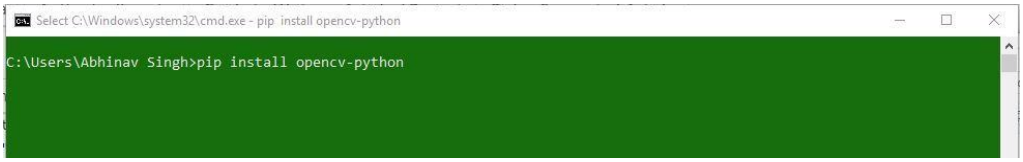


Figure II.15. Installation OpenCV (1)

- Installation terminée :

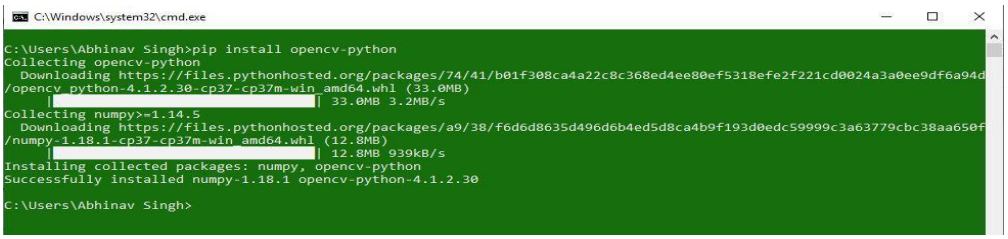


Figure II.16. Installation OpenCV (2)

Pour vérifier si OpenCV est correctement installé, exécutez simplement les commandes suivantes pour effectuer une vérification de version : **[12]**

Python

>>>import cv2

>>>print(cv2. __version__)

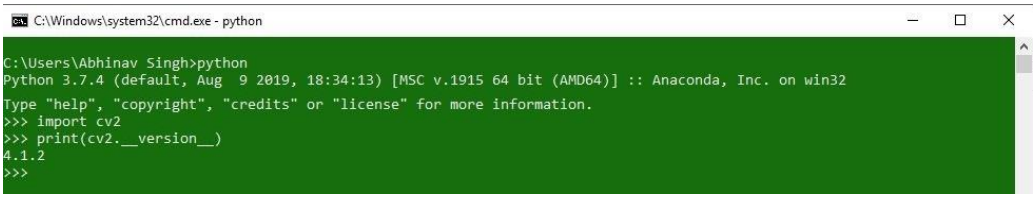


Figure II.17. Installation OpenCV (3)

II. 4. 3 Implémentation et résultats :

On commence par importer cv2 (librairie OpenCv) et sys, afin d'utiliser des fichiers contenant des données OpenCV utilisées pour détecter des objets, puis on crée une cascade de faces.

```
import cv2

import sys


cascPath = "haarcascade_frontalface_default.xml"

faceCascade = cv2.CascadeClassifier(cascPath)

log.basicConfig(filename='webcam.log',level=log.INFO)
```

Après cela, nous définit la source vidéo à la webcam par défaut, que OpenCV peut facilement capturer.

```
video_capture = cv2.VideoCapture(0)
```

Ici, nous capturons la vidéo. La fonction **read()** lit une image de la source vidéo (webcam). Ceci renvoie :

- Le cadre vidéo réel lu (un cadre sur chaque boucle)
- Un code de retour

Le code de retour nous indique si nous sommes à court de cadres, ce qui se produira si nous lisons à partir d'un fichier. Cela n'a pas d'importance lors de la lecture de la webcam.

```
while True:

    if not video_capture.isOpened():

        print('Unable to load camera.')

        sleep(5)
```

```
        pass
    # Capture frame-by-frame
    ret, frame = video_capture.read()
```

Dans cette partie du code, nous cherchons simplement le visage dans notre cadre capturé.

```
 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30)
    )
    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    if anterior! = len(faces):
        anterior = len(faces)
        log.info ("faces: "+str(len(faces)) +" at "+str(dt.datetime.now()))
    # Display the resulting frame
    cv2.imshow('Video', frame)
```

Pour finaliser, nous afficherons le résultat, déjà avec le rectangle autour du visage de la personne, dans une fenêtre à l'aide de la fonction $cv2.\ imshow$ ().

```
# Display the resulting frame
cv2.imshow('Video', frame)
```

```
# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

II. 4. 3. 1 Résultat après exécution du code :

Après avoir exécuté le code du programme, le visage est déterminé de manière précise, même à partir de photos du téléphone ou de la carte personnelle.

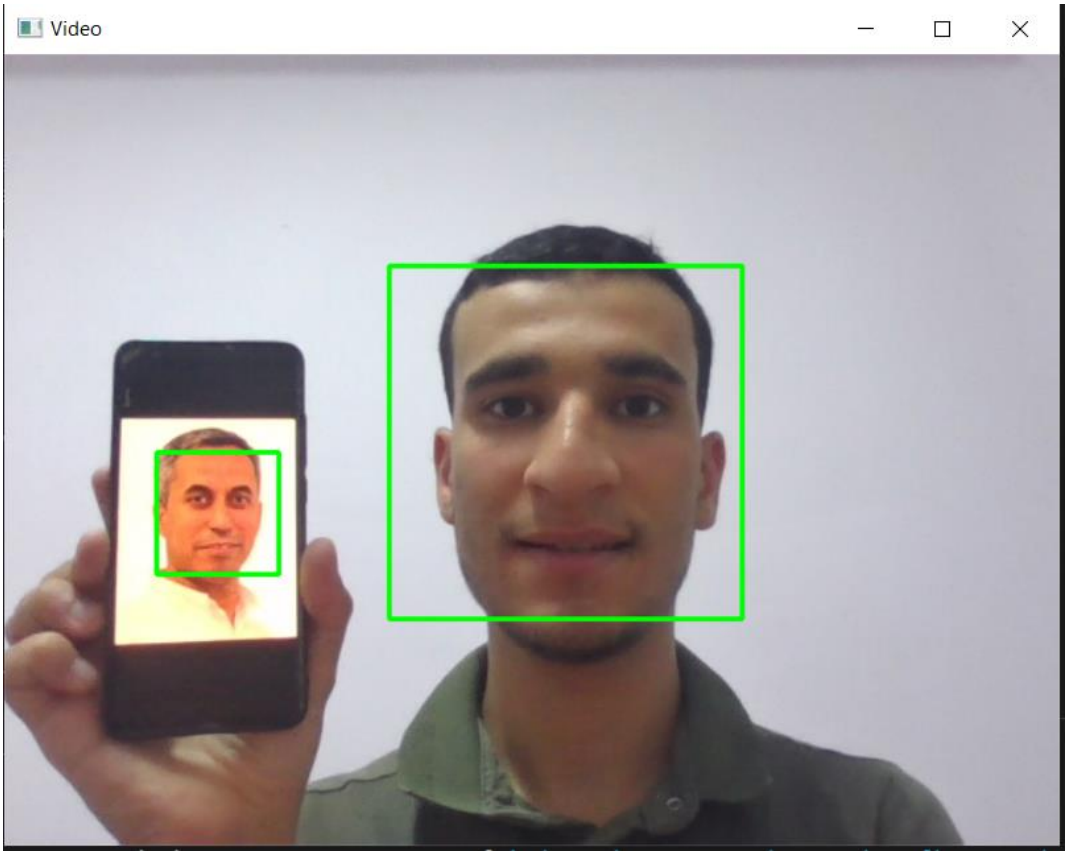


Figure II.18. Le résultat final

II. 5 Conclusion :

Dans ce chapitre, nous avons entamé notre réalisation pratique. Pour cela nous avons essayé de décrire les procédures et les étapes que nous avons menées afin d'arriver à notre but.

Nous avons commencé ce chapitre par une description générale du projet, puis nous avons expliqué la première méthode de reconnaissance faciale et nous avons montré les résultats obtenus, ensuit nous avons enregistré les étapes de la deuxième méthode avec les résultats, de sorte que les outils et le logiciel ont été clarifiés pour chaque méthode.

L'utilisation de la carte esp32cam, malgré les fonctionnalités limitées de la caméra, s'est avérée très appropriée dans l'achèvement du système de reconnaissance faciale, qui prédit un avenir prometteur pour la carte dans le domaine de la reconnaissance facial.

Conclusion générale

Le but de notre projet est de réaliser le système de reconnaissance faciale et nous avons utilisé la carte ESP32 comme caméra principale ainsi que le logiciel Arduino IDE. D'autre part, nous avons utilisé la bibliothèque Opencv et le langage Python pour créer une application de reconnaissance faciale à travers une caméra web en utilisant l'environnement de développement de Visual Studio Code.

Pour réaliser notre travail, nous avons suivi différentes étapes et procédures, à partir de l'étude théorique du système de reconnaissance faciale et des phases d'application progressive de la technique, puis nous avons détaillé les outils utilisés dans la première ou la deuxième méthode.

Nous avons commencé la partie appliquée dans la première méthode, où nous avons connecté la carte ESP32-CAM au réseau Wi-Fi local et la même chose à un ordinateur qui a utilisé comme une interface utilisateur du système, Notre utilisation de la carte ESP32-cam est précisément une référence à son prix bon marché et sa disponibilité sur des fonctionnalités supplémentaires qui n'existent pas sur le marché, en particulier le Wi-Fi, car il est important pour notre projet, Dans le processus de développement, nous tournons vers le logiciel populaire Arduino IDE et enfin afficher les résultats par l'interface utilisateur dans le navigateur; Dans la deuxième méthode, nous avons utilisé la bibliothèque Opencv et langage de programmation Python, mais cette fois nous avons utilisé un ordinateur de caméra pour augmenter la résolution de l'image.

Les nombreuses difficultés, la perturbation des composants électroniques et les nombreux ralentissements que nous avons connus nous ont permis de comprendre nos erreurs dans notre approche et notre stratégie de développement. Il nous a permis de faire des mises à jour de notre projet et des améliorations qui ont fait la différence dans la façon dont nous utilisons la carte ESP32-cam par rapport à la qualité de la caméra informatique qui nous a donné un aperçu des résultats souhaités de notre système.

Le processus de reconnaissance faciale dans les deux systèmes s'est bien déroulé selon nos résultats et nous espérons continuer à développer ce projet car il soutient un domaine critique dans le monde technique d'aujourd'hui en augmentant la qualité de l'image et la vitesse de la reconnaissance faciale pour le rendre pratique et utile dans la vie humaine.

Références bibliographiques

[1] Bruce A. Draper, «Recognizing Faces with PCA and ICA», Department of Computer Science Colorado State University Ft. Collins, CO 80523, U.S.A.

[2] G. BENCHERKI et B. MOUSTAFA, « Implémentation d'un système de reconnaissance de visages à base de PCA », mémoire master, Université Djilali Bounaama Khemis Miliana (2018)

[3] https://www.ethique.gouv.qc.ca/fr/publications/reconnaissance-faciale/

[4] S. BOUANIK et S. BOUDJATAT**,** « Conception et réalisation d'un système de capteurs sans fil pour la maintenance prédictive des machines », mémoire master, Centre Universitaire Abdelhafid Boussouf - Mila (2020)

[5] https://lastminuteengineers.com/esp32-arduino-ide-tutorial/

[6] https://www.tme.eu/fr/news/library-articles/page/21733/Une-large-gamme-de-modules-ESP32/

[7] BOUZIR Rami et KACIMI Athman**,** « Réalisation d'un bras manipulateur à base D'Arduino », mémoire master, Université BOUIRA (2019)

[8] https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/

[9] https://randomnerdtutorials.com/esp32-ble-server-client/

[10] MEKHALFIA Toufik et GHADBANE Toufik, « Etude et réalisation d'un système de commande à distance des installations électriques pour la domotique », mémoire master, Université M'SILA (2018)

[11] HADJAR Yacine et MEDRAR silia, « Véhicule intelligent pour la détection des plaques d'immatriculation suspectes », mémoire master, Université TIZI OUZOU (2019)

[12] https://www.geeksforgeeks.org/how-to-install-opencv-for-python-in-windows/