

# **MINI PROJECT**

**(EC681)**

**Face Recognition based attendance system using  
ESP32 cam**

Under the Guidance of

**Prof. Anup Kumar Mallick**

(Asst. Professor, Electronics & Communication Engineering)

SEMESTER-VI

JUNE- 2023

Department of

Electronics & Communication Engineering

**Kalyani Government Engineering College**

Kalyani, Nadia, West Bengal-741235



## Index

<b>SL No.</b>	<b>Topics</b>	<b>Page No.</b>
1	Abstract	5
2	Introduction	6
3	Objective	7
4	Components Required	8
5	Circuit Diagram	9
6	Circuit analysis	10
7	Component description	11 - 12
8	Software requirements	13 - 14
9	Working principle	15 - 23
10	Result	24 - 25
11	Application	26
12	Future scope	27
13	Conclusion	28
14	References	29

## Group Members

Sl. no	Name	Roll No.
1	Rajkrishna Rana	10200321044
2	Aman Kumar Singh	10200321050
3	Gouranga Ghosh	102003210060
4	Saumalya Das	10200321063

# CERTIFICATE

This is certify that the project entitled “**Face Recognition based attendance system using ESP32 cam**” has been completed under the guidance of **Prof. Anup Kumar Mondal Sir** and submitted as a mini Project to **Kalyani Government Engineering College**, in partial fulfilment of the requirement for the award of B.Tech degree in Electronics & Communication Engineering for the session 2021-2024.

---

Signature of the Project Guide

---

Examiner Signature

# ABSTRACT

The face recognition system using ESP32-CAM, OpenCV, and Python is a project aimed at developing an efficient and portable solution for face recognition. The system utilizes the ESP32-CAM module, which combines an ESP32 microcontroller, camera module, and Wi-Fi connectivity, along with the powerful OpenCV library and the Python programming language. The methodology involves capturing and preprocessing facial images, detecting faces using OpenCV's face detection algorithms, training a face recognition model using machine learning techniques, and implementing real-time face recognition on the ESP32-CAM module. The system holds potential applications in security, surveillance, access control, and personalized user experiences. The developed system provides a foundation for further research and improvements, such as performance optimization, multi-factor authentication, real-time tracking, and integration with IoT and cloud services. Through this project, the face recognition system demonstrates its capability to recognize the person and implement it as an attendance system (i.e. it marks the person is present and also note the time when its attendance is recorded).

# INTRODUCTION

The face recognition system using ESP32-CAM, OpenCV, and Python is a project aimed at developing a robust and efficient face recognition solution. Face recognition technology has gained significant attention in recent years due to its wide range of applications in security, surveillance, access control, and personalized user experiences. By combining the capabilities of the ESP32-CAM module, OpenCV library, and Python programming language, we can create a versatile and portable system for facial recognition.<sup>[3]</sup>

The ESP32-CAM module serves as the hardware platform for this project. It integrates an ESP32 microcontroller, camera module, and Wi-Fi connectivity, making it an ideal choice for implementing a face recognition system. The OpenCV library, a popular computer vision library, provides a comprehensive set of functions and algorithms for image processing, including face detection and recognition. Python, a widely used programming language, offers a flexible and user-friendly environment for developing the face recognition system.<sup>[1]</sup>

The face recognition system involves two main stages: face detection and face recognition. In the face detection stage, the system identifies and locates faces within an image or video stream using computer vision algorithms provided by OpenCV. Once the faces are detected, the face recognition stage begins, where the system matches the detected faces against a pre-existing database of known individuals to identify them.<sup>[3]</sup>

The developed face recognition system has practical implications in various domains. It can enhance security systems by accurately identifying individuals and enabling access control. It can also be applied in surveillance scenarios, enabling real-time monitoring and identification of people. Additionally, the system can provide personalized user experiences by recognizing individuals and customizing interactions accordingly.<sup>[1]</sup>

## **OBJECTIVE**

The objective of the attendance system using the ESP32 cam and face detection is to automate and streamline the attendance tracking process. By leveraging face detection technology, the system aims to provide accurate and reliable attendance records, minimizing errors and fraud. It also enables real-time monitoring of attendance, allowing for immediate feedback and intervention if needed. Additionally, the system aims to generate comprehensive attendance reports and analytics to provide valuable insights for decision-making and optimize attendance management processes. <sup>[2]</sup>

## COMPONENTS REQUIRED

SL. No.	COMPONENTS	SPECIFICATION	QUANTITY
1	ESP32-CAM Board AI-Thinker	32-bit CPU, 520 KB SRAM, Wi-Fi + Bluetooth/BLE	1
2	FTDI Module	TTL 3.3V/5V	1
3	Mini-USB Cable	FTDI Compatible	1
5	Bread Board	17x5 cm <sup>2</sup>	1
6	Connecting Wires	Jumpers	10
7	Power Supply	5v (from USB)	1



# CIRCUIT DIAGRAM

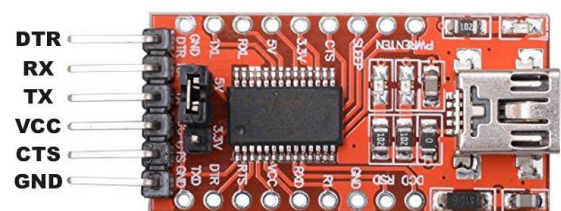
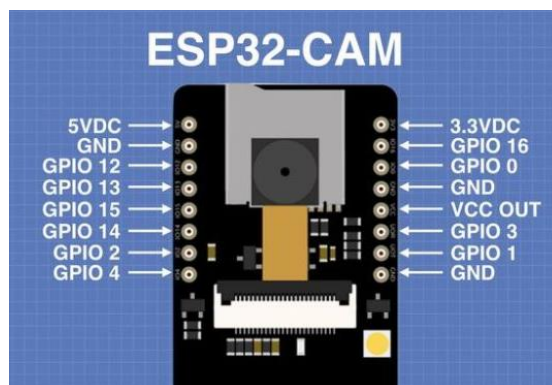
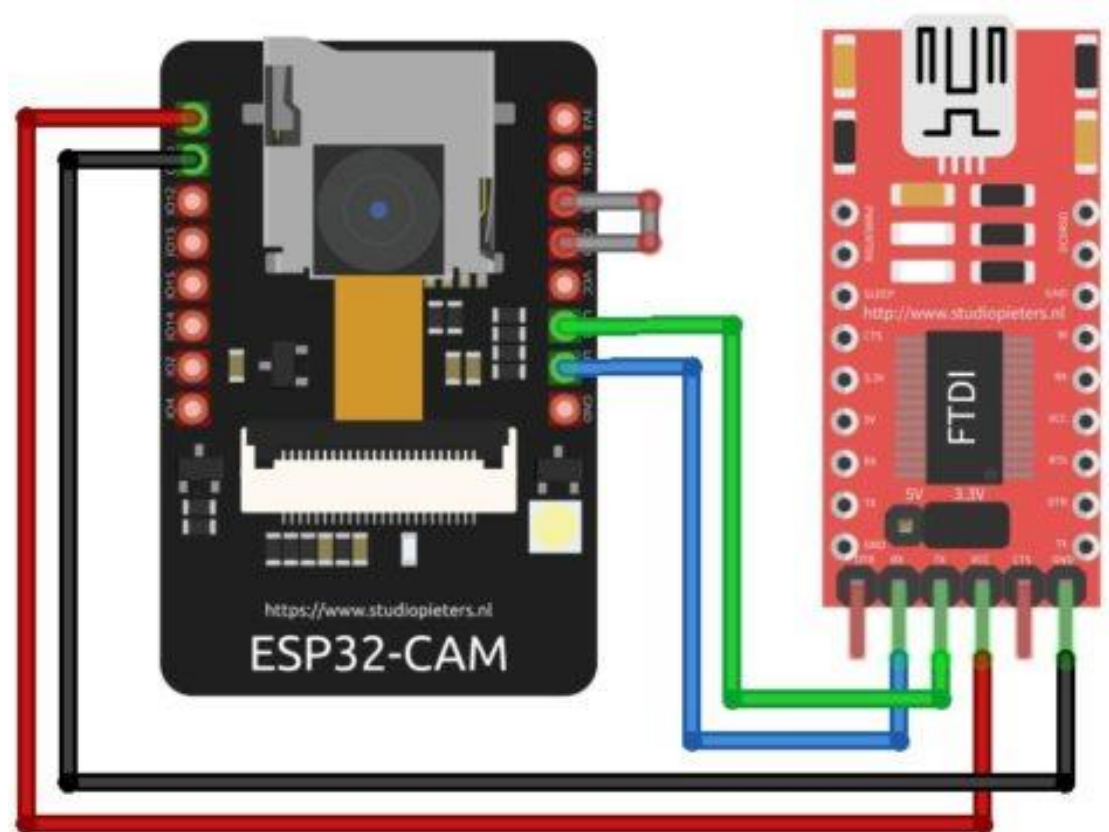


Fig 1 : Circuit Diagram

## CIRCUIT ANALYSIS

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC
U0R	TX
U0T	RX
GPIO0	GND

Connect the **5V** & **GND** Pin of ESP32 to 5V & GND of FTDI Module. Similarly, connect the **Rx** to **U0T** and **Tx** to **U0R** Pin. And the most important thing, we need to short the **IO0** and **GND** Pin together. This is to put the device in **programming mode**. Once programming is done we can remove it. <sup>[3]</sup>

# COMPONENTS DESCRIPTION

## 1. ESP32-CAM Board AI-Thinker :

The ESP32-CAM board is a development board based on the ESP32 system-on-chip (SoC) and is designed for applications involving camera functionality. The board is manufactured by AI-Thinker, a Chinese company that specializes in IoT solutions. Here are some key details and features of the ESP32-CAM board:

**Microcontroller:** The board is powered by the ESP32-WROOM-32 module, which contains the ESP32 SoC. The ESP32 is a powerful dual-core microcontroller with Wi-Fi and Bluetooth capabilities.<sup>[3]</sup>

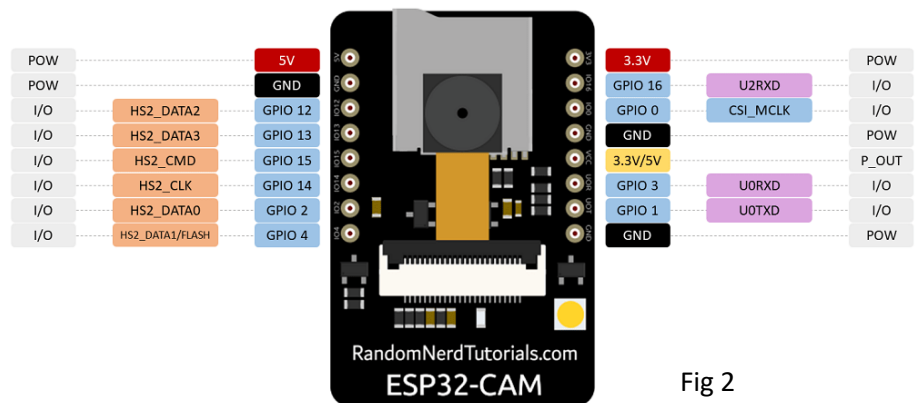


Fig 2

**Camera:** The ESP32-CAM board features a camera module that enables image and video capture. The camera module uses an OV2640 image sensor, which is a 2-megapixel camera capable of capturing images and video at various resolutions.<sup>[1]</sup>

**Memory and Storage:** The ESP32-CAM board has 4MB of flash memory for storing firmware and programs. It also has additional PSRAM (pseudo static random-access memory) of 8MB, which provides extra memory for data storage and processing.

**Connectivity:** The ESP32-CAM board includes Wi-Fi and Bluetooth connectivity. It supports 802.11b/g/n



Fig 3: ESP32 CAM

Wi-Fi standards for wireless communication and can connect to Wi-Fi networks for data transfer. It also supports Bluetooth Classic and Bluetooth Low Energy (BLE) for wireless communication with other devices.<sup>[1]</sup>

**GPIO Pins:** The board features a set of general-purpose input/output (GPIO) pins that can be used for connecting external sensors, actuators, or other

peripheral devices. The number of GPIO pins available on the board varies depending on the specific variant of the ESP32-CAM board.<sup>[2]</sup>

**Power Supply:** The ESP32-CAM board can be powered through a micro USB port. It requires a 5V power supply, which can be provided through a USB connection or an external power source.

## 2. FTDI Module :

FTDI (Future Technology Devices International) is a company known for producing a range of USB communication and interface devices. One of their popular products is the FTDI module, which refers to a module or board that incorporates an FTDI USB-to-serial converter chip. Here are some details about FTDI modules

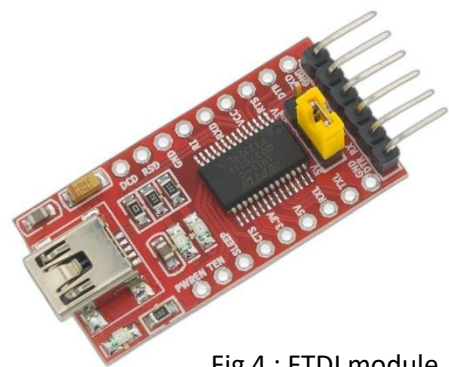


Fig 4 : FTDI module

**FTDI Chip:** The core component of an FTDI module is the FTDI chip itself. FTDI chips are widely used for converting USB signals to serial communication protocols such as UART (Universal Asynchronous Receiver-Transmitter) or RS-232 (Recommended Standard 232). These chips enable easy connectivity between a computer or other USB host devices and microcontrollers, embedded systems, or other serial devices.<sup>[1]</sup>

**USB Interface:** FTDI modules feature a USB interface that allows them to connect to a computer or USB host device. The USB interface is used for both data communication and powering the module, eliminating the need for additional power sources.

**Serial Communication:** FTDI modules typically provide one or more UART or RS-232 interfaces for serial communication. These interfaces can be used to establish communication with external devices, such as microcontrollers, sensors, or other peripherals. The FTDI chip handles the conversion of USB signals to the serial format and vice versa.<sup>[2]</sup>

# SOFTWARE REQUIREMENTS:

## 1. Arduino IDE :

Arduino is the required software environment to program the Arduino by writing code and uploading it to the Arduino. It also outputs the results for analysis using both a serial monitor and a serial plotter.

It is Arduino software, making code compilation too easy. It is available for all operating systems i.e. MAC, Windows, and Linux, and runs on the Java platform that comes with inbuilt functions and

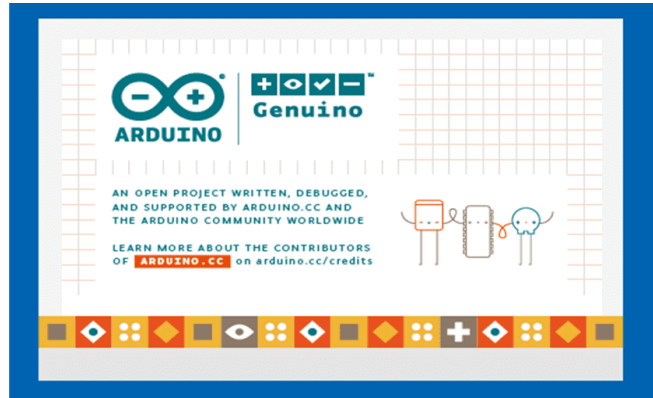


Fig 5 : Arduino Ide

commands that play a vital role in debugging, editing, and compiling the code.

It is easy to use, it supports all Arduino boards, and it has a built-in library which is easy to use. The Arduino IDE is very user-friendly.<sup>[3]</sup>

## 2. Visual Studio :

Visual Studio is a popular integrated development environment (IDE) created by Microsoft. It provides a comprehensive set of tools and features for software development across various platforms, including Windows, macOS, and Linux.

We need to install Visual studio. We are doing so because there are certain dependencies that we will be required, to install the libraries.<sup>[1]</sup>

OpenCV (Open Source Computer Vision) is a popular open-source library for computer vision and image processing tasks. It provides a wide range of functions and algorithms for working



with images and videos. OpenCV is primarily implemented in C++ but also offers Python bindings, making it accessible and widely used in the Python programming language. <sup>[2]</sup>

It can implement Image Manipulation, Computer Vision Algorithm, Machine Learning Integration etc.

### 3. Python IDLE :

Python IDLE (Integrated Development and Learning Environment) is an integrated development environment that comes bundled with the Python programming language. It provides a basic and lightweight environment for writing, running, and debugging Python code.

It has features like :

**Interactive Shell:** Python IDLE includes an interactive shell, also known as the Python shell or REPL (Read-Eval-Print Loop), where we can execute Python code interactively. <sup>[1]</sup>

**Code Editor:** Python IDLE offers a code editor with features like syntax highlighting, auto-indentation.

**Integrated Debugger:** Python IDLE includes a basic debugger that allows we to debug wer Python code. <sup>[2]</sup>

**Code Execution:** Python IDLE allows we to run Python scripts directly from the editor.

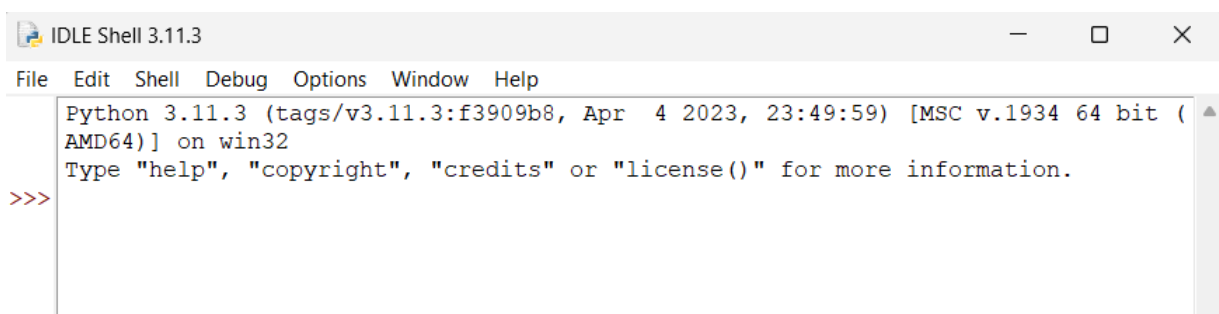


Fig 6 : Python Ide

# WORKING PRINCIPLE

In this Face Recognition Based Attendance System using ESP32 CAM Module. We will be using OpenCV & Visual Studio for this application. OpenCV is an open-sourced image processing library that is very widely used not just in industry but also in the field of research and development. Visual Studio is an IDE made by Microsoft for different types of software development & contains completion tools, compilers, and other features to facilitate the software development process. <sup>[2]</sup>

Face Recognition Based Attendance system using ESP32 CAM and Python. The main heavy program will be at the server-side that is our computer, or one can even use raspberry-pi as a server. In this attendance system, we will not just detect the person but also store the information of the person detected in a Microsoft Excel File. Moreover, the duration of time they have stayed in the frame is also recorded into an excel sheet. <sup>[1]</sup>

Here we explain step by step how we run the program and take the attendance in the excel sheet.

## Step 1: Installing ESP32CAM Library

Here we will not use the general **ESP webserver example** rather another streaming process. Therefore we need to add another ESPCAM library. The esp32cam library provides an object oriented API to use **OV2640 camera on ESP32 microcontroller**. It is a wrapper of **esp32-camera library**. We download this library from **Github**. <sup>[2]</sup>

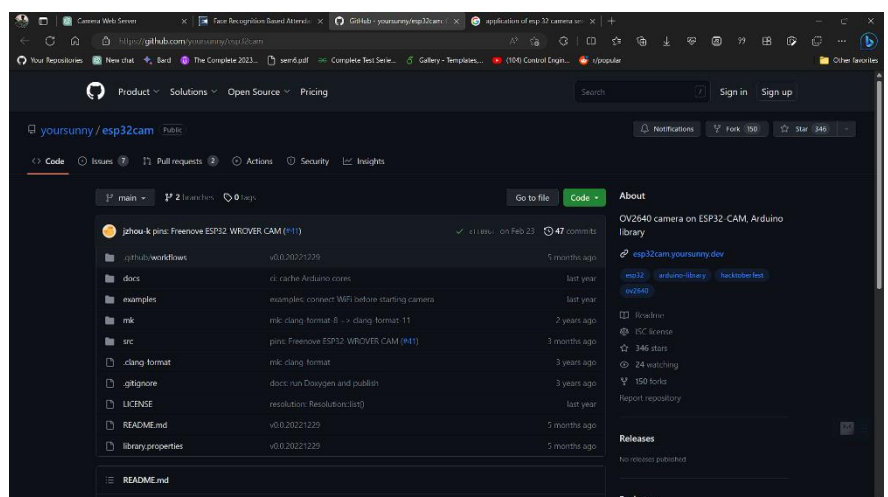


Fig 7 : ESP32CAM lib.



Once downloaded add this zip library to Arduino Libray Folder. To do so follow the following steps:

*Open **Arduino** -> **Sketch** -> **Include Library** -> **Add .ZIP Library...** -> **Navigate to downloaded zip file** -> **add***

## Step 2 :Source Code/Program for ESP32 CAM Module

Here is a source code for Face Recognition Based Attendance System using ESP32 CAM & OpenCV. <sup>[3]</sup>

```
#include <WebServer.h>
```

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

```
#include <esp32cam.h>
```

```
const char* WIFI_SSID = "ssid";
```

```
const char* WIFI_PASS = "password";
```

```
WebServer server(80);
```

```
static auto loRes = esp32cam::Resolution::find(320, 240);
```

```
static auto midRes = esp32cam::Resolution::find(350, 530);
```

```
static auto hiRes = esp32cam::Resolution::find(800, 600);
```

```
void serveJpg()
```

```
{
```

```
    auto frame = esp32cam::capture();
```

```
    if (frame == nullptr) {
```

```
        Serial.println("CAPTURE FAIL");
```

```
        server.send(503, "", "");
```

```
        return;
```

```
    }
```

```
    Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
```

```
        static_cast<int>(frame->size()));
```

```
    server.setContentLength(frame->size());
```

```
    server.send(200, "image/jpeg");
```

```
    WiFiClient client = server.client();
```

Before Uploading the code we have to make a small change to the code. Change the SSID and password variable and in accordance with wer WiFi network.



```

    frame->writeTo(client);
}
void handleJpgLo()
{
    if (!esp32cam::Camera.changeResolution(loRes)) {
        Serial.println("SET-LO-RES FAIL");
    }
    serveJpg();
}
void handleJpgHi()
{
    if (!esp32cam::Camera.changeResolution(hiRes)) {
        Serial.println("SET-HI-RES FAIL");
    }
    serveJpg();
}
void handleJpgMid()
{
    if (!esp32cam::Camera.changeResolution(midRes)) {
        Serial.println("SET-MID-RES FAIL");
    }
    serveJpg();
}
void setup(){
    Serial.begin(115200);
    Serial.println();
    {
        using namespace esp32cam;
        Config cfg;
        cfg.setPins(pins::AiThinker);
        cfg.setResolution(hiRes);
        cfg.setBufferCount(2);
    }
}

```

```

    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
}

WiFi.persistent(false);
WiFi.mode(WIFI_STA);
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
}

Serial.print("http://");
Serial.println(WiFi.localIP());
Serial.println(" /cam-lo.jpg");
Serial.println(" /cam-hi.jpg");
Serial.println(" /cam-mid.jpg");

server.on("/cam-lo.jpg", handleJpgLo);
server.on("/cam-hi.jpg", handleJpgHi);
server.on("/cam-mid.jpg", handleJpgMid);

server.begin();
}

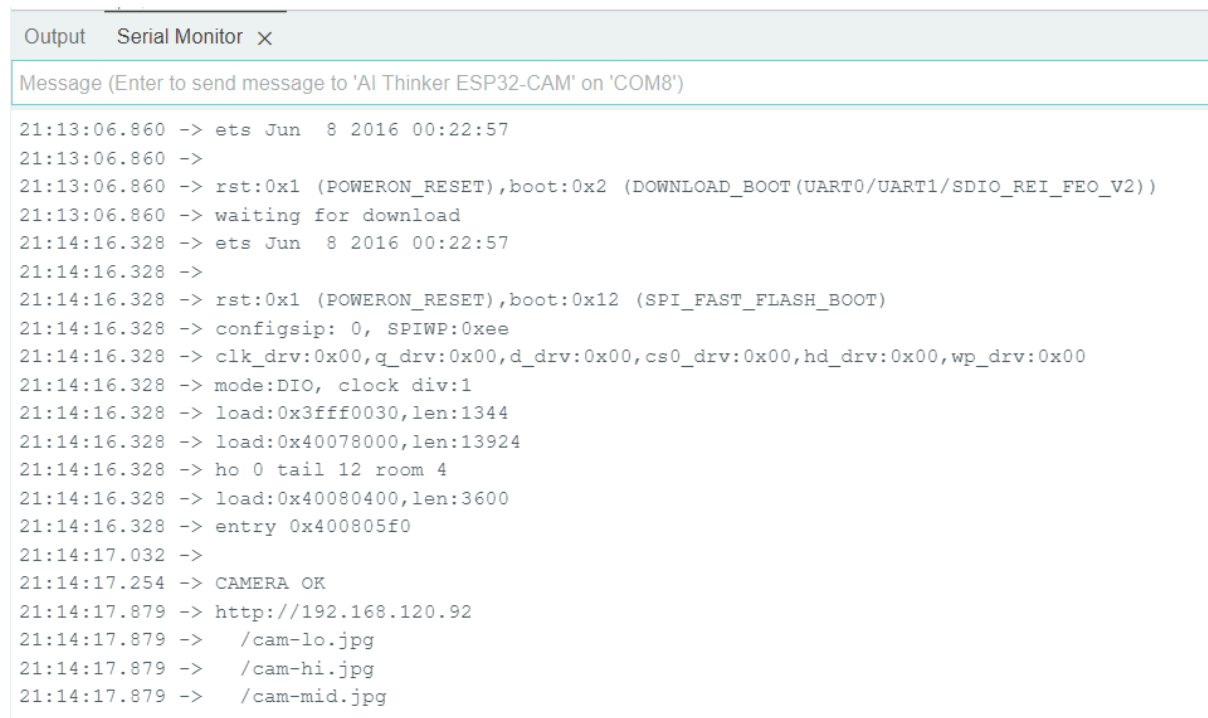
void loop()
{
    server.handleClient();
}

```

Now compile and upload it to the ESP32 CAM Board. But during uploading, we have to follow few steps every time. Make sure the IO0 pin is shorted with the ground when pressed the upload button. If it give the dots and dashes while uploading press the reset button immediately. <sup>[1]</sup>

Once the code is uploaded, remove the I01 pin shorting with Ground and press the reset button once again. If the output is the Serial monitor is still not there then press the reset button again.

Now we can see a similar output as in the image below.



```
21:13:06.860 -> ets Jun  8 2016 00:22:57
21:13:06.860 ->
21:13:06.860 -> rst:0x1 (POWERON_RESET),boot:0x2 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_FEO_V2))
21:13:06.860 -> waiting for download
21:14:16.328 -> ets Jun  8 2016 00:22:57
21:14:16.328 ->
21:14:16.328 -> rst:0x1 (POWERON_RESET),boot:0x12 (SPI_FAST_FLASH_BOOT)
21:14:16.328 -> configip: 0, SPIWP:0xee
21:14:16.328 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
21:14:16.328 -> mode:DIO, clock div:1
21:14:16.328 -> load:0x3fff0030,len:1344
21:14:16.328 -> load:0x40078000,len:13924
21:14:16.328 -> ho 0 tail 12 room 4
21:14:16.328 -> load:0x40080400,len:3600
21:14:16.328 -> entry 0x400805f0
21:14:17.032 ->
21:14:17.254 -> CAMERA OK
21:14:17.879 -> http://192.168.120.92
21:14:17.879 -> /cam-lo.jpg
21:14:17.879 -> /cam-hi.jpg
21:14:17.879 -> /cam-mid.jpg
```

Fig 8 : Serial Monitor Output

Here, copy the IP address visible, we will be using it to edit the URL in python code

### Step 3 : Installing Visual Studio

Now in order to proceed further, we need to install Visual studio. We are doing so because there are certain dependencies that we will be required, to install the libraries, later in this tutorial. To install go to this Visual Studio website. Then, download the latest community version.<sup>[2]</sup>

Once downloaded install the software a welcome installation screen would appear, now select the community version.

After this from numerous selecting boxes, select Desktop development with C++. At the right-hand side select the boxes as in the image below.<sup>[1]</sup>

Now that we have selected the click on Install/Modify at the Right corner below. This process will take time as it will download large files. <sup>[2]</sup>

Once installed it will ask for restarting the PC. So that will be done automatically.

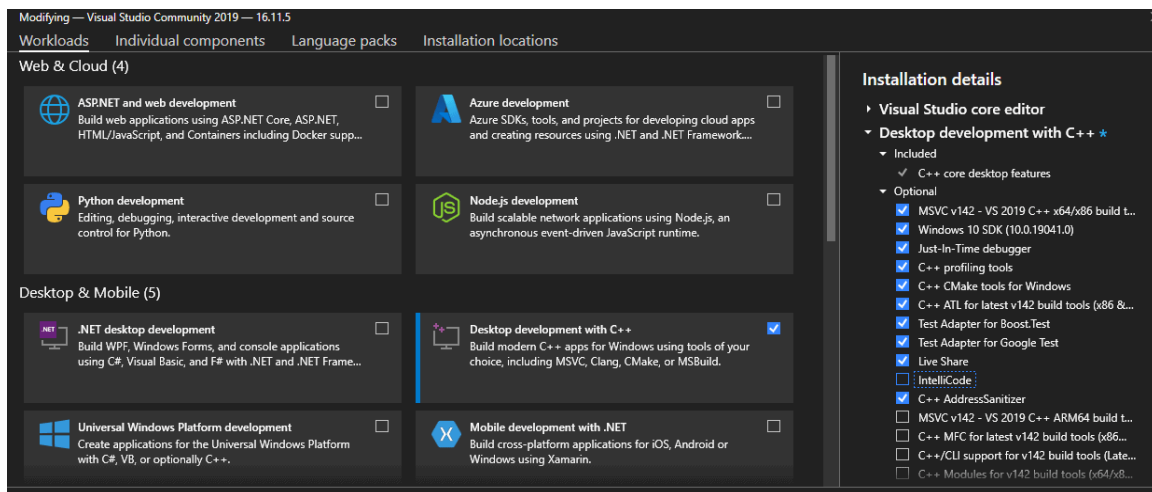


Fig 9 : Visual Studio

#### Step 4: Installing required libraries and run python program:

Now we have to install our required libraries (which is give in the picture) in python project.

chardet==3.0.4, click==7.1.2, cmake==3.18.2.post1, decorator==4.4.2, face-recognition==1.3.0, face-recognition-models==0.3.0, idna==2.10, imageio==2.9.0, imageio-ffmpeg==0.4.2, moviepy==1.0.3, numpy, pandas, opencv-python, Pillow==8.0.1, proglog==0.1.9, requests==2.24.0, tqdm==4.51.0, urllib3==1.25.11, wincertstore==0.2, dlib

First, we need to update the URL variable in the code with the URL copied from Arduino serial monitor earlier. <sup>[1]</sup>

Secondly, update the path variable in the code with the path of the image\_folder folder.

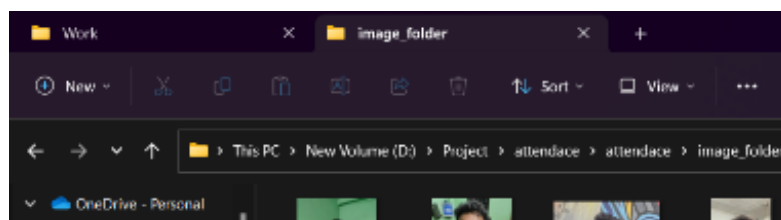


Fig 10 : image folder path

Now we install this libraries with the help of pip command. <sup>[2]</sup>

```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91704>pip install pandas_
```

Fig 11 : installing python libraries

Then we write the following code for the face detection and record the attendance in a excel sheet.

### Python Code for Face Recognition Attendance System. <sup>[3]</sup>

```
import pandas as pd
import cv2
import urllib.request
import numpy as np
import os
from datetime import datetime
import face_recognition

path = r'D:\python\attendance\attendance\image_folder'
url='http://192.168.231.162/cam-hi.jpg'
##"cam.bmp / cam-lo.jpg /cam-hi.jpg / cam.mjpeg ""

if 'Attendance.csv' in os.listdir(os.path.join(os.getcwd(),'attendance')):
    print("there iss..")
    os.remove("Attendance.csv")
else:
    df=pd.DataFrame(list())
    df.to_csv("Attendance.csv")
images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cl in myList:
    curlImg = cv2.imread(f'{path}/{cl}')
    images.append(curlImg)
```

Here we need to update the URL variable in the code with the URL copied from Arduino serial  
.. ..

```

        classNames.append(os.path.splitext(cl)[0])
print(classNames)
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
def markAttendance(name):
    with open("Attendance.csv", 'r+') as f:
        myDataList = f.readlines()
        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
            if name not in nameList:
                now = datetime.now()
                dtString = now.strftime('%H:%M:%S')
                f.writelines(f'\n{name},{dtString}')
encodeListKnown = findEncodings(images)
print('Encoding Complete')
#cap = cv2.VideoCapture(0)
while True:
    #success, img = cap.read()
    img_resp=urllib.request.urlopen(url)
    imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
    img=cv2.imdecode(imgnp,-1)
# img = captureScreen()
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
    facesCurFrame = face_recognition.face_locations(imgS)

```

```

encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
    matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
    faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
# print(faceDis)
    matchIndex = np.argmin(faceDis)
    if matches[matchIndex]:
        name = classNames[matchIndex].upper()
# print(name)
        y1, x2, y2, x1 = faceLoc
        y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
        cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
        markAttendance(name)
cv2.imshow('Webcam', img)
key=cv2.waitKey(5)
if key==ord('q'):
    break
cv2.destroyAllWindows()
cv2.imread

```

# RESULT

Now we are good to go. Below image is show our database preloaded image which we use for our model training.

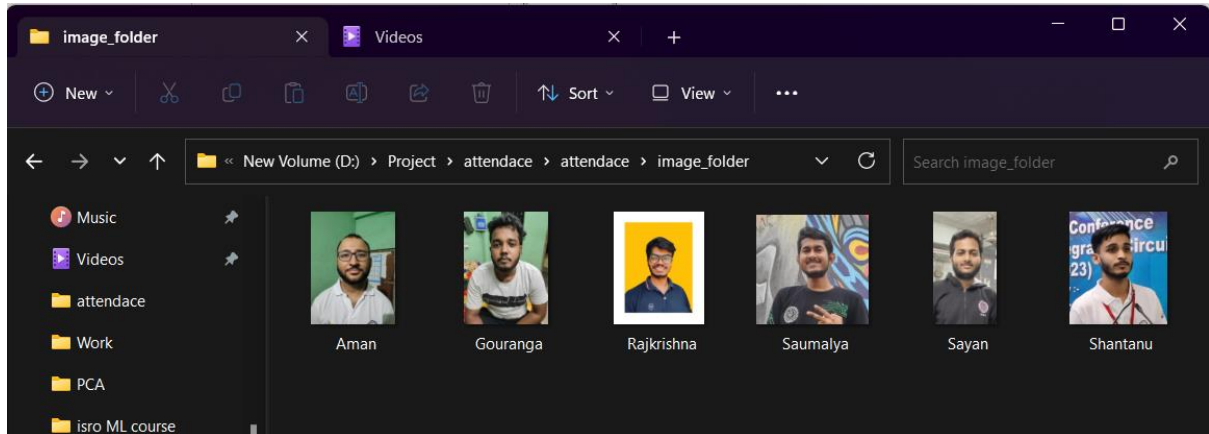


Fig 12 : Image Database

So, now we run the code & stand in front of the ESP32 Camera by showing the face directly to Camera. And the below pictures showing our face scanning results.

Whenever the image of our friend is brought in front of the camera, the face is identified and attendance is recorded.

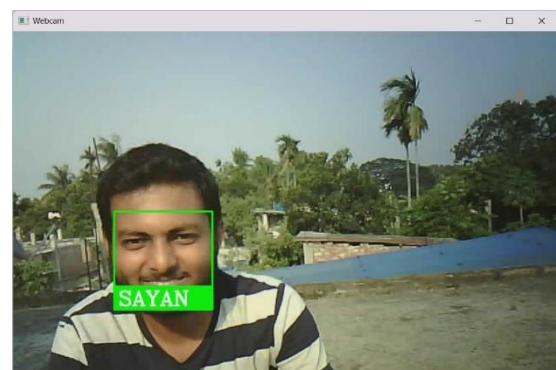
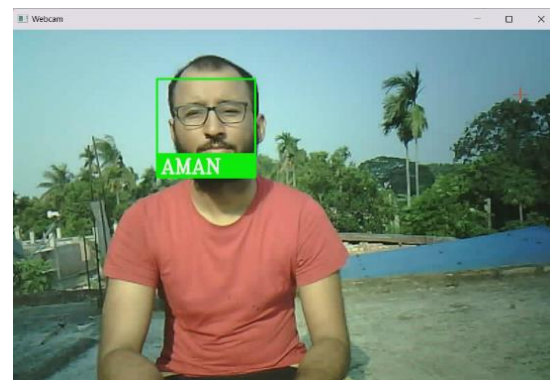






Fig 13 : Realtime Face Detection

Here our project is detecting all the faces which we have already put in our database and showing the name in real-time.

Now press 'q' to close.

Now open the Attendance.csv file, it is in the current working directory. Here we will get all the information of people who were detected and at what time.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	GOURANGA	15:53:00																				
2	GOURANGA	15:53:01																				
3	GOURANGA	15:53:01																				
4	GOURANGA	15:53:02																				
5	AMAN	15:53:08																				
6	AMAN	15:53:08																				
7	AMAN	15:53:08																				
8	AMAN	15:53:08																				
9	AMAN	15:53:08																				
10	SHANTANU	15:53:17																				
11	SHANTANU	15:53:17																				
12	SHANTANU	15:53:17																				
13	SHANTANU	15:53:17																				
14	SHANTANU	15:53:17																				
15	SHANTANU	15:53:17																				
16	SHANTANU	15:53:17																				
17	SHANTANU	15:53:17																				
18	SHANTANU	15:53:17																				
19	SAYAN	15:53:24																				
20	SAYAN	15:53:24																				
21	SAYAN	15:53:24																				
22	SAYAN	15:53:24																				
23	SAYAN	15:53:24																				
24	SAYAN	15:53:24																				
25	SAYAN	15:53:24																				
26	SAYAN	15:53:24																				
27	SAYAN	15:53:24																				
28	SAYAN	15:53:24																				
29	RAJKRISHNA	15:53:31																				
30	RAJKRISHNA	15:53:31																				
31	RAJKRISHNA	15:53:31																				

Fig 14 : Attendance Database

For the accuracy part,

It can detect faces up to 1 metre distance, the lighting condition should be good and the face should be clearly visible.

If the above condition does not match the project may generate errors (e.g., name of another person may arise, or it will not detect the faces).

# APPLICATION

**Educational Institutions:** Implementing the ESP32 camera with face detection in schools, colleges, or universities can automate attendance tracking for students and staff. It eliminates the need for manual roll calls, reduces administrative workload, and provides accurate attendance records.<sup>[2]</sup>

**Corporate Offices:** The ESP32 camera with face detection can be used in office environments to track employee attendance. It simplifies the attendance marking process, prevents buddy punching (where one employee clocks in for another), and ensures accurate timekeeping for payroll purposes.<sup>[3]</sup>

**Events and Conferences:** Use the ESP32 camera with face detection to manage attendance at events, conferences, or seminars. Attendees' faces can be captured and matched against a registration database, facilitating efficient heck-ins and enhancing event security.<sup>[2]</sup>

**Manufacturing Facilities:** Implement the ESP32 camera-based attendance system in manufacturing plants to monitor employee attendance and work hours. It helps manage shifts, track productivity, and streamline payroll processes.

**Membership-based Organizations:** Use the ESP32 camera attendance system in membership-based organizations such as clubs, libraries, or community centers. It ensures that only authorized members can access the facilities and helps manage membership-related activities.<sup>[2]</sup>

These are just a few examples of how an attendance system using the ESP32 camera and face detection can be applied. The specific application and customization will depend on the requirements and context of each setting.<sup>[3]</sup>

# FUTURE SCOPE

The face recognition system using ESP32-CAM, OpenCV, and Python has promising potential for further development and enhancement. Here are some future scopes to consider:

**Real-Time Tracking:** Extend the system to incorporate real-time face tracking capabilities. This would enable the system to track and recognize faces in a video stream or in dynamic environments, such as surveillance applications. <sup>[1]</sup>

**Emotion Recognition:** Integrate emotion recognition capabilities into the system to detect and analyze human emotions from facial expressions. This can have applications in fields such as market research, healthcare, and human-computer interaction. <sup>[3]</sup>

**Privacy Enhancements:** Address privacy concerns by implementing privacy-preserving techniques, such as face anonymization or secure storage and transmission of facial data. <sup>[2]</sup>

**Integration with IoT and Cloud Services:** This can enable remote monitoring and control, centralized management, data analysis, and integration with other smart systems. Cloud-based solutions can provide scalability and flexibility for large-scale face recognition deployments.

**Application-Specific Customization:** Customize the system for specific applications, such as attendance management systems, access control systems, or personalized user experiences in smart devices. This would involve tailoring the system's functionalities and user interfaces to meet the specific requirements of different use cases. <sup>[2]</sup>

By exploring these future scopes, the face recognition system using ESP32-CAM, OpenCV, and Python can be further improved, expanded, and applied to various domains, contributing to advancements in security, automation, and user-centric technologies.

# CONCLUSION

The developed face recognition system has practical implications in various domains. It can enhance security systems by accurately identifying individuals and enabling access control. It can also be applied in surveillance scenarios, enabling real-time monitoring and identification of people. Additionally, the system can provide personalized user experiences by recognizing individuals and customizing interactions accordingly. <sup>[2]</sup>

By combining the capabilities of the ESP32-CAM module, OpenCV library, and Python programming language, this project contributes to the field of face recognition technology. The results obtained from this project can serve as a foundation for further research and development, paving the way for advancements in security, automation, and user-centric technologies.

## REFERENCES

1. E. S. Nandhini, P. Madhumitha, and S. Ananthi, "ESP32-CAM Based Attendance System Using Face Recognition," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 0916-0920. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9072681>
2. S. Hanmandlu, V. Jain, and C. C. Fung, "ESP32-CAM Based Automatic Attendance System using Facial Recognition," 2020 International Conference on Artificial Intelligence and Sustainable Technologies (ICAISTech), Gwalior, India, 2020, pp. 1-6. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9244504>
3. Face Recognition Based Attendance System using ESP32 CAM  
Attendance System using ESP32-CAM | Capture & Store Data in Excel file  
Link : [Face Recognition Based Attendance System using ESP32 CAM \(how2electronics.com\)](https://how2electronics.com)