



# **SMART ANTI-THEFT SYSTEM**

## **A PROJECT REPORT**

**Submitted by**

**S. VAISHNAVI (211521205172)**

**E. ROHINI (211521205124)**

**in partial fulfilment for the award of the degree  
of**

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

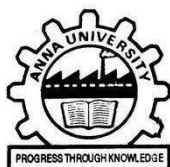
**PANIMALAR INSTITUTE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**NOV-DEC 2023**

# PANIMALAR INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025



## BONAFIDE CERTIFICATE

Certified that this project report titled “**SMART ANTI-THEFT SYSTEM**” is the bonafide work of “**S.VAISHNAVI(211521205172),E.ROHINI(211521205124)**” who carried out the project work under my supervision.

### SIGNATURE

**Dr.S. Suma Christal Mary M.E.,Ph.D.,**  
PROFESSOR & HEAD OF THE DEPARTMENT,  
Department of InformationTechnology  
Panimalar Institute of Technology ,poonamallee,  
Chennai 600123.

### SIGNATURE

**Ms.R.SUSEENDRA M.Tech,**  
ASSISTANT PROFESSOR,  
Department of informationTechnology  
Panimalar Institute of Technology,  
poonamallee, Chennai 600123.

---

Certified that the candidates were examined in the university project viva-voice Examination held on \_\_\_\_\_at Panimalar Institute of Technology, Chennai 600123.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We seek the blessing from the **Founder** of our institution **Dr. A. JEPPIAAR, M.A., Ph.D.**, for having been a role model who has been our source of inspiration behind our success in education in his premier institution.

Our sincere thanks to the Honorable **Chairman** of our prestigious institution **Mrs. REMIBAI JEPPIAAR** for her sincere endeavor in educating us in her premier institution.

We would like to express our deep gratitude to our beloved **Secretary and Correspondent Dr. P. CHINNADURAI, M.A., Ph.D.**, for his kind words and enthusiastic motivation which inspired us a lot in completing this project.

We also express our sincere thanks and gratitude to our dynamic **Directors Mrs. C. VIJAYARAJESHWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. S. SARANYA SREE SAKTHI KUMAR, MBA., Ph.D.**, for providing us with necessary facilities for completion of this project.

We also express our appreciation and gratefulness to our respected **Principal Dr. T. JAYANTHY, M.E., Ph.D.**, who helped us in the completion of the project. We wish to convey our thanks and gratitude to our **Head of the Department, Dr.S.Suma Christal Mary, M.E., Ph.D.**, for her full support by providing ample time to complete our project. Special thanks to our Project Guide **Mr. Vinston Raja, M.Tech., Associate professor** for their expert advice, valuable information and guidance throughout the completion of the project.

Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

# CHAPTER 1

## **1.ABSTRACT**

The project aims to design a framework for providing a house owner/member with the immediate notification of an ongoing theft or unauthorized access to their premises. For this purpose, a rigorous analysis of existing systems was undertaken to identify research gaps. The problems found with existing systems were that they can only identify the intruder after the theft, or cannot distinguish between human and non-human objects. Wireless Sensors Networks (WSNs) combined with the use of Internet of Things (IoT) are expanding smart home concepts and solutions, and their applications. This project proposes a novel IOT based smart home anti-theft system that can detect an intruder. The fundamental idea is to design a cost-effective and efficient system for an individual to be able to detect any kind of theft in real-time and provide instant notification of the theft to the house owner. The system also promises to implement home security with large video data handling in real-time.

Key Words: WSN, IOT, Security, Anti-Theft, Smart home.

The Anti-Theft System using ESP32-CAM and Arduino Uno leverages the capabilities of these versatile microcontroller platforms to create a robust security solution for homes. The project employs the ESP32-CAM's camera module and Wi-Fi connectivity for image capture and remote monitoring, while the Arduino Uno coordinates sensor input and triggers anti-theft measures. The system is designed to detect motion using a PIR motion sensor, activating an alarm and capturing images or streaming video when potential threats are identified. Optionally, the Arduino Uno can control external devices through a relay, enhancing the deterrent effect. The communication between the ESP32-CAM and Arduino Uno is facilitated by serial communication, allowing seamless integration of their respective functionalities. The project offers flexibility, scalability, and the potential for remote monitoring, making it an effective and customizable solution for enhancing home security.

## **TABLE OF CONTENTS**

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>1</b>	<b>ABSTRACT</b>	<b>5</b>
<b>2</b>	<b>INTRODUCTION</b>	
	2.1 OVERVIEW	6
	2.2 PROBLEM DEFINITION	9
	2.3 SCOPE	10
<b>3</b>	<b>COMPONENTS</b>	
	3.1 ESP 32 CAMERA	13
	3.2 ARDUINO UNO	15
<b>4</b>	<b>SYSTEM ANALYSIS</b>	
	4.1 EXISTING SYSTEM	17
	4.2 PROPOSED SYSTEM	19
<b>5</b>	<b>SYSTEM SPECIFICATION</b>	
	5.1 HARDWARE SPECIFICATION	21
	5.2 SOFTWARE SPECIFICATION	23
<b>6</b>	<b>BLOCK DIAGRAM</b>	
	6.1 CIRCUIT DIAGRAM	25
<b>7</b>	<b>WORKING MODEL</b>	
	7.1 CONNECTION OF ESP32 CAMERA AND ARDUINO UNO	27 29

<b>8</b>	<b>SOFTWARE IMPLEMENTATION</b>	
	8.1 ARDUINO IDE SETUP	31
	8.2 ESP CAM SETUP	43
	8.3 EMAIL SETUP	53
<b>9</b>	<b>SOFTWARE CODE</b>	
	9.1 CODE EXPLANATION	55
	9.2 OUTPUT	66
<b>10</b>	<b>APPLICATION</b>	68
<b>11</b>	<b>FUTURE ENHANCEMENT</b>	70
<b>12</b>	<b>CONCLUSION</b>	73
<b>13</b>	<b>REFERENCES</b>	75

# CHAPTER 2



## INTRODUCTION

### 2.1 OVERVIEW

The Smart Anti-theft project aims to design a framework for providing a house owner/member with the immediate notification of an ongoing theft or unauthorized access to their premises. Wireless Sensors Networks (WSNs) combined with the use of Internet of Things (IoT) are expanding smart home concepts and solutions, and their applications. This project proposes a novel IOT based smart home anti-theft system that can detect an intruder. A smart home designed and developed on an integrated framework of sensors, cameras, and customized hardware to analyze unauthorized access. The system operates at two different levels: through a hardware interface and through a software interface.

In this era of IoT, we are developing many security IoT projects which requires camera module. ESP32 camera module is a cost effective ESP32 chip based wifi camera module. In this project report we will see ESP32 Cam code upload using Arduino Uno.

The ESP32-CAM is certainly a perfect example of just how much electronics you can pack onto a tiny circuit board for an incredibly low price. The only thing better than getting a 2 MP camera along with a 32-bit microcontroller with integrated WiFi and Bluetooth for about 10 bucks is that it also comes with a great example sketch that essentially turns it into a surveillance camera with face-detection capabilities. A pretty incredible board, and it also has a MicroSD card.

Arduino Uno is a popular microcontroller board that is commonly used for various electronics projects. While the Arduino Uno itself doesn't have the computational power to perform complex face detection tasks, it can be used in conjunction with external components and modules to create a basic face detection system. Here's a general overview of the functions and components use for face detection with Arduino Uno: Image Capture, Image Processing, Communication, Face Detection Algorithm, Arduino Code, Power Supply, Display, Storage.

## **2.2 PROBLEM DEFINITION**

This problem definition provides a comprehensive overview of the goals, objectives, and constraints for developing a Smart Anti-Theft System. With the increasing number of home burglaries and thefts, there is a growing need for innovative security systems that can provide enhanced protection for homes. Traditional security measures may not always be sufficient, and there is a demand for smarter, technologically advanced solutions. Designing and developing a Smart Anti-Theft System for homes that utilizes cutting-edge technology to detect and prevent unauthorized access, providing homeowners with a reliable and efficient security solution.

Importantly, the project focuses on affordability and accessibility, ensuring that it remains a cost-effective solution, allowing a diverse range of users, including small businesses and individuals, to harness its capabilities. The system's versatility and customization options are a key aspect, designed to cater to a wide range of use cases while providing the ability for integration with other applications. Moreover, the project emphasizes the development of a user-friendly interface, both web-based and mobile applications.

Developing a system that is capable of real-time monitoring and detection of potential threats. Creating a user-friendly interface for homeowners to easily control and customize the security settings. Implementation of advanced sensors and camera to accurately identify unauthorized access or suspicious activities. Ensuring seamless integration with existing home automation systems for enhanced convenience. Developing a robust alert/notification system to promptly inform homeowners and relevant authorities in case of a security breach. The system should be affordable and within a reasonable budget for homeowners. Ensure that the system is energy-efficient and does not impose a significant burden on the home's power supply. Design the system with privacy considerations in mind, avoiding unnecessary data collection. Increased security for homes, reducing the risk of theft and unauthorized access. Improved peace of mind for homeowners through real-time monitoring and alerts. Integration with existing home automation systems, enhancing overall convenience.

## 2.3 SCOPE

Designing a smart anti-theft system for homes involves a comprehensive project scope to ensure that all aspects of the system are considered and implemented effectively. Intruder Detection implement sensors (e.g., motion detectors, door/window sensors) to identify unauthorized entry. Surveillance that integrate cameras with video analytics to monitor and record suspicious activities. Remote Monitoring enables homeowners to monitor their home security system remotely through a mobile application or web interface. Automation Integration allows integration with smart home devices, such as lights and alarms, to enhance security measures. Alerts and Notifications implement a system that informs homeowners immediately upon detecting a security threat.

To provide a brief description of the smart anti-theft system for homes. Outline the primary objectives and goals of the project. Define the overall architecture of the anti-theft system by specifying the hardware and software components involved and describing the interaction between different system elements. To identify the core functionalities of the system and specify how the system detects and responds to potential security threats. Define user interaction features (e.g., mobile app, web interface).Outline the security mechanisms implemented in the system explaining how the system protects against unauthorized access and tampering. Describe encryption methods for communication and data storage. Specify the types of sensors used for intrusion detection.

Describe the communication protocols used within the system components.Specify how the system communicates with external devices (e.g.smartphones, central monitoring stations). Detail how the system alerts homeowners and authorities in case of a security breach. Define notification methods (e.g., SMS, email, push notifications). Specify how users authenticate themselves to access the system. Outline user roles and permissions within the system. Investigate compatibility with existing smart home platforms. Specify integration points and functionalities with other smart home devices. Detail how the system manages power consumption. Include information about backup power sources in case of outages. Design the user interface for both mobile and web applications. Ensure a user-friendly experience for configuring and monitoring the system. Define a testing strategy for the anti-theft system. Include plans for unit testing, integration testing, and user acceptance testing.

# CHAPTER 3

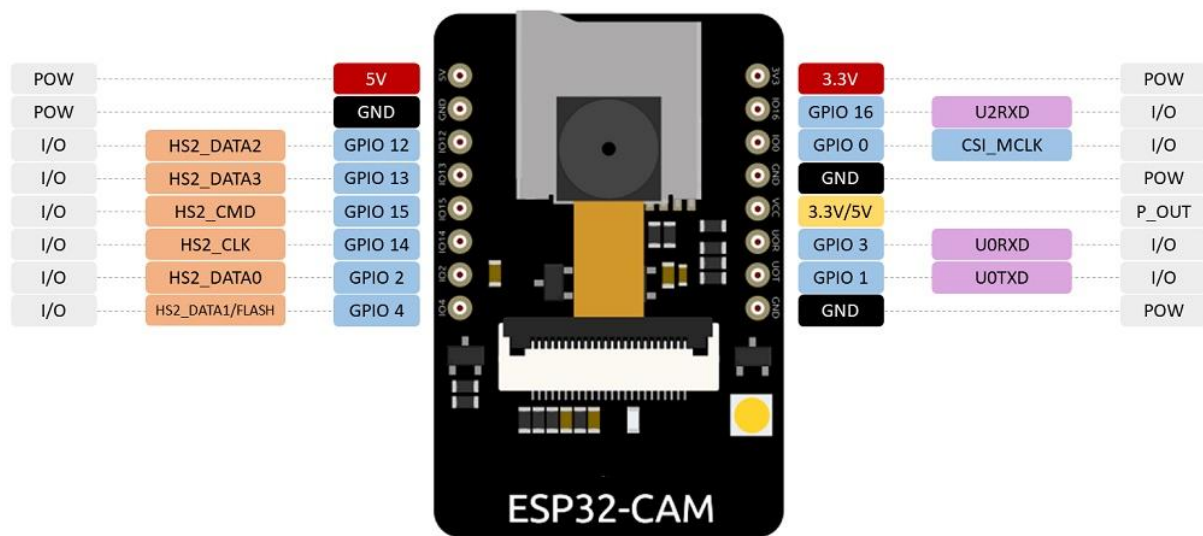
## COMPONENTS

### 3.1 ESP 32 CAMERA

ESP32 CAM is a small camera module built using **ESP32-S** chip. This module is economical and it is equipped with OV2640 camera. This module also has several GPIOs (**General Purpose Input Output**) which helps in interfacing other peripherals. ESP32 cam also equipped with a microSD card slot which is useful to store images and videos. ESP32 module is a powerful yet inexpensive microcontroller from Espressif and A-Thinker with advanced features like Bluetooth, WiFi, and multipurpose GPIO ports.

This amazing little module packs a lot of power and features into a small package. The ESP32-CAM is a full-featured microcontroller that also has an integrated video camera and microSD card socket. It's easy to use, and is perfect for IoT devices requiring a camera with advanced functions like image tracking and recognition. The sample software distributed by Espressif includes a sketch that allows you to build a web-based camera with a sophisticated control panel. After the programming is done in the device it is very easy to use. It is certainly a perfect example of just how much electronics you can pack onto a tiny circuit board for an incredibly low price. It has a 2 MP camera along with a 32-bit microcontroller with integrated WiFi and Bluetooth.

The board has onboard 5volt to 3.3-volt voltage regulator chip, an ESP32 chip with Bluetooth(BLE), wifi, GPIO pins, 2MP camera & a big sized SMD led but unfortunately this cheap board has not any onboard programmer chip like other nodemcu board. So we have to program it using Arduino UNO. It can be widely used in various IoT applications. The camera module resolution can vary, but common models offer resolutions ranging from 2 to 5 megapixels.



**ESP 32 CAM PIN OUT DIAGRAM**



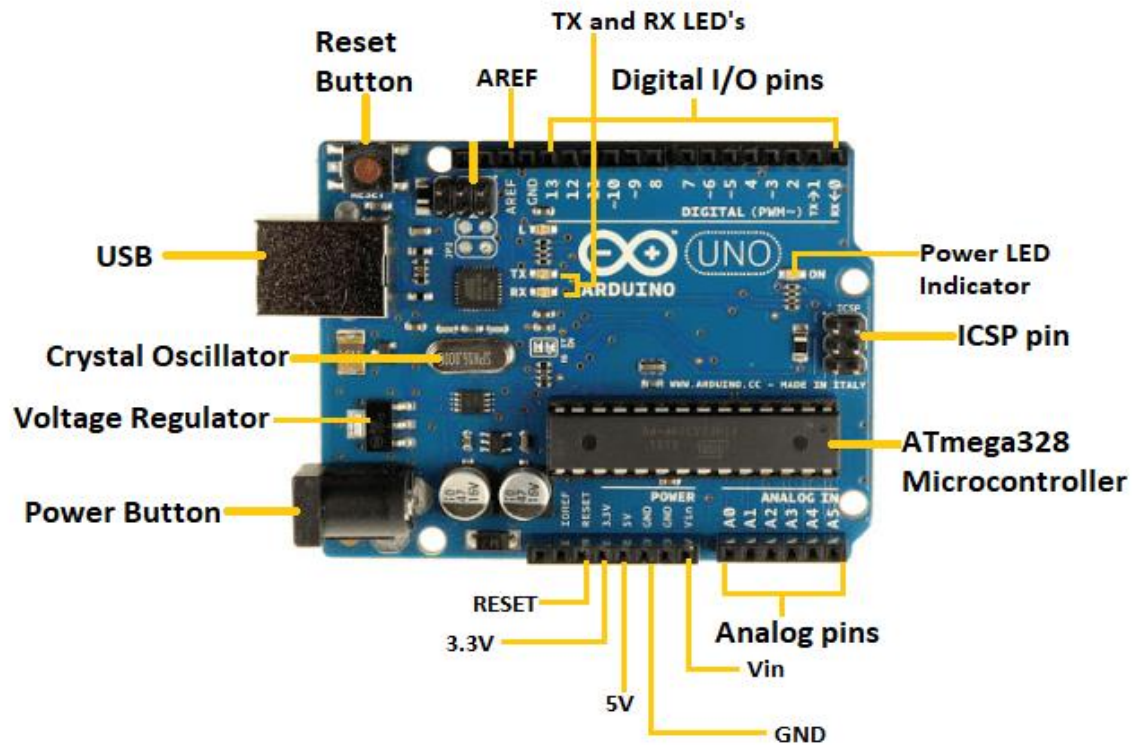
**ESP 32 CAMERA MODULE**

## 3.2 ARDUINO UNO

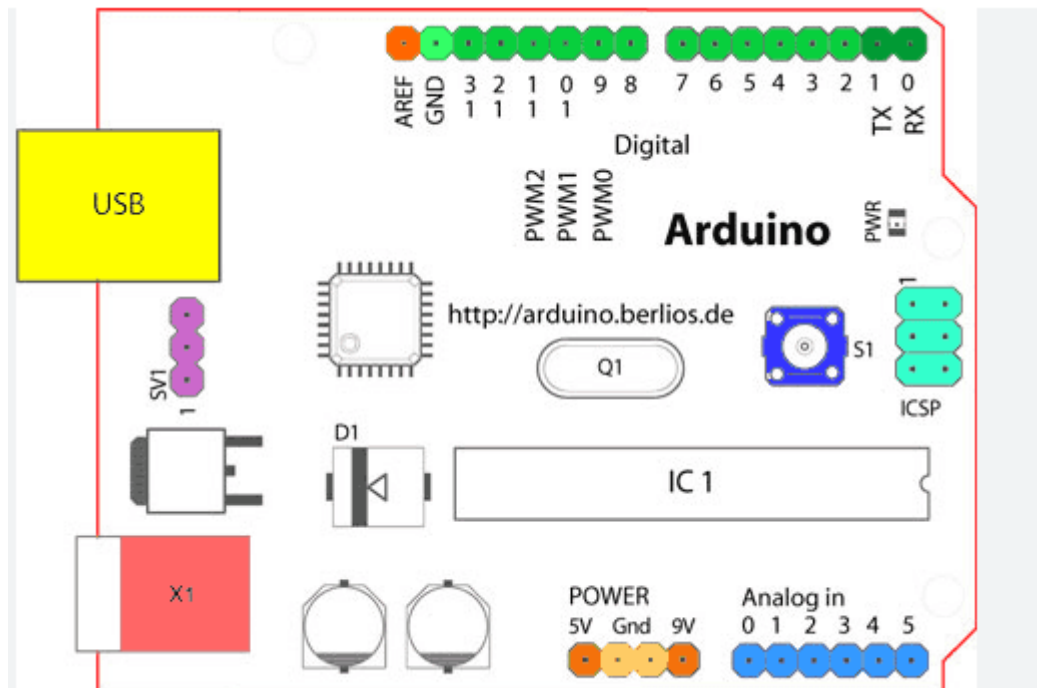
The Arduino Uno is one of the most popular and widely used development boards in the Arduino ecosystem. The Arduino Uno is based on the ATmega328P microcontroller, which is part of the AVR family by Atmel (now a part of Microchip Technology). The ATmega328P has 32 KB of flash memory for program storage, 2 KB of SRAM, and 1 KB of EEPROM. The Clock Speed is at ATmega328P on the Arduino Uno typically runs at 16 MHz. Digital I/O Pins has 14 digital input/output pins (of which 6 can be used as PWM outputs) for interfacing with digital devices like LEDs and switches. Analog Inputs has features of 6 analog input pins, allowing the measurement of analog signals using the built-in analog-to-digital converter (ADC).

The Arduino Uno supports serial communication through UART. It has SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit) communication capabilities, making it compatible with various sensors and devices. The USB Interface board includes a USB interface, which is used for programming the board and serial communication with a computer. Power Supply of Arduino Uno can be powered via USB connection, an external power supply, or a battery. It operates at 5V, but it has both 5V and 3.3V output pins for powering external components. The Programming Language of Arduino Uno is typically programmed using the Arduino IDE, which simplifies the coding process with a user-friendly interface and a vast collection of libraries. Arduino Uno is an open-source hardware platform, and its design files are freely available for anyone to use or modify. Arduino Uno is compatible with a variety of expansion boards called "shields" that can be stacked on top of the board to add additional functionality.

The Community and Documentation of Arduino has a large and active community with forums, tutorials making it easy for users to find support and resources. Arduino Uno is suitable for a wide range of applications, including prototyping, hobbyist projects, educational purposes, and as a platform for learning electronics and programming. The Limitations of Arduino Uno while it is versatile, in terms of processing power, memory, and the number of available I/O pins compared to more advanced microcontrollers. Since face detection involves processing images, you'll likely need an external device to handle the computational load. The code sets up a basic SMTP client on your Arduino with an Ethernet shield. Make sure to replace the placeholder values for the MAC address, SMTP server, and your email address.



**ARDUINO UNO**



**OVERVIEW OF ARDUINO UNO**



# CHAPTER 4

## SYSTEM ANALYSIS

### 4.1 EXISTING SYSTEM

The existing project reports for Smart Anti-theft system for home is done using FTDI. Building a smart anti-theft system for your home using FTDI and an ESP32 camera involves combining hardware and software components. The Hardware Components include ESP32 Camera Module need an development board with a camera module. The camera can capture images or video. FTDI Programmer is used to program the ESP32 it should be compatible with your ESP32 board. Sensors depending on our requirements, you might want to add motion sensors, door/window sensors, or any other relevant sensors. Buzzer/Alarm an audible alarm can be useful to deter potential intruders. Power Supply ensure you have a stable power supply for both the ESP32 and other components.

The Software Components are Arduino IDE, ESP32 Board, FTDI Drivers, ESP32 Camera Libraries. Install the Arduino IDE on your computer. ESP32 Board adds support for the ESP32 board in the Arduino IDE and install the necessary drivers for the FTDI programmer. To make sure we have the latest FTDI drivers installed on your computer. ESP32 Camera Libraries install the required libraries for the ESP32 camera module. The Integration includes connecting Hardware to the ESP32, sensors, and other components according to your circuit diagram. Use the FTDI programmer to upload the code to the ESP32. Test the system by triggering the sensors and ensuring that the anti-theft logic works as expected.

Programming of ESP32 is done by writing code to capture images or video using the ESP32 camera. Implementation of code to handle sensor inputs (motion sensors, door/window sensors, etc.) To activate the alarm (buzzer) when a sensor is triggered. The images or videos are stored on an SD card or upload them to a server for remote viewing. Implementation of logic to determine when the anti-theft system should be armed or disarmed. Remote Monitoring implement a way to remotely monitor your home, perhaps by accessing the camera feed through a web interface.

## 4.2 PROPOSED SYSTEM

The proposed system is made up of both hardware and software elements.

ESP 32 Camera 2. Arduino UNO 3. Power supply. Connection established between the Arduino UNO and ESP 32 Camera sense the data from the sensors are spontaneously processed by the microcontroller and if something is sensed above the limit it sends an alert message to the owner of the house. One thing to note about this ESP 32 CAM module is that it has components on both sides of the printed circuit board. The “top” of the board has the connector for the camera module, as well as the microSD (called “TF”) card socket. A square white LED on the top of the module, this can act as a “flash” for illuminating the subject we are trying to view with the camera. The underside of the circuit board has the ESP32-S module. It also has a connector for an external antenna, as well as an internal antenna that is etched onto the circuit board.

Another key component located underneath the board is the reset switch. Because there are so many components on the bottom of the module you may find it easier not to use a solderless breadboard when experimenting with the ESP32-CAM module. The use of jumpers with female Dupont connectors is recommended. The Serial communication is a process of data transmission sequentially through a communication channel or bus. In serial communication, the data is sent one bit at a time. The serial transmission can be performed using only a single wire and four wires. Serial communication is also known as UART or USART communication.

Setting Up ESP32-CAM and Wiring Components to Arduino Uno and Writing Arduino Code, ESP32-CAM Code Integrate with Home Automation and ensure that you have the necessary libraries installed for both Arduino and ESP32. Adjust pin numbers and connections based on your specific hardware setup. Consider security measures for the system, such as implementing secure communication between devices and securing access to the ESP32-CAM.

# CHAPTER 5

## **SYSTEM SPECIFICATION**

These specifications provide a general overview of the capabilities of the Arduino Uno and ESP32-CAM. When working on a project, it's crucial to consider the specific requirements and features needed for the application. ESP32 Cam module, Arduino Uno, Jumper cables, Arduino IDE. The ESP32-CAM is based upon the ESP32-S module, so it shares the same specifications. The ESP32-CAM includes an OV2640 camera module. The device also supports OV7670 cameras.

### **5.1 HARDWARE SPECIFICATION**

#### **Features:**

#### **ESP32-CAM**

- Ultra-small 802.11b/g/n Wi-Fi + BT/BLE SoC module.
- Low-power dual-core 32-bit CPU for application processors.
- Main frequency up to 240MHz, computing power up to 600 DMIPS.
- Built-in 520 KB SRAM, external 4M PSRAM.
- Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC.
- Support OV2640 and OV7670 cameras, built-in flash.
- Support image WiFi upload, support TF card.
- Material: Copper
- Module model: ESP32-CAM
- Size: 27\*40.5\*4.5( $\pm 0.2$ )mm
- SPI Flash: 32Mbit by default
- RAM: Internal 520KB + external 4M PSRAM

## Arduino Uno:

• <b>Microcontroller:</b>	○ Atmel ATmega328P
• <b>Operating Voltage:</b>	○ 5V
• <b>Input Voltage:</b>	○ 7-12V
• <b>Input Voltage:</b>	○ 6-20V
• <b>Digital I/O Pins:</b>	○ 14 (of which 6 provide PWM output)
• <b>Analog Input Pins:</b>	○ 6
• <b>DC Current per I/O Pin:</b>	○ 20 mA
• <b>DC Current for 3.3V Pin:</b>	○ 50 mA
• <b>Flash Memory:</b>	○ 32 KB (ATmega328P)
• <b>SRAM:</b>	○ 2 KB (ATmega328P)
• <b>EEPROM:</b>	○ 1 KB (ATmega328P)
• <b>Clock Speed:</b>	○ 16 MHz
• <b>USB Connector:</b>	○ Type B
• <b>Communication:</b>	○ UART, I2C, SPI
• <b>Dimensions:</b>	○ 68.6mm x 53.4mm

## Jumper wires:

Jumper wires are a fundamental component for connecting various elements within an electronics project. When selecting jumper wires for use with Arduino Uno and ESP32-CAM, we are considering the following specifications: Wire Gauge, Length, Connector Types, Color Coding, Stranded vs. Solid Core, Insulation Material, Compatibility, Quantity.

## 5.2 SOFTWARE SPECIFICATION

### 1. Programming Language:

- The ESP 32 CAM is typically programmed using the Arduino IDE, Platform IO, or a similar environment. The primary programming language is C/C++.

### 2. Data Transmission:

- If we plan to transmit the data to a remote server or cloud service, you'll need code to establish Wi-Fi connectivity and send data over HTTP, MQTT, or any other relevant protocols. This may involve libraries for Wi-Fi communication.

### 3. ESP32 Board Support:

- Integrate the ESP32 board support into the Arduino IDE. Follow the instructions provided by the official ESP32 Arduino Core repository on GitHub. This usually involves adding the ESP32 board support through the Arduino Board Manager.

### 4. Camera Libraries:

- To work with the camera module on the ESP32-CAM, we'll need specific libraries. The most commonly used library for ESP32-CAM is the [Arduino Camera Web Server library](#). This library provides functions to capture images, stream video, and set up a basic web server to access the camera feed.

### 5.USB-to-UART Driver:

- If we are connecting the ESP32-CAM to your computer for programming or debugging via USB, ensure that you have the necessary USB-to-UART driver installed. The driver depends on the specific USB-to-UART chip on your ESP32-CAM module.

### 6.Camera Web Server Example:

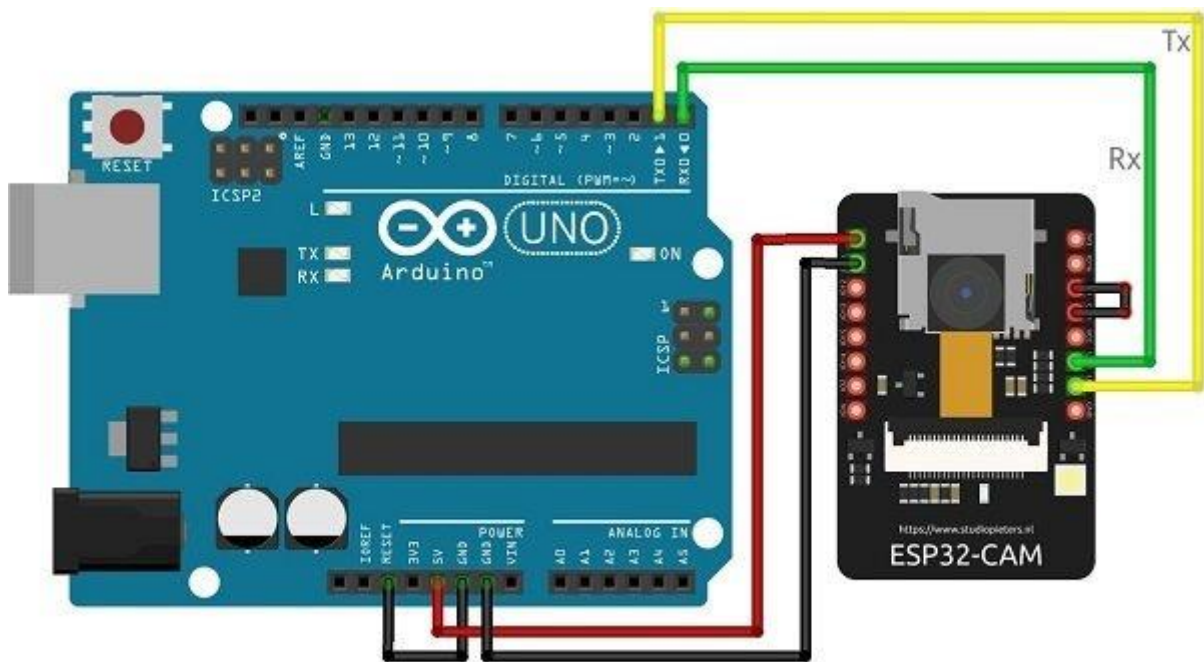
- Many ESP32-CAM projects start with the CameraWebServer example provided by the Arduino Camera Web Server library. This example sets up a basic web server on the ESP32-CAM, allowing you to access the camera feed through a web browser.

# CHAPTER 6



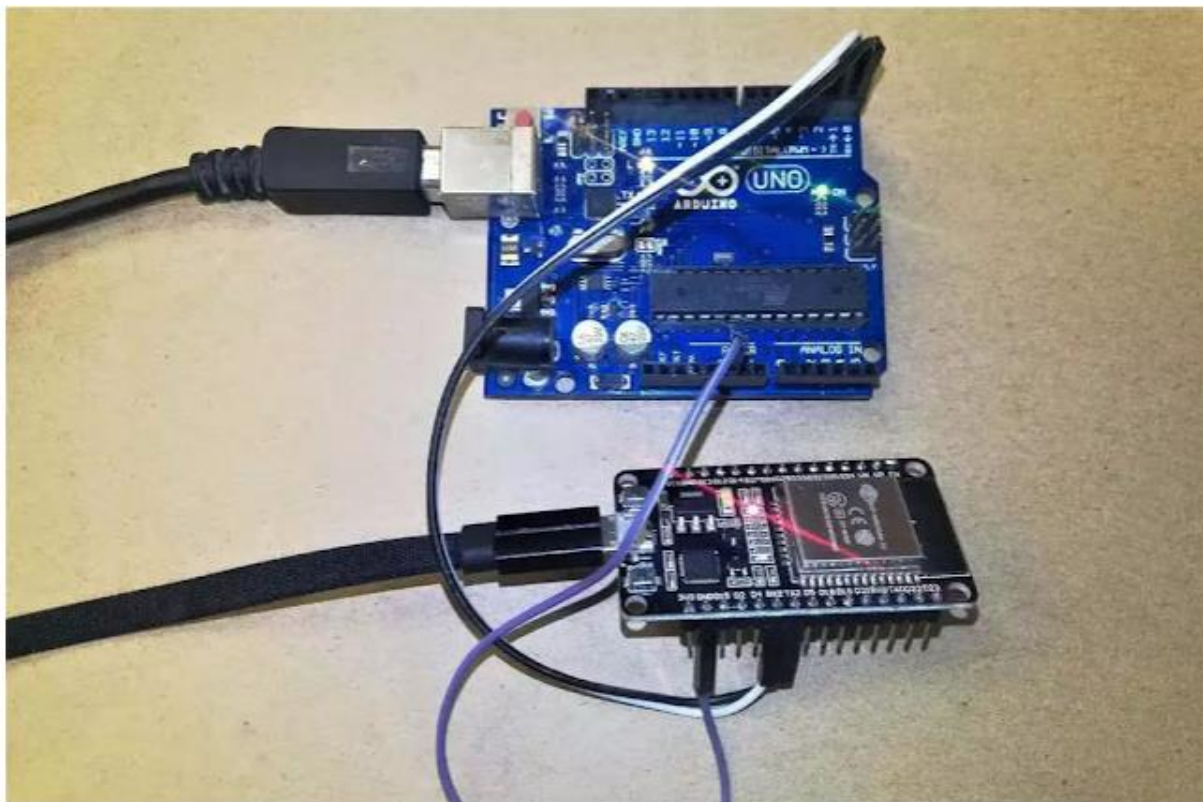
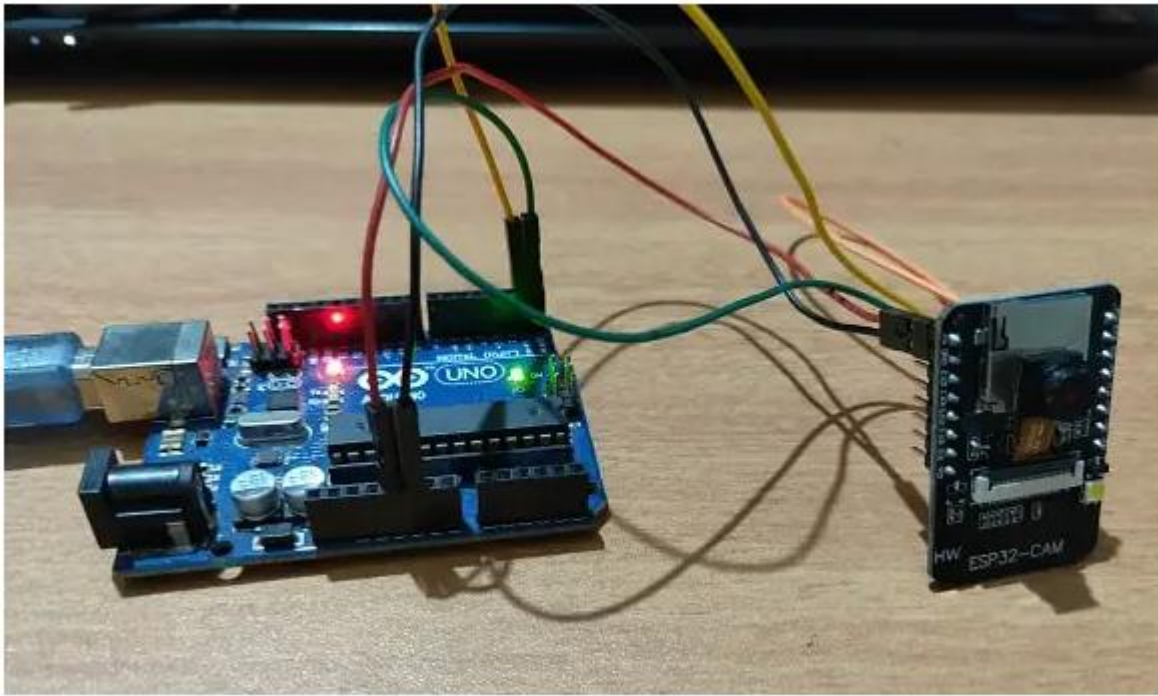
## BLOCK DIAGRAM

### 6.1 CIRCUIT DIAGRAM



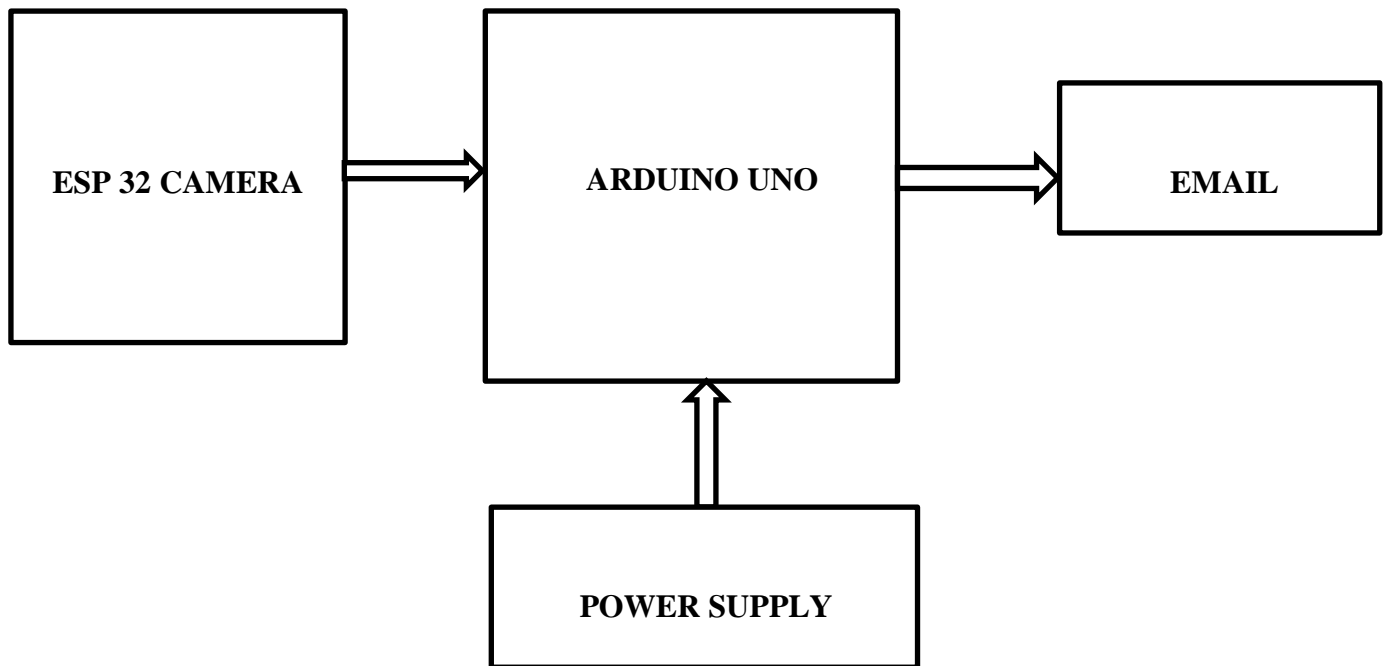
### CIRCUIT DIAGRAM ARDUINO UNO AND ESP 32 CAMERA

After connecting and uploading the code you ESP32 cam is ready to test. Open the serial monitor and reset the ESP32. You can see the IP address on which the cam web server has started. Make sure the cam board DO is connected to gnd. It will enable esp 32 flash mode otherwise you can't program it. after compleat, the programming remove D0 to gnd



**ESP32 CONNECTION WITH ARDUINO UNO**

## BLOCK DIAGRAM



- Connect the Reset Pin of the Arduino with the GND.
- Connect the 100 pin of the ESP32 Cam with the GND Pin, for this you can use a female to female type jumper wire.
- Connect the 5V and GND pins of the ESP32 Cam with the Arduino's 5V and Ground.
- Connect the Receive Pin of the ESP32 Cam with the RX pin of the Arduino.
- Connect the Transmit Pin of the ESP32 Cam with the TX pin of the Arduino.
- The ESP32 Cam interfacing with the Arduino is completed. Now, let's go to the computer screen and install the ESP32 Cam board.

# CHAPTER 7

## WORKING MODEL

### 7.1 CONNECTION OF ESP32 CAMERA AND ARDUINO UNO

The working principle of an anti-theft system using ESP32-CAM and Arduino Uno involves using the Arduino Uno to monitor sensors and control the alarm, while the ESP32-CAM captures images or streams video when a potential threat is detected. Here's a step-by-step breakdown:

#### Components Used:

##### 1. ESP32-CAM:

- Equipped with a camera module (e.g., OV2640).
- Connects to the home Wi-Fi network for communication.
- Captures images or streams video upon command.

##### 2. Arduino Uno:

- Monitors sensors (e.g., PIR motion sensor) for detecting motion.
- Activates an alarm (e.g., buzzer) when motion is detected.
- Optionally, controls external devices (e.g., lights) using a relay.

##### 3. Sensors:

- PIR Motion Sensor: Detects human motion in its range.
- Other sensors (optional): Depending on requirements, you might use additional sensors like door/window contact sensors or vibration sensors.

##### 4. Alarm:

- Buzzer or other alarm devices to alert and deter intruders.

#### Working Steps:

##### 1. Initialization:

- Both the ESP32-CAM and Arduino Uno are powered and initialized.
- ESP32-CAM connects to the home Wi-Fi network
- .

##### 2. Idle State:

- The system is in an idle state, monitoring for motion.

### 3. Motion Detection:

- The Arduino Uno constantly monitors the PIR motion sensor.
- If motion is detected, the Arduino Uno triggers the alarm (activates the buzzer).

### 4. Anti-Theft Actions:

- Upon detecting motion, the Arduino Uno sends a signal to the ESP32-CAM to capture an image or start video streaming.
- The captured image or video stream can be used for remote monitoring or evidence.

### 5. Optional External Device Control:

- If using a relay module, the Arduino Uno can also control external devices (e.g., turning on lights) to simulate an occupied home and deter intruders.

### 6. Alert Notification (Optional):

- You can extend the system to send alerts or notifications to a user's smartphone or a central monitoring system using internet connectivity provided by the ESP32-CAM.

### 7. System Reset:

- After a predefined period or upon user acknowledgment, the system resets, and both the Arduino Uno and ESP32-CAM return to the idle state.

## Communication Between Arduino Uno and ESP32-CAM:

- The Arduino Uno and ESP32-CAM communicate through serial communication using jumper wires.
- The Arduino Uno sends commands to the ESP32-CAM (e.g., to capture an image) through the serial connection.

## Considerations:

- Ensure secure communication between the Arduino Uno and ESP32-CAM, especially if the system is accessible over the internet.
- Implement power management to ensure the reliable and continuous operation of the system. Customize the code to suit specific sensor configurations, alarm preferences, and external device control requirements.

# CHAPTER 8

## 8. SOFTWARE IMPLEMENTATION

### 8.1 ARDUINO IDE SETUP

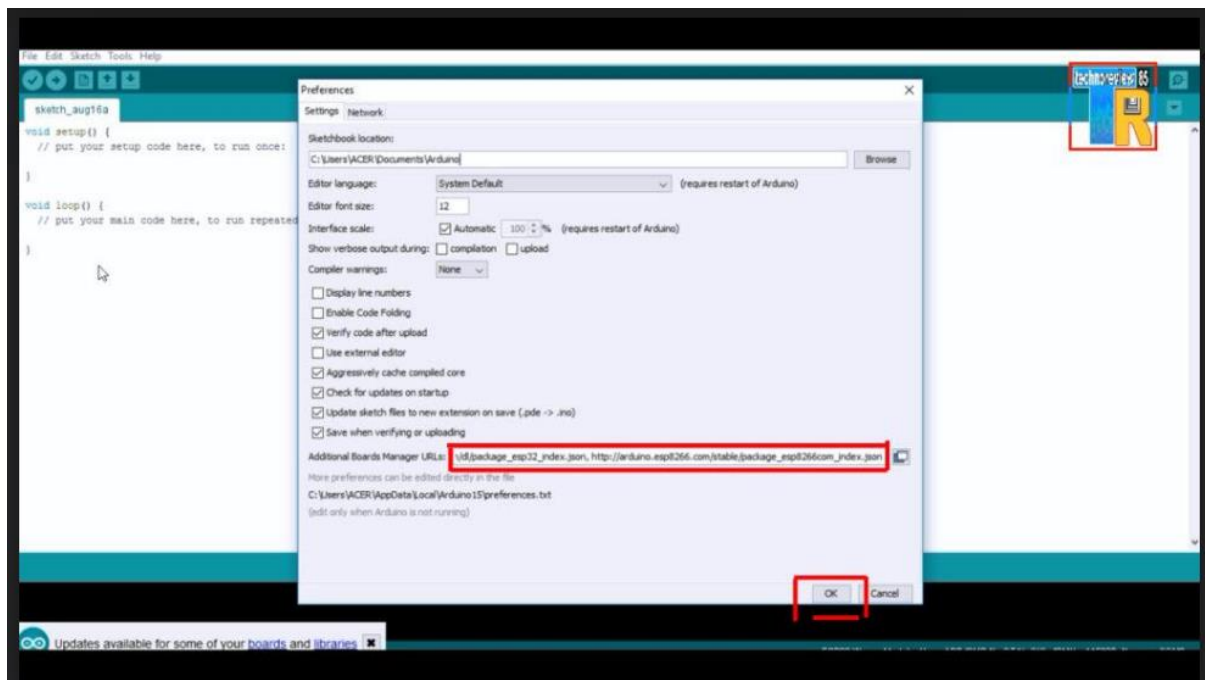
In this , I am going to describe how to program ESP32 cam module using arduino IDE & without FDTI & with USB

The components required are

- 1.ESP32 cam ( I am using AI thinker)
- 2.Arduino UNO board
- 3.Jumper wire
- 4.Arduino IDE software

Now open Arduino IDE on your PC, go to File > Preferences >  
Now paste below link in the board manager URL

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

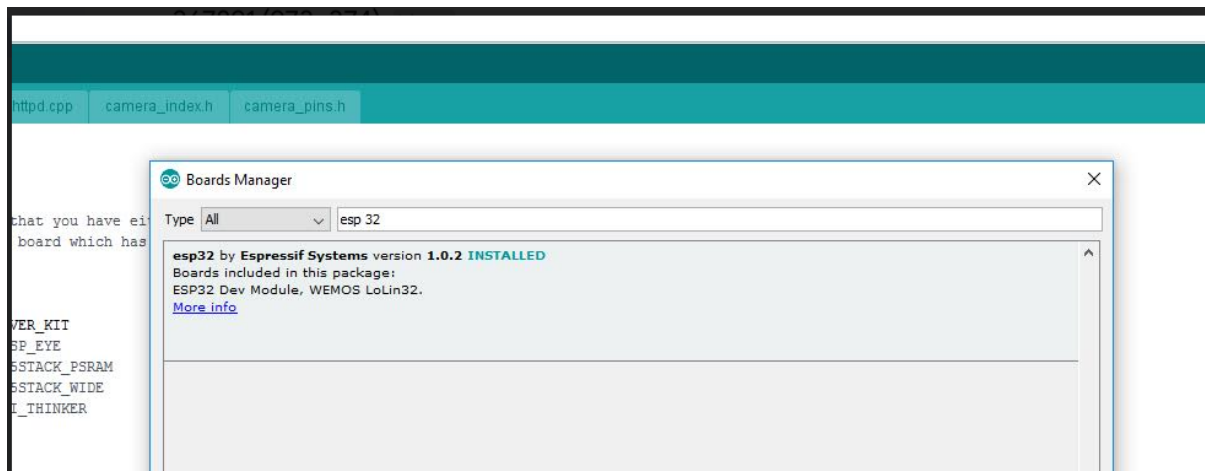


esp32cam programming

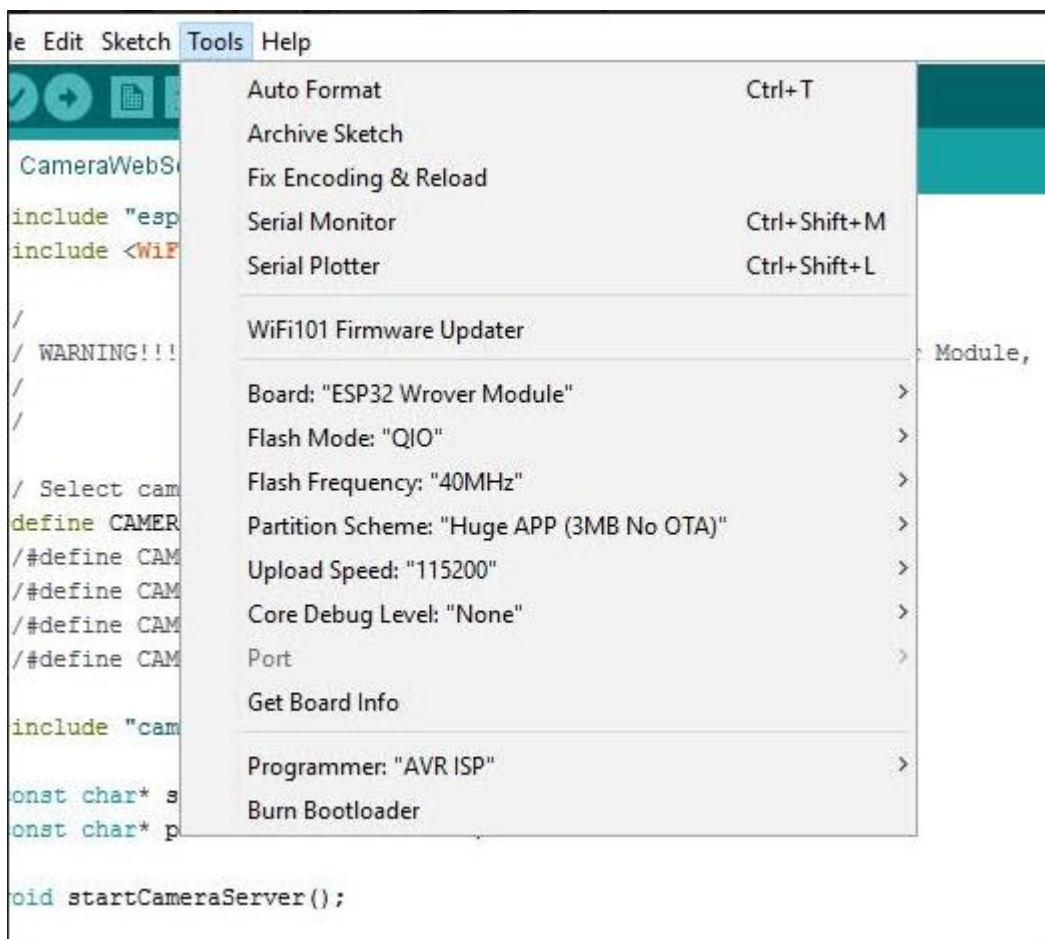
Now go to Tools > Board > Board manager > & search for ESP 32

You can see esp 32 board Download & install the latest version package





After complete, the installation go to Tools > Select port where arduino UNO is connected



Now go to board > select board > ESP32 " Wrover Module"  
Flash Mode > QIO  
Flash Frequency > 40MHZ  
Partition Scheme > Huge App (3mb No OTA)  
Upload speed > 115200  
Programmer > AVR ISP

Now it is ready for uploading sketch

You can test it Go to File > Example > ESP32 > Camera > Camera web server and upload the sample sketch

## **8.2 ESP32 CAM SETUP**

In this all about face recognition with the ESP32-Cam board. I'll show you how to set up a video streaming web server with ESP32- Cam and perform face detection and recognition of the particular person.

### **Configuration of Video Streaming Server:**

Follow the below steps to build a video streaming web server with the ESP32-Cam that you can access on your local network.

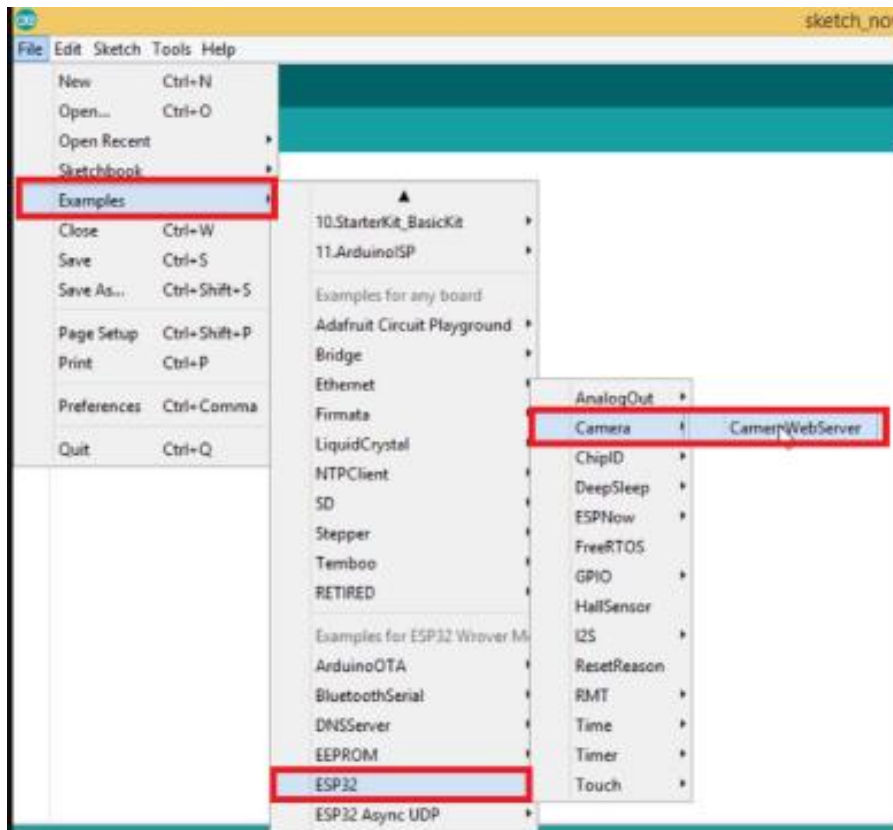
#### **1. Install the ESP32 add-on**

In this example, we use the Arduino IDE to program the ESP32-Cam board. So, you need to have Arduino IDE installed as well as the ESP32 add-on. If you haven't installed the ESP32 add-on in your machine, follow the below tutorials and get it installed.

- Install ESP32 Add-on in ArduinoIDE

#### **2. CameraWebServer Example Code**

In your Arduino IDE, go to File > Examples > ESP32 > Camera and open the CameraWebServer example.



You need to make some modifications before uploading the code. Insert your network credentials in the following variables:

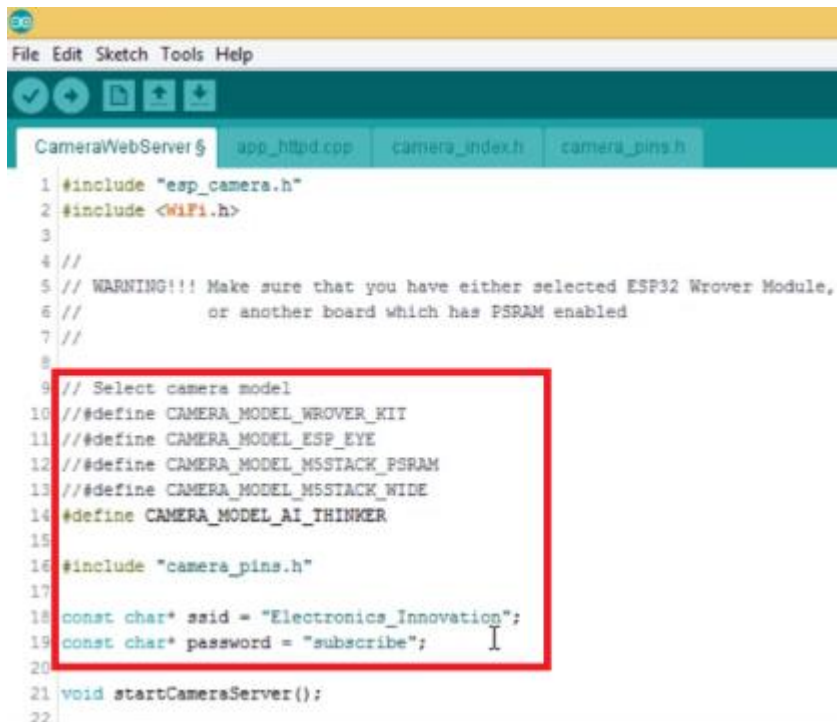
```
const char* ssid = "REPLACE_WITH_YOUR_WIFI_SSID";
```

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

Then, make sure you select the right camera module. In my case, it is AI-THINKER Model.

```
#define CAMERA_MODEL_AI_THINKER
```

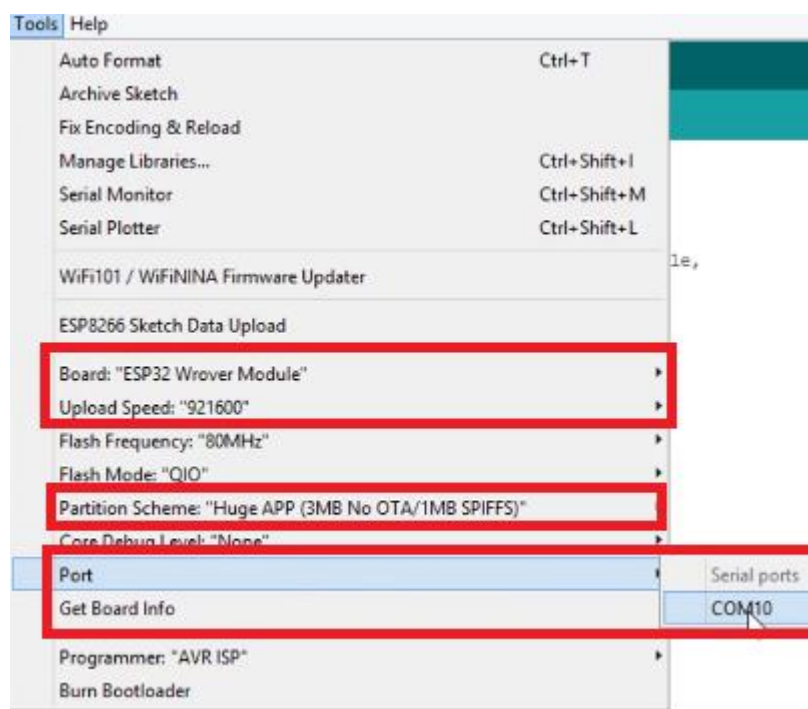
Now, the code is ready to be flashed into your ESP32-Cam module.



```
1 #include "esp_camera.h"
2 #include <WiFi.h>
3
4 //
5 // WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
6 // or another board which has PSRAM enabled
7 //
8
9 // Select camera model
10 #define CAMERA_MODEL_WROVER_KIT
11 #define CAMERA_MODEL_ESP_EYE
12 #define CAMERA_MODEL_MSSTACK_PSRAM
13 #define CAMERA_MODEL_MSSTACK_WIDE
14 #define CAMERA_MODEL_AI_THINKER
15
16 #include "camera_pins.h"
17
18 const char* ssid = "Electronics_Innovation";
19 const char* password = "subscribe";
20
21 void startCameraServer();
22
```

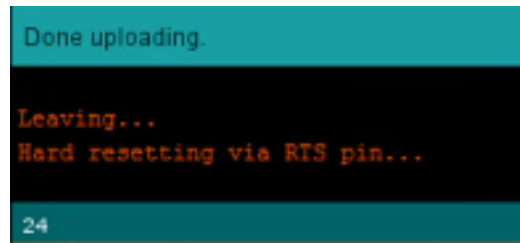
To upload the code, follow the next steps:

1. Go to Tools > Board and select ESP32 Wrover Module
2. Go to Tools > Port and select the COM port the ESP32 is connected to
3. In Tools > Partition Scheme, select “Huge APP (3MB No OTA)”
4. Press the ESP32-CAM on-board RESET button
5. Then, tap on the upload button to upload the code



Uploading done: Hard resetting via RTS pin...

Don't be panic, It's not an error. It is letting us know that the ESP32-Cam is getting reset after the successful uploading of the code. This is the new feature of ArduinoIDE. This feature is available in version 1.8.0 or Above.

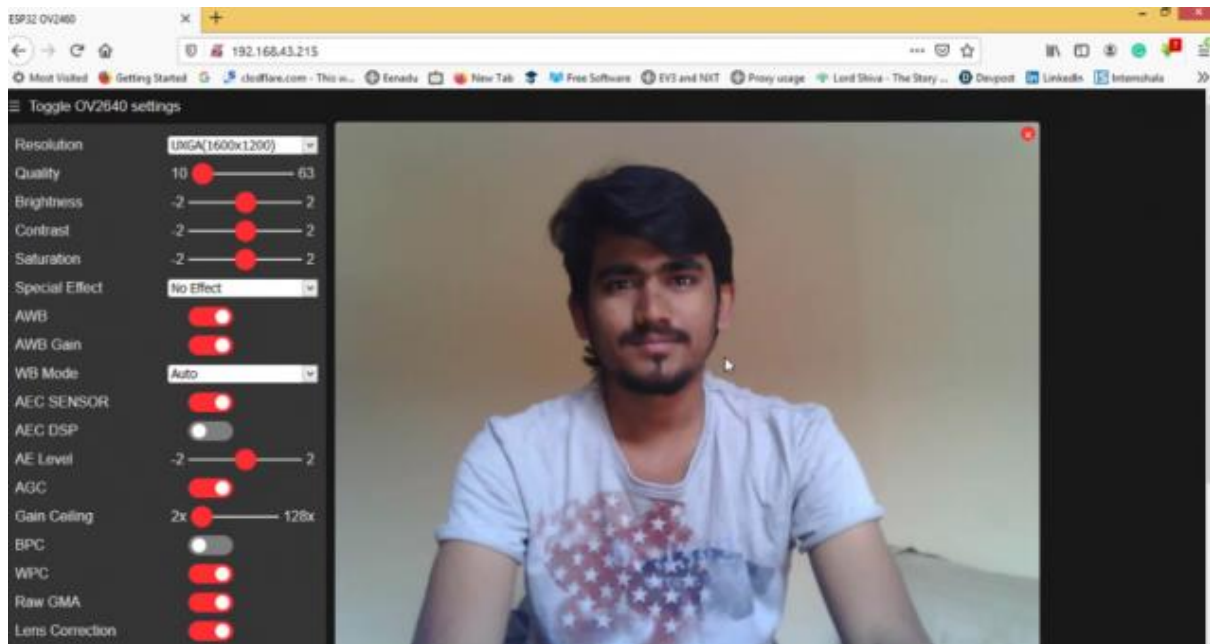


The Status of the WiFi Connection, Camera module and ESP32 IP address of the webserver will print in the Serial Monitor as shown below.

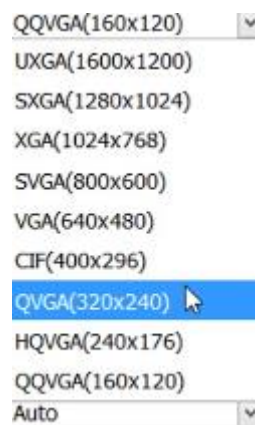


### ***Video Streaming Server:***

Now, you can access your camera streaming server on your local network. Open a browser and type the ESP32-CAM IP address. Press the **Start Streaming** button to start video streaming.

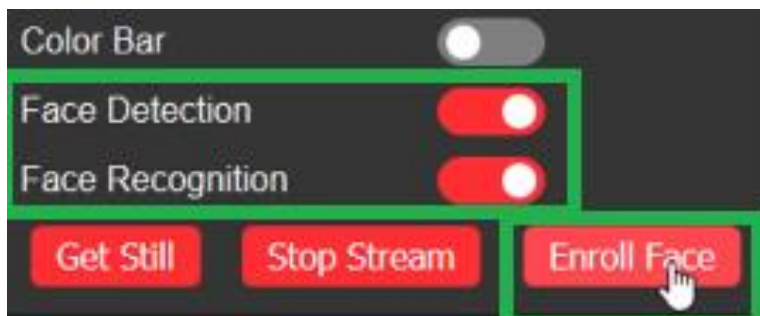


There are also several camera pixel configurations that you can play with to adjust the image settings. For better streaming experience it is recommended to choose “QVGA(320X240)”



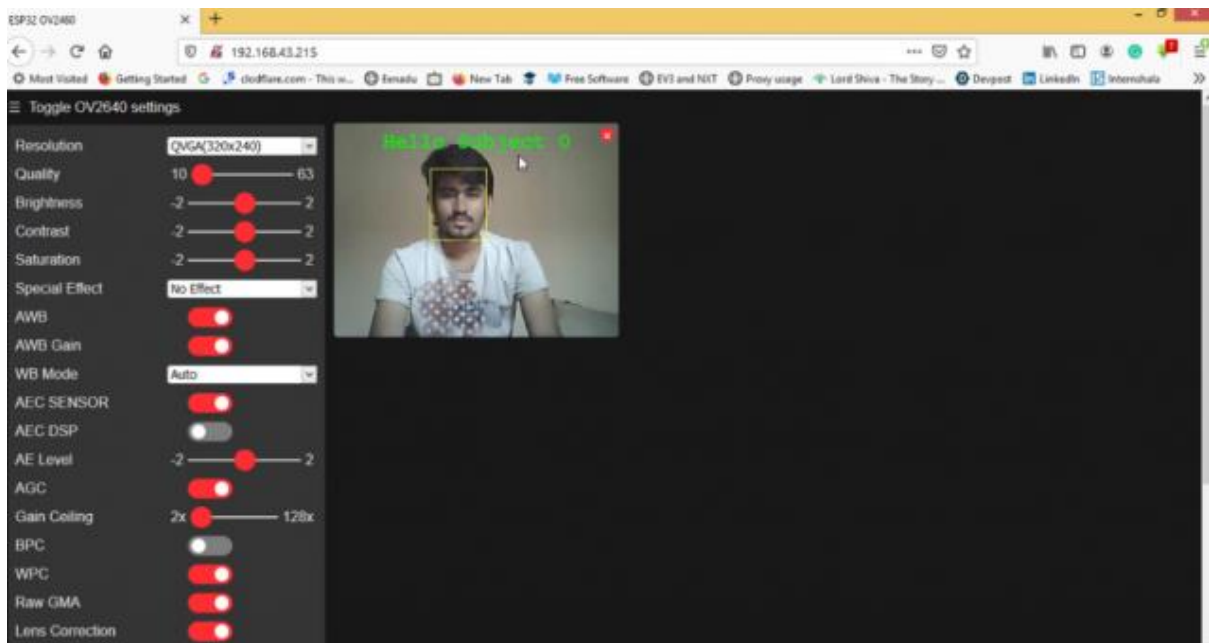
### ***Face detection and recognition:***

Enable the face detection and face recognition options that are present in the bottom left corner, then tap on the Enroll face button.



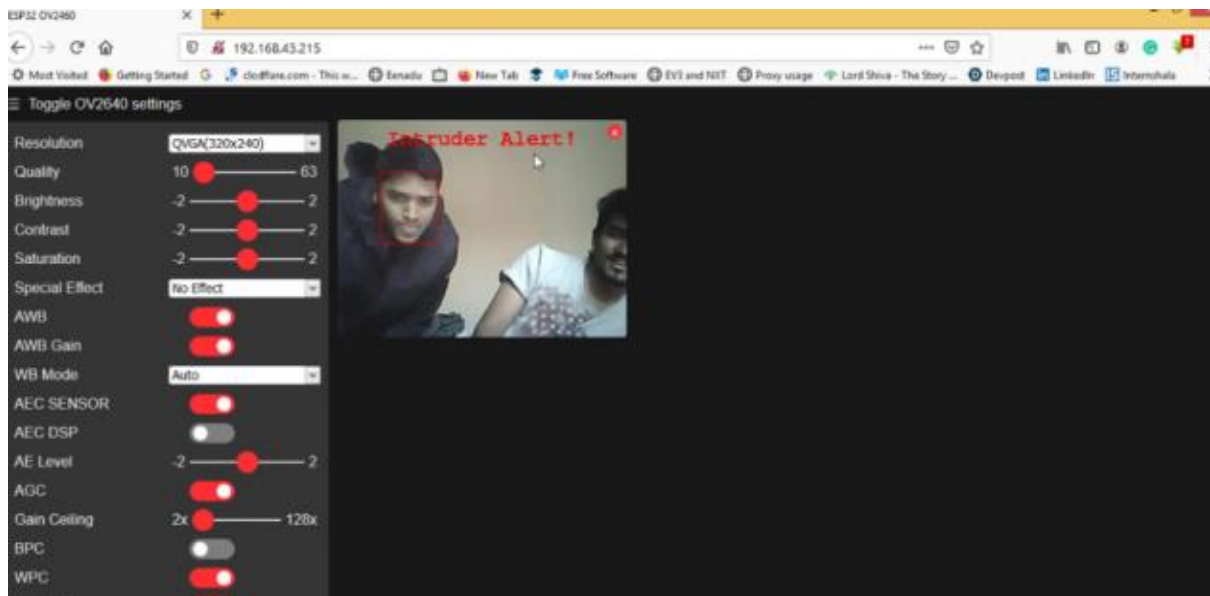
It will make several attempts to save the face. After enrolling a new user, it should detect the face later on (subject 0). And that's it. Now you have your video streaming web server up and running with face detection and recognition with the example from the library.

After complete enrollment of the new face. It should start recognizing the face and will display Hello Subject 0, as shown below. You can change the subject 0 to the particular person named in the code.





If any other persons who didn't enroll, it will recognize them and display Intruder alert.



So, this is how we have initiated an ESP32-Cam web server and Configured for face detection and Face recognition. It's the cheapest module so far which can perform face recognition.

The ESP32-CAM provides an inexpensive way to build more advanced home automation projects that feature video, taking photos, and face recognition.

### 8.3 EMAIL SETUP

#### **Sending Email Alerts using ESP32 via SMTP Server**

Emails are worldwide used as an essential part of digital communication. Emails are mostly used for official communication purposes because it is most convenient, cost-effective, record-keeping, global reach, and environmentally friendly. Emails are a very faster way of communication, only you need a steady internet connection.

In this project, we are going to send emails (Plain text and HTML) using the ESP32 board. The ESP32's built-in Wi-Fi connectivity enables it to communicate with the internet using SMTP servers for sending emails. This project has various



benefits for various applications, such as delivering notifications, alarms, and alerts.

#### Components Required:

For this project, we need very few requirements including Software and Hardware

- 1.ESP32 Microcontroller Module
- 2.Gmail Account
- 3.Arduino IDE

#### **Configuring Google Gmail Account:**

We are using Gmail as a sender Email account; don't worry it's free at all. By using the Gmail SMTP server, we will send and receive mail. SMTP server known as Simple mail transfer Protocol is responsible for sending and receiving mail between the sender and receiver. Different email providers like Gmail, Yahoo, Hotmail, Outlook, etc. have unique SMTP addresses and settings.

Gmail SMTP Server Settings are provided below: other emails provider setting you can check on the Internet

Server: [smtp.gmail.com](https://smtp.gmail.com)

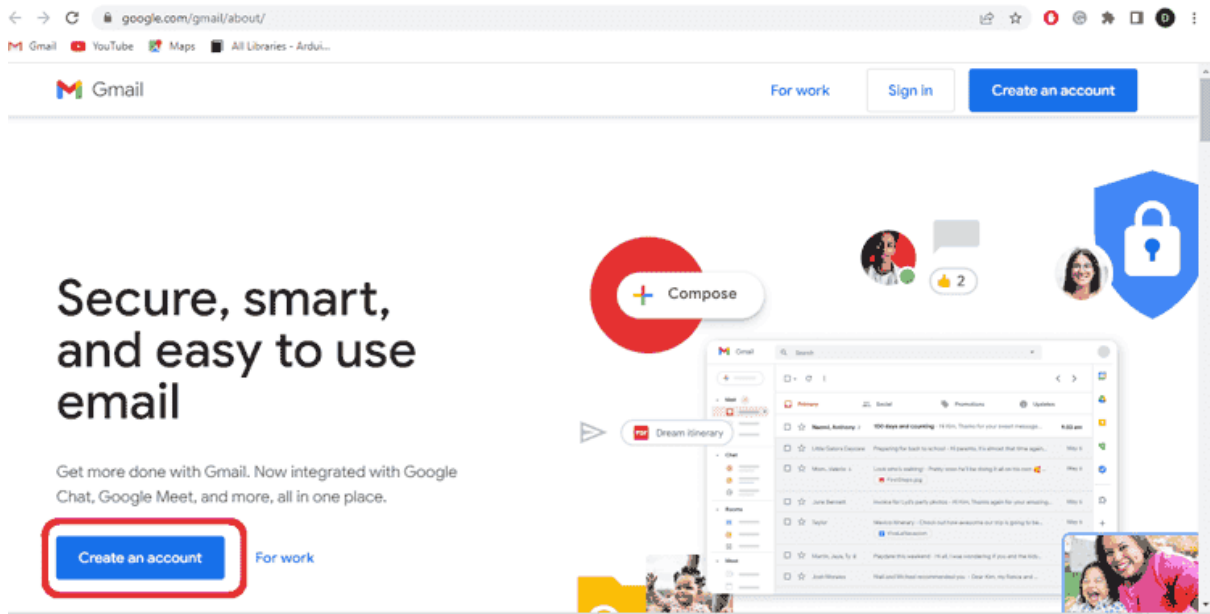
Server port (TLS): 587

Server port (SSL): 465

SMTP TLS/SSL required: yes

Now, to send the emails using esp32 we need a sender mail Account, which we use a Gmail account. You can also use your older Gmail account, but it is not recommended because it will be blocked if something wrong happens during programming. Opening a new Gmail account is very easy, all you have to do is just follow the basic steps:

- 1.Go to the Google Gmail page.
- 2.Click on Create a new account.



Fill in the Required details. All the details are mandatory to fill in as these are required to open a Google Gmail Account.

A screenshot of the Google Account creation page. The URL in the address bar is 'accounts.google.com/signup/v2/webcreateaccount?service=mail&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&flowName=GlifWebSignin...'. The page title is 'Create your Google Account to continue to Gmail'. The form includes fields for 'First name', 'Last name', 'Username' (with a placeholder '@gmail.com'), and 'Password' (with a 'Confirm' field). A note states 'You can use letters, numbers & periods'. There is a checkbox for 'Show password' and a 'Next' button. A 'Sign in instead' link is at the bottom left. On the right, there is a graphic of a blue shield with a person icon and the text 'One account. All of Google working for you.'

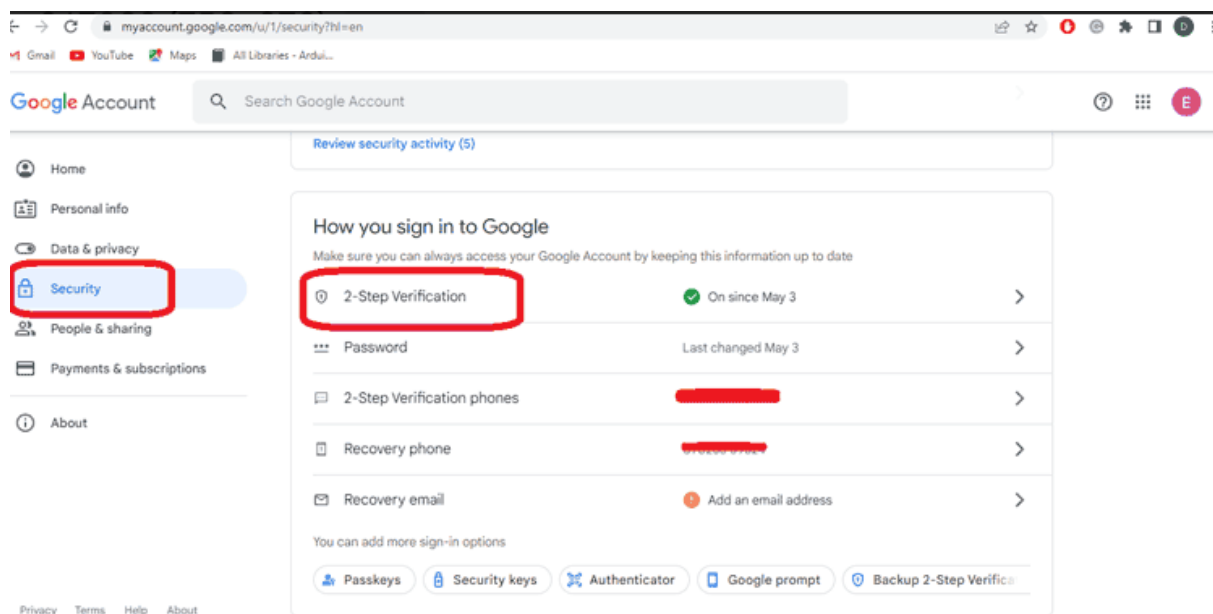
After filling in the details, click 'next'. Accept all terms and conditions and verify the mobile number if it asked for it. Hence your account will be created.

## Create an App Password:

To email with an ESP32, generate a unique app password in your Gmail account for the app or device that doesn't support 2-step verification. An app password serves as a substitute for your regular password to access Gmail on such devices. These passwords can be revoked from the Google Account security settings and are valid only for the specified app or device.

First, the necessary condition to create an app password is to enable 2-step verification on your account.

- 1.Login to your Google account, go to Manage Google account settings
- 2.Go to Security settings, and scroll down till you see “2-step verification

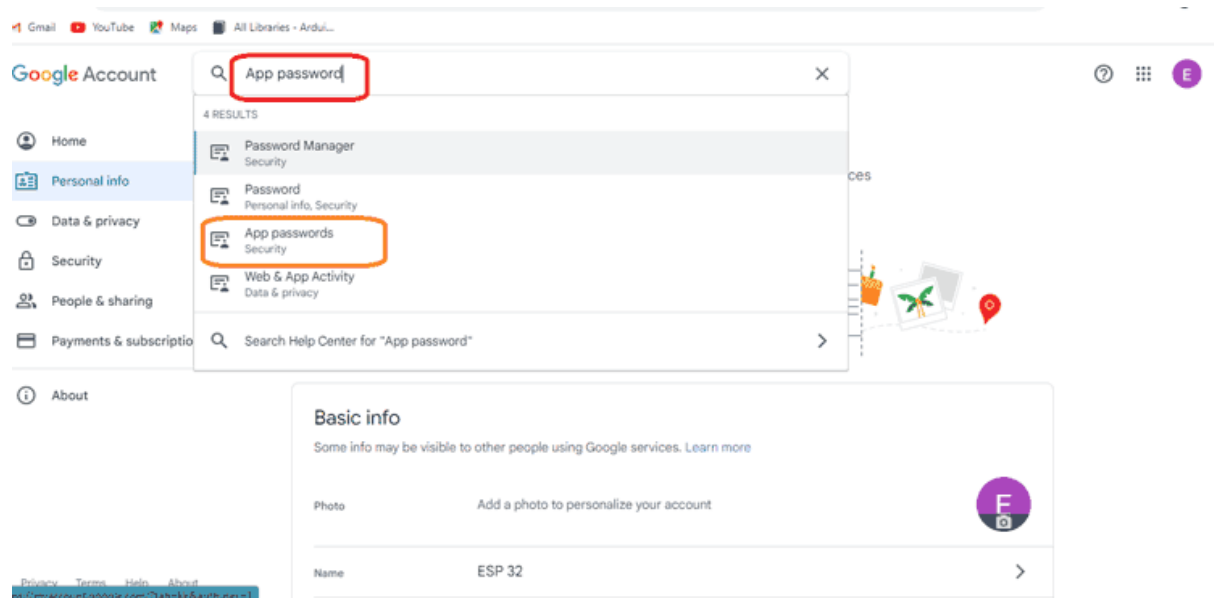


- 3.Enable 2-step verification by adding and verifying mobile number

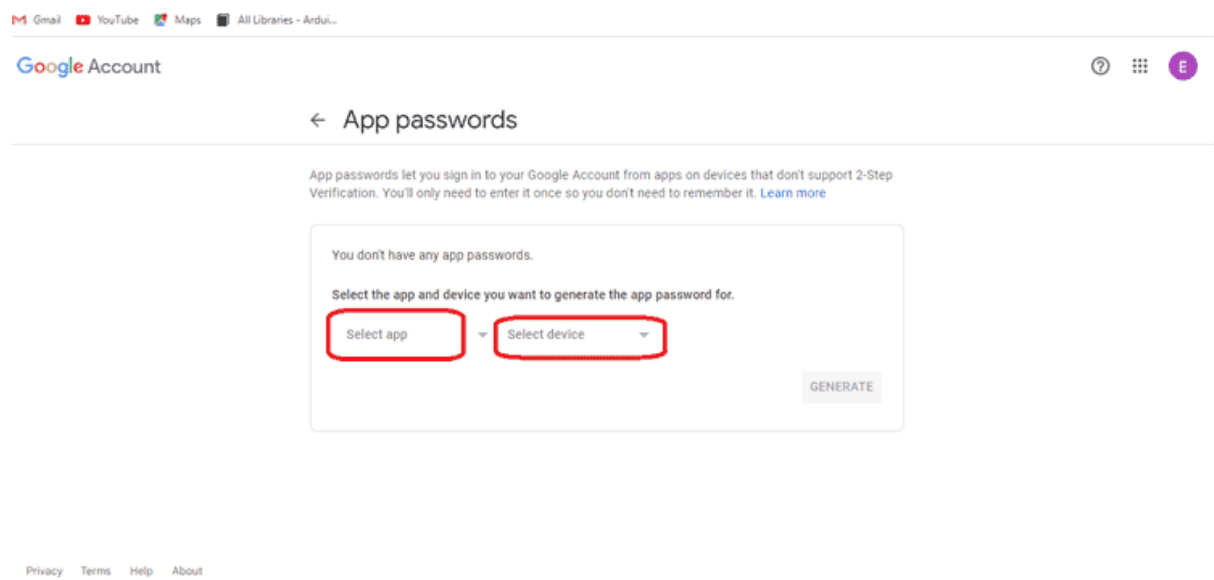
4.After Enabling it, you always prompt to enter a verification code that was sent on your smartphone whenever you are trying to log in to that account on any other device.

## Creating an App Password:

Now, go to the search bar and search “App Password”. Click on the App password option.



Here, you will see a window where you can create an App password for the app you want to use with your Google Account. In selected fields, select the app as mail type and the device as others.



After selecting the fields, give a name when it prompts to the next field and finally click Generate. You will see that an App password would be generated, and save the code for later, which we will use in our programming to provide access of the Gmail account to ESP32.

### Generated app password

Email

Password

Your app password for your device

jfhw ecbe pjml zymi

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.


DONE

Finally, you see that a device is added.

## ← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Name	Created	Last used	
ESP32	1:15 PM	–	

Select the app and device you want to generate the app password for.

Select app

▼

Select device

▼

GENERATE

### Programming ESP32 Using Arduino IDE:

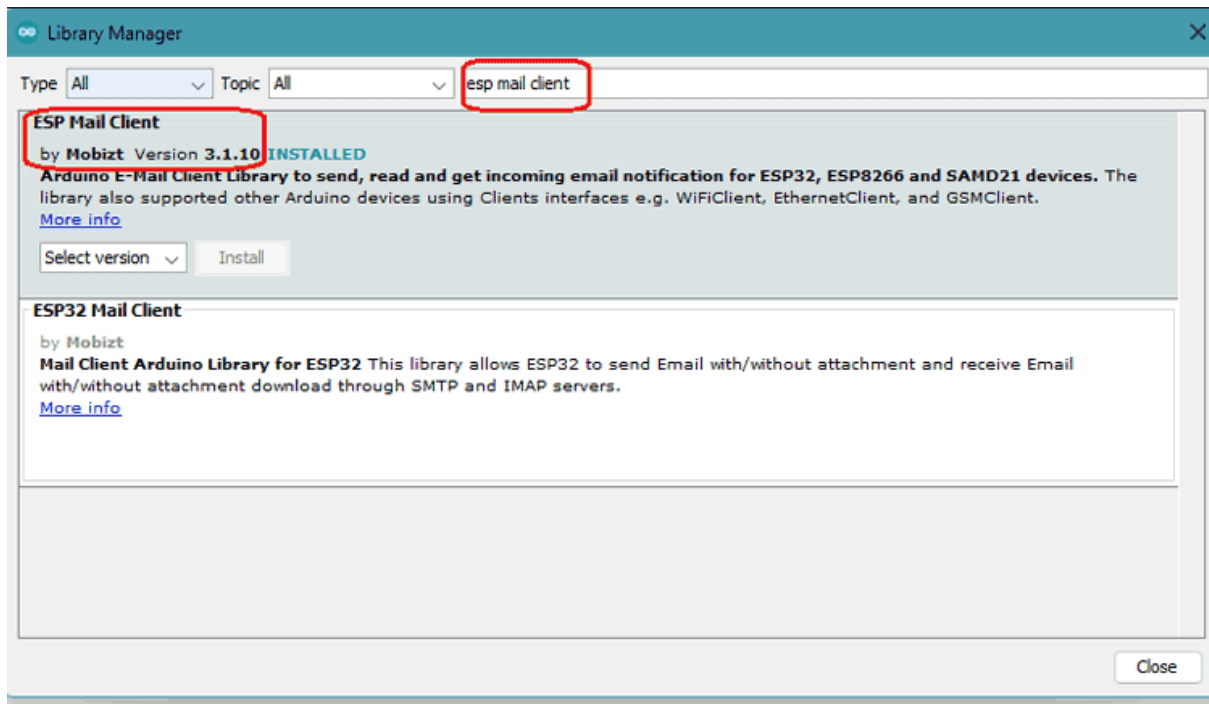
The program you create can include various features such as customizing the email message content, specifying the recipient's email address, and including message or other data in the email.

- Set up the **ESP32 microcontroller board** with the Arduino IDE software development environment, including the necessary Wi-Fi and email communication libraries.
- Configure the Wi-Fi network settings on the ESP32 to enable connectivity to the internet.
- Set up an **SMTP server** to handle the email communications, and provide the required server credentials such as the server address, port number, login credentials, and security settings.
- Write a program using the Arduino IDE that uses the ESP32's Wi-Fi and email libraries to establish a connection to the SMTP server and send emails.

### ESP Mail Client Library Setup:

It is necessary to use the ESP mail client library to work with SMTP servers. So, let's learn to install the library.

- Go to the library manager of your Arduino by **Sketch> include libraries> manage libraries**.
- Search for “ESP mail client” and you will find the first library named “**ESP Mail Client by Mobitz**”.
- Install it and that’s it.



Sending HTML and Plain Text using ESP32:

The code requires all the necessary libraries for successful execution. Libraries include **ESP Mail Client**, and **Wifi.h** only. For sending mail using ESP32 requires an Internet connection and due to that we use library **WIFI.h**. We have also included some variables and objects to store data for further programming.

```
#include <WiFi.h>
#include <ESP_Mail_Client.h>
#define WIFI_SSID "Semicon Media 2.4"
#define WIFI_PASSWORD "cdfiP29to665"
#define SMTP_server "smtp.gmail.com"
#define SMTP_Port 465
#define sender_email "own.esp32@gmail.com"
#define sender_password "nlzvmivlrxdxttgy"
```

```
#define Recipient_email "xyz@gmail.com"  
#define Recipient_name ""  
SMTPSession smtp;
```

WIFI.h library connects the ESP32 to a Wi-Fi network in the Arduino IDE. You will need to provide the SSID and password of the network you want to connect to, into the created variables named WIFI\_SSID and WIFI\_Password. You need to replace these credentials with your network credentials.

Then we have to provide an SMTP server address and port variables to store values to connect to the Servers. Since we have used the Google Gmail SMTP Protocol so we have to provide a Gmail SMTP address and Port. Choosing Gmail as an SMTP server is optional, you can also choose another Mail provider's options.

Also, we defined variables for Sender's Email address & password and the Receiver's Email address & name to store the data values. The recipient's name is optional, if you want you can define otherwise leave it as it is.

```
void setup(){  
  Serial.begin(115200);  
  Serial.println();  
  Serial.print("Connecting...");  
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
  while (WiFi.status() != WL_CONNECTED)  
  { Serial.print(".");  
    delay(200);  
  }  
  Serial.println("");  
  Serial.println("WiFi connected.");  
  Serial.println("IP address: ");  
  Serial.println(WiFi.localIP());  
  Serial.println();
```



```
smtp.debug(1);
```

Inside the Setup function(), we initialize the serial port on baud rate 115200. Then we start to command WiFi.begin() to initialize the WIFI to connect to a network by providing the correct SSID and password.

Then we monitor the connection by command WL\_CONNECTED and, we print periods to the serial monitor as a progress indicator by a delay of 200 milliseconds. Finally, the IP Address will print on the serial monitor after a successful connection. And finally, we start the function of object 'smtp' that is created of the class "SMTPsession".

```
ESP_Mail_Session session;  
  
session.server.host_name = SMTP_server ;  
  
session.server.port = SMTP_Port;  
  
session.login.email = sender_email;  
  
session.login.password = sender_password;  
  
session.login.user_domain = "";
```

We have configured the Session by creating an object named '**session**' of the class '**ESP\_Mail\_Session**' and providing all the details to the class members which are defined earlier as variables.

```
SMTP_Message message;  
  
message.sender.name = "ESP 32";  
  
message.sender.email = sender_email;  
  
message.subject = "ESP32 Testing Email";  
  
message.addRecipient(Recipient_name,Recipient_email);
```

Now, in the above lines of code, we can provide the details of the email message which we going to send. It includes the sender's name, email, and message subject which is the subject of the mail. Some of these variables we have defined already but you need to change them as per your communication data. Else subject and sender name can be directly defined here as per your choice.

### **HTML message body**

```
//Send HTML message
```

```
String  htmlMsg  =  "<div  style=\"color:  #000000;\"><h1>  Hello  
Semicon!</h1><p> Mail Generated from  ESP32</p></div>";  
  
message.html.content = htmlMsg.c_str();  
message.html.content = htmlMsg.c_str();  
message.text.charSet = "us-ascii";  
message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
```

The above lines of code define the **HTML text body**. For the heading, we are specifying 'Hello Semicon'. The next line will consist of 'Mail Generated from ESP32'. This message displayed to the sender will be in Black as we have specified the hex code for that. You can add your own HTML text and settings according to your preference.

### Simple Text Message body

```
//Send simple text message  
  
String textMsg = "How are you doing";  
message.text.content = textMsg.c_str();  
message.text.charSet = "us-ascii";  
message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
```

The above code sends plain text but before that, you have to comment out the complete HTML text body.

```
if (!smtp.connect(&session))  
    return;  
if (!MailClient.sendMail(&smtp, &message))  
    Serial.println("Error sending Email, " + smtp.errorReason());  
}
```

The following line of code check and start the SMTP session and send the mail. In the last, you will find the complete code.

### Working

The main task we are going to do is to send an email via the Gmail SMTP server. The necessary changes shouldn't be forgotten in the code like Receiver and sender Gmail addresses and also comment out any of one in the HTML body or the Plain text body inside the code.

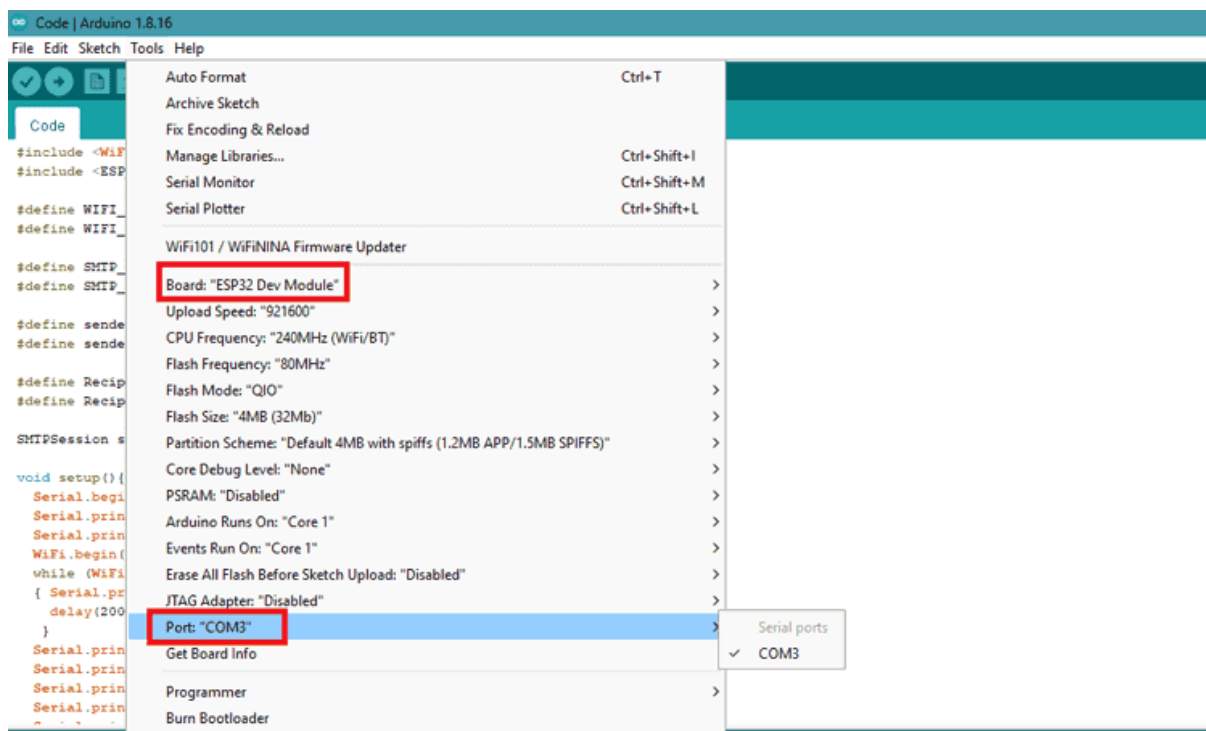
Configure the message body as per your requirement like changing the subject, sender name, and message.

Also don't forget to provide your WiFi credentials to successfully connect the board to the internet.

The sender password is the **App password** (created earlier) which is the only key to access the Sender's Gmail account. So be careful while copy-pasting it.

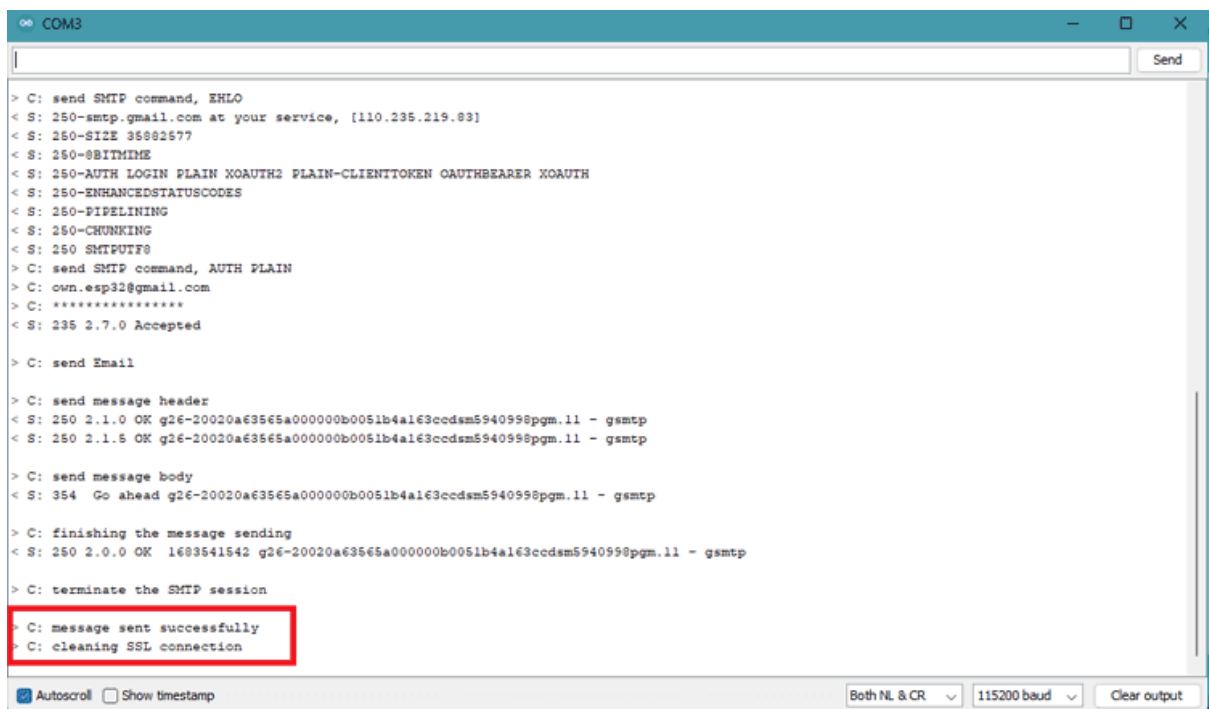
### Further working steps:

- Open the code sketch in Arduino IDE, and connect the ESP32 module via a micro-USB cable.
- Select the board as **ESP32 Dev Module** (only work if you installed ESP32 board manager already)
- Select a **COM** port



Upload the code by following the steps already discussed above and give it a try. Open the serial monitor set the correct baud rate and wait until it shows an IP Address.

On the serial monitor, there will be every information you get like whether mail sends, Gmail authorization is accessed, or not.



```
COM3
> C: send SMTP command, EHLO
< S: 250-smtp.gmail.com at your service, [110.235.219.83]
< S: 250-SIZE 35882577
< S: 250-8BITMIME
< S: 250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
< S: 250-ENHANCEDSTATUSCODES
< S: 250-PIPELINING
< S: 250-CHUNKING
< S: 250 SMTPUTF8
> C: send SMTP command, AUTH PLAIN
> C: own.esp32@gmail.com
> C: *****
< S: 235 2.7.0 Accepted

> C: send Email

> C: send message header
< S: 250 2.1.0 OK g26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmtP
< S: 250 2.1.5 OK g26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmtP

> C: send message body
< S: 354 Go ahead g26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmtP

> C: finishing the message sending
< S: 250 2.0.0 OK 1683541542 g26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmtP

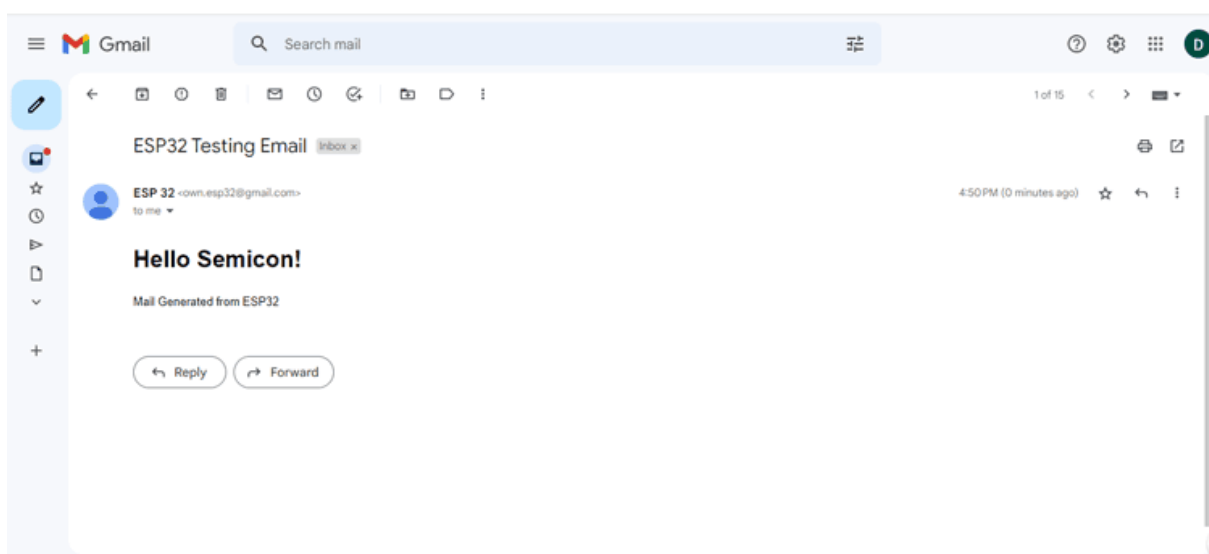
> C: terminate the SMTP session

< C: Message sent successfully
< C: cleaning SSL connection

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output
```

Finally, if everything is okay, a successfully sent message will be displayed. Check for the mail on your Email ID. An Email will be sent whenever you boot up the device.

Let's see how real our mail looks:



# **CHAPTER 9**

## 9. SOFTWARE CODE

```
#include <esp_camera.h>

#include <WiFi.h>;

#include<ESP_Mail_Client.h>;

#define ssid <"AndroidAPA97A">;

#define password <"rohini19">;


#define SMTP_server <"smtp.gmail.com">;

#define SMTP_Port 465

#define sender_email <"e.rohininew@gmail.com">;

#define sender_password <"samsunggalaxya03s">;

#define Recipient_email <"iot2automation@gmail.com">;

#define Recipient_name <"rohivaish19">;

//

// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG
// quality

//      Ensure ESP32 Wrover Module or other board with PSRAM is selected

//      Partial images will be transmitted if image exceeds buffer size

//

//      You must select partition scheme from the board menu that has at
//      least 3MB APP space.

//      Face Recognition is DISABLED for ESP32 and ESP32-S2, because it
//      takes up from 15

//      seconds to process single frame. Face Detection is ENABLED if
```

PSRAM is enabled as well

```
// =====
```

```
// Select camera model
```

```
// =====
```

```
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
```

```
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
```

```
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
```

```
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
```

```
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version  
B Has PSRAM
```

```
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
```

```
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
```

```
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
```

```
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
```

```
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
```

```
//#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
```

```
// ** Espressif Internal Boards **
```

```
//#define CAMERA_MODEL_ESP32_CAM_BOARD
```

```
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
```

```
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
```

```
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
```

```
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
```

```
#include "camera_pins.h";
```

```
// =====
```

```
// Enter your WiFi credentials

const char* ssid = &quot;AndroidAPA97A&quot;;

const char* password = &quot;rohini19&quot;;

void startCameraServer();

void setupLedFlash(int pin);

SMTPSession smtp;

void setup() {

    Serial.begin(115200);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)

    { Serial.print(&quot;.&quot;);

        delay(200);

    }

    Serial.println(&quot;&quot;);

    Serial.println(&quot;WiFi connected.&quot;);

    Serial.println(&quot;IP address: &quot;);

    Serial.println(WiFi.localIP());

    Serial.println();

    smtp.debug(1);

    Serial.setDebugOutput(true);

    Serial.println();

    camera_config_t config;

    config.ledc_channel = LEDC_CHANNEL_0;

    config.ledc_timer = LEDC_TIMER_0;
```



```
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming

//config.pixel_format = PIXFORMAT_RGB565; // for face
detection/recognition

config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
```

```

config.fb_count = 1;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
//           for larger pre-allocated frame buffer.
if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
        config.jpeg_quality = 10;
        config.fb_count = 2;
        config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
        // Limit the frame size when PSRAM is not available
        config.frame_size = FRAMESIZE_SVGA;
        config.fb_location = CAMERA_FB_IN_DRAM;
    }
} else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#ifdef CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
}

#ifdef CAMERA_MODEL_ESP_EYE
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);

```

```

#endif

// camera init

esp_err_t err = esp_camera_init(&config);

if (err != ESP_OK) {

    Serial.printf(&quot;Camera init failed with error 0x%x&quot;, err);

    return;

}

sensor_t * s = esp_camera_sensor_get();

// initial sensors are flipped vertically and colors are a bit saturated

if (s-&gt;id.PID == OV3660_PID) {

    s-&gt;set_vflip(s, 1); // flip it back

    s-&gt;set_brightness(s, 1); // up the brightness just a bit

    s-&gt;set_saturation(s, -2); // lower the saturation

}

// drop down frame size for higher initial frame rate

if(config.pixel_format == PIXFORMAT_JPEG){

    s-&gt;set_framesize(s, FRAMESIZE_QVGA);

}

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)

    s-&gt;set_vflip(s, 1);

    s-&gt;set_hmirror(s, 1);

#endif

```

```
#if defined(CAMERA_MODEL_ESP32S3_EYE)

    s->set_vflip(s, 1);

#endif

// Setup LED FLash if LED pin is defined in camera_pins.h

#if defined(LED_GPIO_NUM)

    setupLedFlash(LED_GPIO_NUM);

#endif

WiFi.begin(ssid, password);

WiFi.setSleep(false);

while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use http://");

Serial.print(WiFi.localIP());

Serial.println(" to connect");

ESP_Mail_Session session;

session.server host_name = SMTP_server;

session.server port = SMTP_Port;

session login email = sender_email;

session login password= sender_password;
```

```

    session login user_domain= &quot;&quot;;

/* Declare the message class */

SMTP Message message;

message.sender.name = &quot;ESP 32&quot;;

message.sender.email sender_email;

message.subject = &quot;Theft alert&quot;;

message.addRecipient (Recipient_name, Recipient_email);


//Send HTML message

String htmlMsg = &quot;&lt;div style=\&quot;color:
#000000;\&quot;&gt;&lt;h1&gt;Hello

Semicon!&lt;/h1&gt;&lt;p&gt;Mail Generated from
ESP32&lt;/p&gt;&lt;/div&gt;&quot;; message.html.content=
htmlMsg.c_str();

message.html.content = htmlMsg.c_str();

message.text.charset= &quot;us-ascii&quot;;

message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

if (!smtp.connect (&session))

    return;

if (!MailClient.sendMail(&smtp, &message))

    Serial.println(&quot;Error sending Email,&quot; + smtp.errorReason());
}

void loop() {

    // Do nothing. Everything is done in another task by the web server

    delay(10000);

```

## 9.1 CODE EXPLANATION

This code is a setup configuration for an ESP32-based camera system using the ESP-IDF framework. Let's go through the important parts:

### 1. WiFi and Email Configuration:

- Defines WiFi credentials (``ssid`` and ``password``).
- Configures the SMTP server details for sending emails: server address (``SMTP_server``), port (``SMTP_Port``), sender's email and password (``sender_email`` and ``sender_password``), and recipient's email and name (``Recipient_email`` and ``Recipient_name``).

### 2. Camera Model Selection:

- The code includes a section for selecting the camera model. You can uncomment one of the ``#define`` statements based on your specific camera model.
- Each camera model may have different specifications, and the selection impacts how the camera is initialized.

### 3. Camera Pins Configuration:

- The code includes an external file (``camera_pins.h``) for defining camera-specific pin mappings.
- The actual file (``camera_pins.h``) would contain pin assignments for the chosen camera model.

The commented-out ``#define`` statements allow you to choose the appropriate camera model for your setup. Depending on the selected model, PSRAM (programmable static random-access memory) may be required for certain resolutions and image qualities. Additionally, some models may have specific pin configurations, which are specified in the associated ``camera_pins.h`` file.

Make sure to uncomment only one ``#define CAMERA_MODEL_...`` line based on your actual camera module, and ensure the associated ``camera_pins.h`` file is correctly configured for your camera model.

### 4. WiFi Setup:

- It starts by initializing serial communication and connecting to a WiFi network using provided SSID and password.
- Waits until a successful WiFi connection is established.

#### 5. Camera Configuration:

- Defines camera configuration parameters such as pins, resolution, pixel format, etc.
- Adjusts settings based on the presence of PSRAM (programmable static random-access memory) for improved performance.
- Handles different frame sizes and pixel formats based on the camera model and ESP32 variant.

#### 6. Camera Initialization:

- Initializes the camera with the specified configuration.
- Checks for errors during initialization and prints a message if an error occurs.

#### 7. Sensor Configuration:

- Retrieves the camera sensor information and adjusts settings.
- Specifically, for the OV3660 sensor, it flips the image vertically, increases brightness, and decreases saturation.

#### 8. Frame Size Adjustment:

- Adjusts the frame size for higher initial frame rate, especially when using JPEG format.

The code seems to be part of a larger project involving a camera module, likely for capturing images or video. The conditional statements and configurations indicate adaptability to different camera models and ESP32 variants. Additionally, there's a mention of PSRAM usage for better performance when available.

This code appears to be setting up a camera system using the ESP32 framework, connecting to WiFi, and sending an email with an alert. Let's break it down:

#### 9. Camera Orientation Configuration:

- If the camera model is M5STACK\_WIDE or M5STACK\_ESP32CAM, it flips the image vertically and mirrors it horizontally.
- If the camera model is ESP32S3\_EYE, it flips the image vertically.

#### 10. LED Flash Setup:

- If an LED pin is defined (`LED_GPIO_NUM`), it calls the `setupLedFlash` function with that pin.

#### 11. WiFi Connection Setup:

- Initiates WiFi connection with the provided SSID and password.
- Disables sleep mode for WiFi.

- Waits for a successful WiFi connection.

#### 12. Camera Server Initialization

- Calls ``startCameraServer()`` to initialize the camera server.

#### 13. Prints WiFi Information:

- Prints the local IP address to the Serial Monitor, indicating a successful WiFi connection.

#### 14. Email Configuration:

- Sets up an ESP Mail Session with SMTP server details.
- Creates an SMTP message with a sender, subject, recipient, and an HTML message body.
- The HTML message body contains a simple greeting generated from the ESP32.

#### 15. Email Sending:

- Connects to the SMTP server using the configured session.
- Sends the email with the constructed message.
- Prints an error message if there is an issue with sending the email.

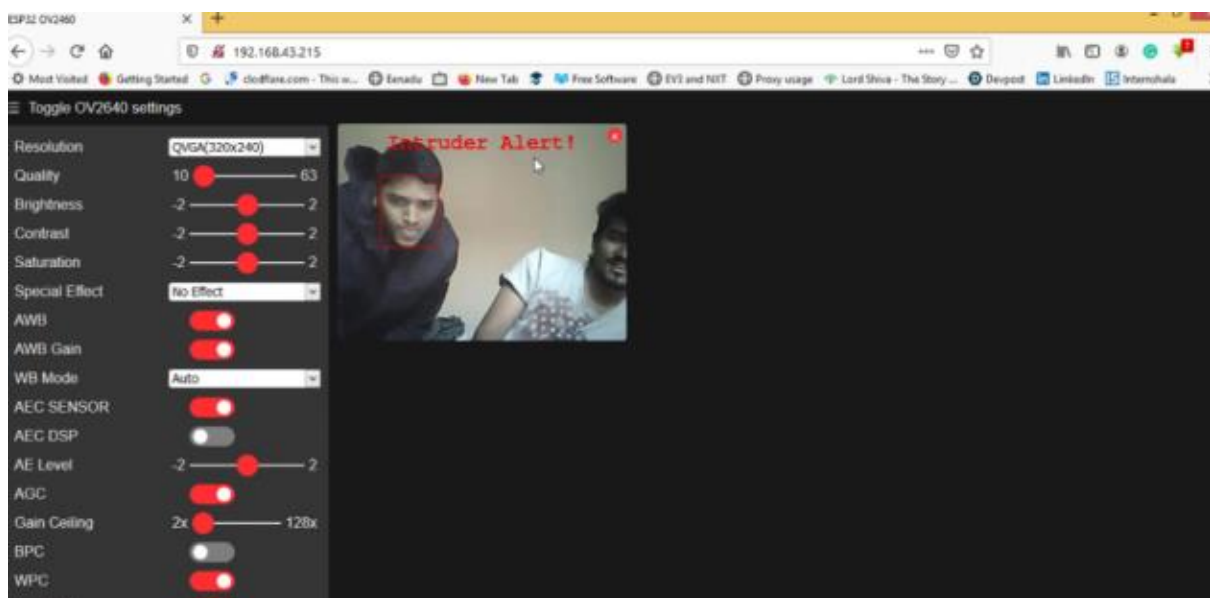
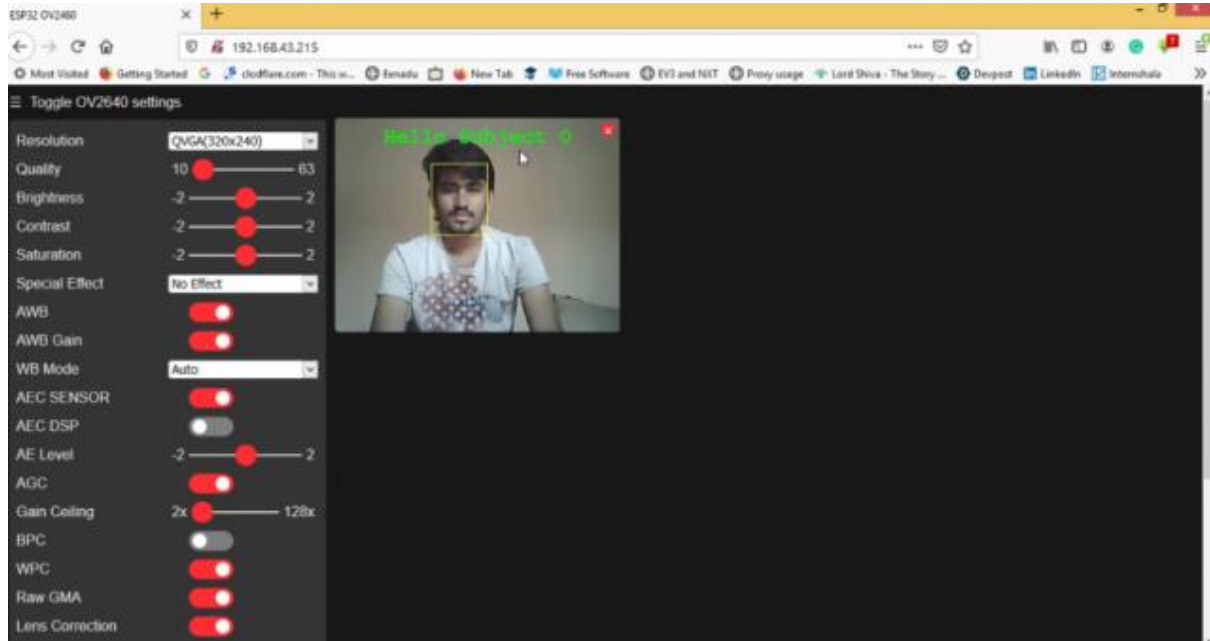
#### 16. Main Loop:

- The ``loop`` function does nothing but includes a delay of 10 seconds between iterations.

Overall, the code initializes the camera, connects to WiFi, and sends an email with a predefined message. It's designed to run as a one-time setup in the ``setup`` function, and the main functionality is in another task handled by the web server.



## 9.2 OUTPUT



```
COM3

> C: send SMTP command, EHLO
< S: 250-smtp.gmail.com at your service, [110.235.219.83]
< S: 250-SIZE 35882577
< S: 250-8BITMIME
< S: 250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
< S: 250-ENHANCEDSTATUSCODES
< S: 250-PIPELINING
< S: 250-CHUNKING
< S: 250 SMTPUTF8
> C: send SMTP command, AUTH PLAIN
> C: own.esp32@gmail.com
> C: *****
< S: 235 2.7.0 Accepted

> C: send Email

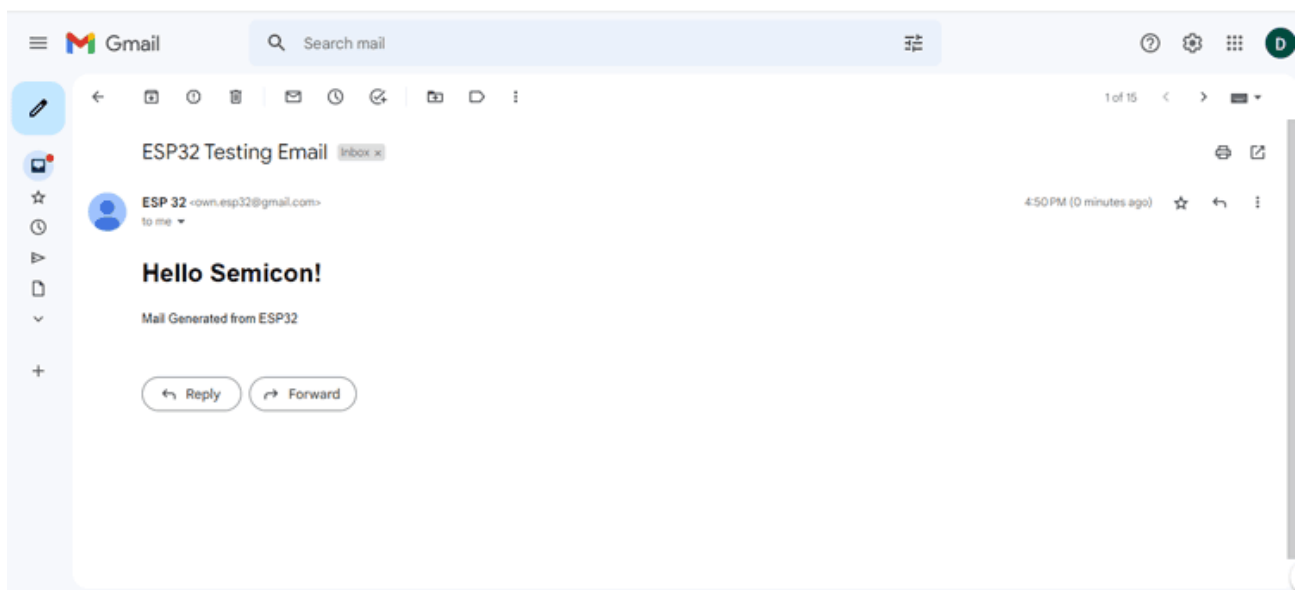
> C: send message header
< S: 250 2.1.0 OK q26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmt
< S: 250 2.1.5 OK q26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmt

> C: send message body
< S: 354 Go ahead q26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmt

> C: finishing the message sending
< S: 250 2.0.0 OK 1693541542 q26-20020a63565a000000b0051b4a163ccdsM5940998pgm.11 - gsmt

> C: terminate the SMTP session
> C: message sent successfully
> C: cleaning SSL connection

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output
```



# **CHAPTER 10**

## 10. APPLICATION

A smart antitheft system using Arduino Uno and ESP32-CAM can have various applications, including:

### 1. Home Security:

Implement the system to secure homes by monitoring and capturing images or videos when unauthorized access is detected.

### 2. Vehicle Security:

Use the system to protect vehicles by installing cameras that can capture images of potential thieves or suspicious activities.

### 3. Office Security:

Employ the system to enhance office security, capturing footage of any unauthorized access or suspicious behavior.

### 4. Personal Belongings Security:

Create a portable antitheft device for personal belongings, such as bags or valuable items, with a compact ESP32-CAM module.

### 5. Remote Monitoring:

Enable remote monitoring through internet connectivity provided by the ESP32-CAM, allowing users to view real-time images or videos from their smartphones.

### 6. Notification System:

Implement a notification system that alerts the user through messages or emails when the antitheft system is triggered.

### 7. Integration with IoT Devices:

Connect the system with other IoT devices, such as smart locks or alarms, to create a comprehensive security solution.

### 8. Data Logging:

Utilize the Arduino Uno to log data related to security events, providing a record of incidents for later analysis.

Remember to consider legal and ethical aspects when implementing such systems, respecting privacy and abiding by local regulations.

# CHAPTER 11

## **11. FUTURE ENHANCEMENT**

Future enhancements for a smart antitheft system using Arduino Uno and ESP32-CAM with email notifications could include:

### **1. Live Streaming Integrate :**

a live streaming feature, allowing the owner to view real-time footage remotely. This could provide immediate insights into the situation and facilitate quick decision-making.

### **2. Cloud Storage Integration:**

Implement cloud storage for storing captured images or videos. This would enable the owner to access a historical record of security events, even if the device is tampered with or damaged.

### **3. Machine Learning for Object Recognition:**

Explore the integration of machine learning algorithms for object recognition. This could help differentiate between normal activities and potential threats, reducing false alarms and enhancing the system's intelligence.

### **4. Two-Way Communication:**

Include two-way communication capabilities, allowing the owner to communicate with individuals near the system. This could serve as an additional deterrent and provide a means for verification.

### **5. Mobile App Integration:**

Develop a mobile application that interfaces with the antitheft system. This app could provide a user-friendly interface for monitoring, configuring settings, and receiving alerts, complementing the email notifications.

### **6. Geofencing:**

Implement geofencing capabilities to automatically activate or deactivate the antitheft system based on the owner's location. This adds an extra layer of automation and convenience.

### **7. Multiple Camera Support:**

Allow the system to support multiple cameras, providing comprehensive coverage of larger areas or different angles, and facilitating a more detailed analysis of security events.

### **8. Energy Efficiency:**

Optimize power consumption to prolong the system's battery life, especially if it's deployed in a location without a constant power source. This could involve incorporating low-power modes or alternative energy sources.

### **9. Integration with Smart Home Systems:**

Enable integration with popular smart home platforms, allowing the antitheft system to collaborate with other smart devices like lights, locks, or alarms for a synchronized security response.

### **10. Enhanced Security Measures:**

Implement additional security measures such as encryption for data transmission, secure storage for captured media, and advanced authentication methods to prevent unauthorized access to the system.

To improve the accuracy of person detection, consider using machine learning models like Haar cascades or TensorFlow Lite. Implement a time delay between notifications to prevent spamming in case of continuous motion detection.

Continued research and development in these areas can contribute to a more sophisticated and robust smart antitheft system, addressing evolving security challenges and user needs.

# CHAPTER 12



## **12. CONCLUSION**

In conclusion, the integration of email notifications in a smart antitheft system using Arduino Uno and ESP32-CAM significantly enhances its functionality. This feature allows for timely alerts to the owner, providing a proactive approach to security. By capturing images or videos upon detection of suspicious activity and sending them via email, the system enables remote monitoring and rapid response.

The system's ability to connect to the internet, securely handle email credentials, and communicate with the owner through email notifications makes it a versatile and effective tool for various applications, including home security, vehicle protection, and personal belongings security. This implementation not only serves as a deterrent to potential thieves but also provides valuable evidence for further analysis and action.

While implementing such systems, it's essential to consider privacy and adhere to legal regulations. Additionally, continuous testing and refinement are crucial to ensuring the reliability and accuracy of the antitheft system. Overall, the combination of Arduino Uno and ESP32-CAM, augmented with email notifications, represents a robust and accessible solution for enhancing security in different scenarios.

# CHAPTER 13

### 13. REFERENCES

1. IndiaInternational Research Journal of Engineering and Technology(IRJET) M. Shamita2[1]M. Tech. Student, Dept. of Electronics and Communication Engineering, PDIT, Hospet, Karnataka, India [2]Associate Professor, Dept. of Electronics and Communication Engineering, PDIT, Hospet, Karnataka.
2. T. Ahmed, S. Ahmed, S. Ahmed, and M. Motiwala, "Real-Time Intruder Detection in Surveillance Networks Using Adaptive Kernel Methods," 2010 IEEE International Conference on Communications, 2010.
3. Z. Zhang, D. Yi, Z. Lei, and S. Z. Li, "Regularized Transfer Boosting for Face Detection Across Spectrum," IEEE Signal Processing Letters, vol. 19, no. 3, pp. 131–134, 2012.Siddalingesha G. R.1H.
4. M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces", published on the IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 91, pp. 586-591, 1991.
5. Stan Li and Kapluk Chan, Guodong Guo "Face Recognition by Support Vector Machines" IEEE Transactions on Pattern Analysis And Maciine Intelligence, Vol. 19, No. 7 July 2018.
6. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 2, Issue 6, June 2022 Copyright to IJARSCT DOI: 10.48175/IJARSCT-5093680 [www.ijarsct.co.in](http://www.ijarsct.co.in) Impact Factor: 6.252 Home Security System using ESP32-CAM and Telegram Application Dr. G. C. Manjunath1,Mr. B. Mahendra2 Ms. Rashmi , Mrs. G. Bhuvana, Ms. Keerthi5 Professor1 andStudent2,3,4,5Proudhavevaraya Institute of Technology, Hospet, Karnataka, India.
7. Security and Safetywith FacialRecognition Feature for Next Genenation Automobiles "Nalini Nagendran, Ashwini Kolhe (2018).
8. S. Kim, D. H. Yeom, Y. H. Joo, and J. B. Park, "Intelligent Unmanned anti-theft system using network camera," International Journal of Control, Automation and Systems Int. J. Control Autom. Syst., vol. 8, no. 5, pp. 967–974, 2010.
9. "Identity Theft Research," Detection, Prevention, and Security Identity Theft Handbook, pp. 263–276, 2015.