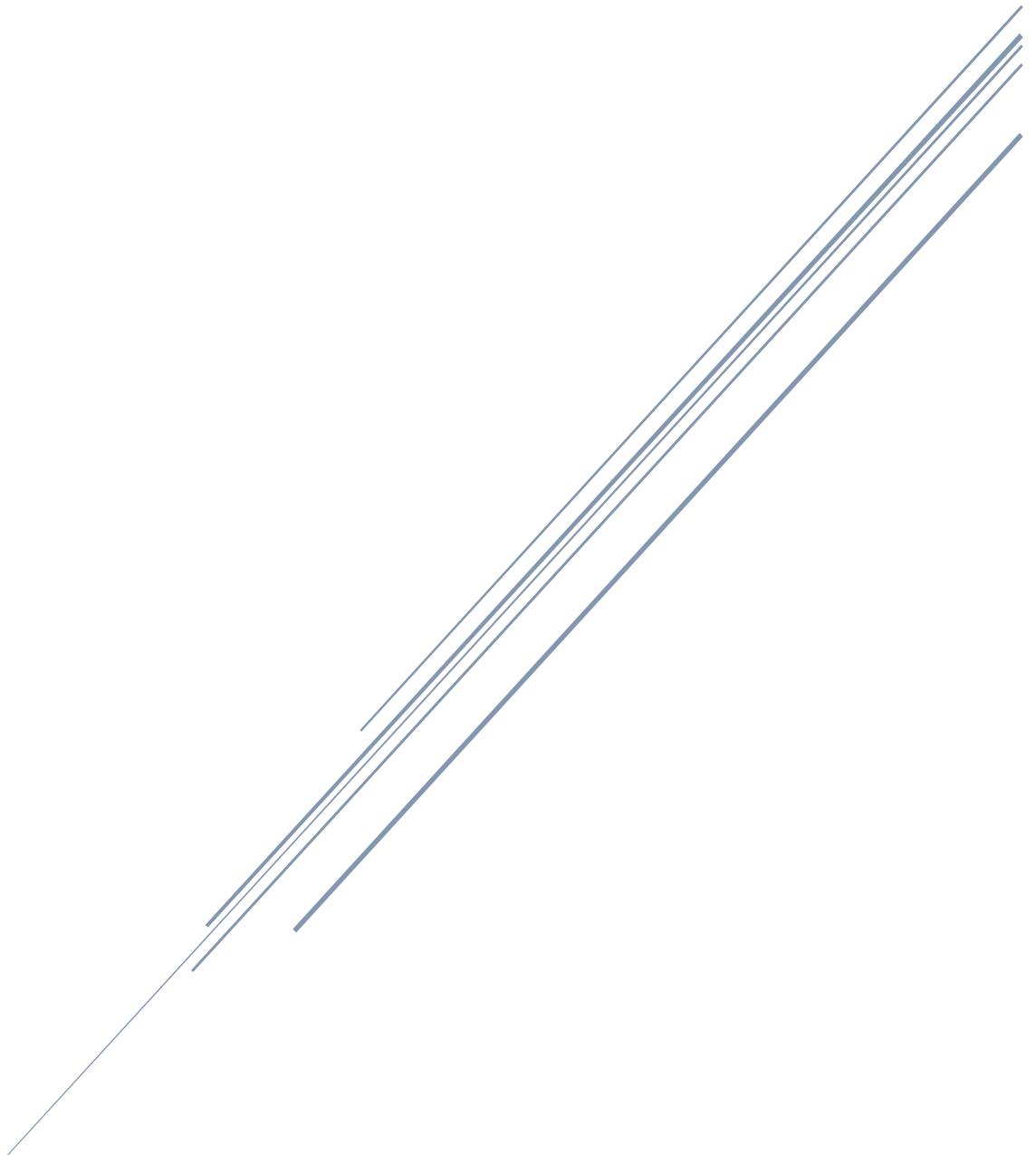


# ARDUINO HOME SECURITY SYSTEM

Nishat Humaira 001043975



University of Greenwich  
BSc Computer Science

## Abstract

Smart home is a concept referring to convenient electronic equipment in a home that can be monitored and controlled remotely. The key problem in the implementation of a smart home is how costly it can be tied in with unnecessary complexity thus overlooking the importance of user interaction. This paper will present a design and implementation of a smart home security system using Arduino that is cost effective and controlled with an android application. The application will facilitate users to control the home security system as well as monitor their home through the camera. The user interface design is aimed to be an easy to use system where users can monitor their home remotely so beginner to advanced level Internet users can use this system with ease.

## Acknowledgements

I would like to thank my project supervisor's Dr Atsushi Suzuki and Ms Yasmine Arafa for their exceptional guidance, valuable insights, unending support, and immense patience. A huge thanks to FLAS Department for all their support as well as Stockwell Street Workshop for the device design.

Finally, I would like to thank my family for their encouragement, motivation, and emotional support. Without them my project could never have been completed.

## Table of Contents

<b>ABSTRACT.....</b>	<b>1</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>2</b>
<b>INTRODUCTION .....</b>	<b>5</b>
<b>LITERATURE REVIEW .....</b>	<b>6</b>
EXISTING SYSTEMS.....	6
<b>METHODOLOGY.....</b>	<b>9</b>
ANALYSIS .....	9
HARDWARE REQUIREMENTS.....	10
SOFTWARE REQUIREMENTS.....	13
<b>SYSTEM DESIGN .....</b>	<b>14</b>
BLOCK DIAGRAM.....	14
FLOWCHART .....	15
<i>Schematic Diagram.....</i>	<i>16</i>
<b>IMPLEMENTATION .....</b>	<b>19</b>
PROTOTYPE .....	19
SERVER-SIDE IMPLEMENTATION .....	20
CLIENT-SIDE IMPLEMENTATION .....	22
<i>Login Screen Development .....</i>	<i>23</i>
<i>Home Screen Development .....</i>	<i>28</i>
FINAL PRODUCT .....	36
<b>TESTING.....</b>	<b>37</b>
TEST RESULTS SUMMARY:.....	38
<b>PRODUCT EVALUATION.....</b>	<b>39</b>
<b>CONCLUSION .....</b>	<b>39</b>
FUTURE WORK .....	40
<b>LEGAL AND ETHICAL ISSUES.....</b>	<b>40</b>
<b>REFERENCES .....</b>	<b>40</b>
<b>APPENDIX.....</b>	<b>42</b>

Figure 1: SimpliSafe .....	6
Figure 2: SimpliSafe parts price list.....	6
Figure 3: Home Monitoring and Security system hardware unit.....	7
Figure 4: Text alert message .....	8
Figure 5 : Agile Methodology.....	9
Figure 6: Arduino Uno Rev3 .....	10
Figure 7: PIR Sensor Pin Diagram.....	11
Figure 8 : HC-SR501 PIR Sensor .....	11
Figure 9: ESP32-CAM.....	12
Figure 10: ESP-WHO Development Framework .....	13
Figure 11 : Arduino Smart Home Security System Block Design .....	14
Figure 12: System Flowchart .....	15
Figure 13 : Mobile Application Flowchart .....	16

Figure 14: Arduino UNO and PIR Sensor Schematic Diagram .....	17
Figure 15: Arduino UNO and ESP32-CAM Schematic Diagram .....	17
Figure 16: ESP32-CAM and PIR Sensor Schematic Diagram .....	18
Figure 17: ESP32-CAM pins (Seeedstudio.com, 2018) .....	19
Figure 18: AHSS Prototype .....	20
Figure 19: Serial Monitor Output .....	21
Figure 20: ESP32-CAM Video Streaming Web Server .....	21
Figure 21: PIR Sensor Code - esp32cam.ino .....	22
Figure 22: MainActivity.java I .....	23
Figure 23: AHSS Application Login Screen .....	24
Figure 24: MainActivity.java II .....	25
Figure 25: MainActivity.java III .....	26
Figure 26: Incorrect Login Credentials .....	27
Figure 27: Login Button Disabled .....	27
Figure 28: Initial Defective Home Screen .....	28
Figure 29: Login Success - Access Granted .....	28
Figure 30: MainScreen.java .....	29
Figure 31: Live Video Feed on App .....	30
Figure 32: Recognised Face Detected.....	31
Figure 33: Unrecognised Face Highlights Intruder Alert .....	31
Figure 34: Face Recognition & Detection in App .....	32
Figure 35: MyFirebaseMessagingService.java .....	33
Figure 36: AndroidManifest.xml .....	34
Figure 37: Firebase implentation in Arduino 'app_httpd.cpp'.....	34
Figure 38: Firebase implentation in Arduino 'app_httpd.cpp'.....	35
Figure 39: Arduino Face Recogniton ESP32-CAM; app_httpd.cpp .....	35
Figure 40: Arduino Home Security System - Final Product.....	36

## Introduction

Due to intrusion and theft constantly occurring all around us home security is a major concern, in some cases they occur when the occupants are not home and also even with occupants in the house. Intruders can make their way in easily unnoticed by anyone. Therefore, many smart security systems have been developed over the years using various sensors, cameras and wireless technologies. Notably, some of the developed smart home security systems have limitations like operating ranges, unnecessary complexity and overlooks the importance of user interaction. Also, the mobile applications developed in order to facilitate users to control their smart security system lack innovation and in most cases do not support a camera to even monitor the home.

This project is set to develop a secure, standard and low-cost smart security system for the home using Arduino, with an easy to use system and simple mobile application design to monitor their home remotely so beginner to advanced level Internet users can use this system with ease. This system comprises of two parts: A mobile application as well as a unit made up of programmable Arduino board, Passive Infrared Sensor (PIR) and ESP32-CAM.

The hardware choices surrounding this project is based on its reliability to perform, cost-effectiveness and previous developments made with these modules. Arduino UNO, a microcontroller, has been chosen as a data processor. It is a programmable circuit board and will interact with the chosen PIR, to detect movement. The PIR used to sense motion is the HC-SR501 PIR Sensor Module as it can be used to detect whether a human has moved in or out of the sensor's range, if motion is detected the camera should be triggered to take a photo of the person (Makhanya, et al., 2019).

The ESP32-CAM chosen to be used to do this is a low-cost Wi-Fi camera that allows you to stream video and even some image filtering and face detection/recognition, thus why it will be essential part of this smart home security system, not only will it allow you to view the camera but also saves the photos captured in its MicroSD card slot. It will allow users to view their home from anywhere, notify them of intruders through the mobile application developed which also facilitates users to control the system simply by pressing buttons. Face recognition is a method of recognising a human face by comparing it to stored information in a database of known faces. It is an integral part of this smart home security system, but there is no need to pay hundreds of pounds for a camera, processor and Wi-Fi capabilities when the ESP32-CAM can stream videos/images over Wi-Fi as well as perform face recognition and detection, all via a demo program included with the drivers (Cook, 2020). The innovative aspect of this project involves the ESP32-CAM, it will recognise the individuals that reside in that home but for any motion that is detected and an unrecognised face is captured it will alert the home owner and store the photo of the intruder for relevant security measures to be conducted. Doing so, will filter out the number of notifications the user receives that are perhaps just family members and therefore not a security threat.

In line with current development this could be a suitable security system that prevents users from being victims of criminal cases like burglary and theft as not only will it notify residents of intruders but it will also give them the reassurance their home is safe through real-time video streaming of their home, it will have greater user satisfaction if they receive fewer alerts about motion being sensed of just members of the household as they pose no threat to residents, their property and valuables.

## Literature Review

A smart home essentially is a convenient integrated set-up consisting of smart (self-learning) systems that allows homeowners to control appliances, lights, cameras and other devices remotely through an internet connection. Together these systems work toward enhancing the comfort, safety, and convenience of the smart home's residents. It is ever so simple to control the home lights, switches, doors, camera's and other most used electric appliances from a laptop or smart phone with just a single touch (Gruhn, 2018). This smart home security system developed at a low cost can essentially be used for upgrading current homes into smart homes at an affordable cost. In the last few decades the technologies involved in home security systems have been constantly evolving and will continue to do so in the coming years. Majority of these home security systems provide a safe environment as well as protect valuables from theft.

## Existing Systems

Existing smart security systems that currently exist on the market, one of which being SimpliSafe, is a smart home security system with innovative technology, it's professionally monitored 24/7. Ready to alert emergency services when users home requires help (SimpliSafe, 2021) . Their packages include detecting fires and burst pipes as well as water leaks. The fact that they offer a smart camera to let users see what is happening live in an emergency, as well as capture crucial evidence from anywhere and watch at any time is the inspiration in it itself for this project. However, the goal is to produce a low-cost standard security system for homes and this SimpliSafe security system with the website stating for an 'affordable' monthly fee, is in fact not so affordable. In Figure \_ the prices for just the parts for this system are listed, notably, these prices do not include the package cost itself, as including monitoring totals to over £200.




Figure 1 shows various SimpliSafe components: a square entry sensor, a round motion sensor, a black keypad, a black camera, and a small black base unit.

### Choose your Burglary Sensors

Sensor Type	Description	Price	Quantity Selector
Entry Sensor	Protects your doors and windows. Works with any door or window. <b>Pro tip:</b> Stop burglars where they break in most. Ground floor doors and windows. For rooms with 3 or more windows, consider a motion sensor instead.	£14.99 each	0
Motion Sensor	Covers an entire room. Get at least one for a main room or hallway. Gets along great with pets. <b>Pro tip:</b> Get one to protect rooms on the ground level with 3 or more windows. Skip getting entry sensors for these windows.	£29.99 each	0
Extra SimpliCam	See what's happening at home anytime from your phone, tablet or computer. Get alerts when your camera detects motion. <b>Pro tip:</b> Place your camera in a main hallway or room that sees a lot of foot traffic.	£49.00 each	0

Figure 1: SimpliSafe

Figure 2: SimpliSafe parts price list

This is also the case for many other products on the markets like even Yale Smart Home Security Alarm (YaleHome, 2021), with a 4-piece kit which is self-monitored, no contract, and features window, door and movement PIR sensors with an external and internal siren. This Yale security system also is offered to customers with a similar price tag to SimpliSafe, yet considered and advertised to be 'affordable'. Hence why, for this Arduino Home Security System which can also comprise of a camera, sensor and self-monitoring capabilities, will aim to *really* be affordable, easy to use and also secure.

Global system for mobile communication (GSM) is a wireless technology, users can get alerts anywhere in the world and therefore comes in handy for home security system projects like the Multi-Functional Secured Smart Home (SSH) System by ShariqSuhail et al, (2016) that has previously been proposed focusing on GSM thus making the system independent of location. Their proposed SSH sends SMS using GSM-Module and mail through Raspberry Pi micro-controller. The prototype SSH based smart system also uses an Arduino micro-controller board for commands processing and control. Essentially a Web-cam takes photos whenever the PIR sensor is triggered it signals the micro-controller and sends the photo to the home owner via email, this rather impressive feature will be considered for this project in addition to saving the photo of the intruder to an SD card within the ESP32-CAM. Doing so, will add to the convenience for users This proposed system also alerts home owners through SMS if any motion is detected within the house whilst it is activated. Moreover, GSM technology is also used to call the owner when motion is detected and incorrect password has been inputted more than three times. Notably though, ShariqSuhail et al (2016) have suggested future work to accommodate biometric system and face recognition techniques therefore for this project the use of a wireless camera and motion detector to transmit status of security levels to homeowners was considered.

Similarly, Home Monitoring and Security system proposed by Suresh et al (2016) the entire system has been developed using Arduino Uno Microcontroller, PIR sensor, DHT11 Temperature & humidity sensor, GSM SIM 900 (Figure 3). This small prototype employed not only Pyro electric sensor but also temperature and humidity sensor for detecting the intruder with Arduino Uno towards alerting the home owner via GSM for action and alert. It was done so using Arduino IDE and GSM module and shown as screenshots. This system enables the security services i.e. local police and local fire brigade to be informed about the intrusion instantly and they can take steps immediately. Suresh et al (2016) argues the case for not using a camera as “none of these systems are economical and feasible for every common man”. However, it is possible to implement a low-cost surveillance system hence when conducting further research into this; ESP32-CAM module was discovered and chosen for this system to provide an even safer environment and protect valuables from theft giving homeowners that added reassurance and peace of mind. In addition, no application was designed to facilitate users to have any sort of control over this Home Monitoring and Security System by Suresh et al (2016) remotely, they have perhaps disregarded and oversimplified certain aspects of their system which consequently overlooks the importance of user interaction.

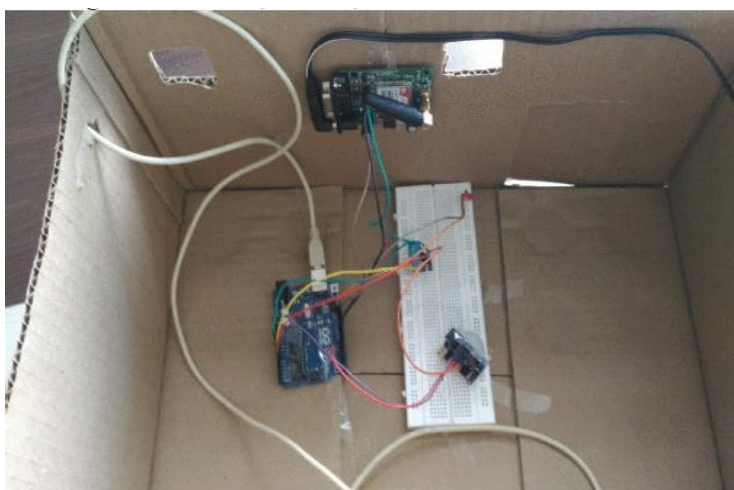
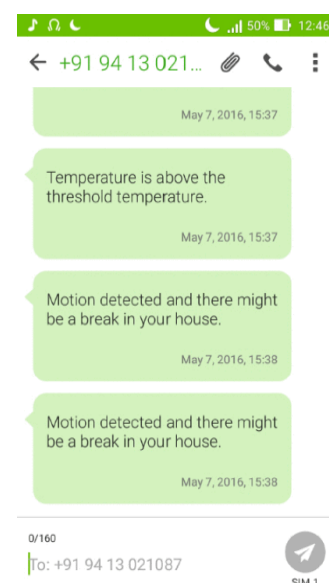


Figure 3: Home Monitoring and Security system hardware unit





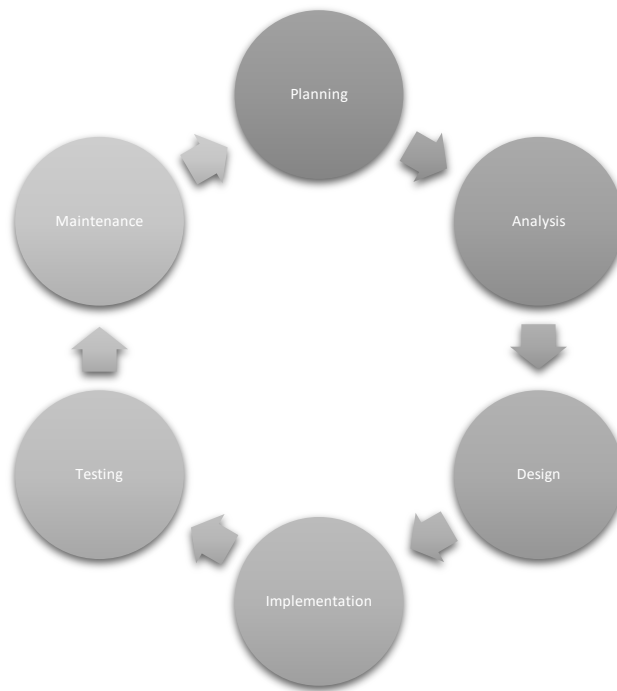
*Figure 4: Text alert message*

In contrast, the Arduino Based Home Monitoring System by Soleh (2018) is another home security system developed to help users track movements using the Arduino platform that is connected to users' smartphone using Wi-Fi network. This security system does have a mobile application at least, however, this application can only send warnings to the user but the user cannot view or monitor their residence area through this app because this system did not use camera type alarm system. Some benefits of this system that can be highlighted is that this system application can send a notification message to the user if the motion sensor detects movement over a Wi-Fi network. The application itself is a very simple design that is easy to be used by the users, perhaps too basic. This application of Wireless Motion Detection System using Arduino can help users protect their homes and valuables. The chosen technology, Arduino and Wi-Fi ESP8266 V3 ensured the connection between the alarm systems together with the developed applications, allowing users to be well informed about their home security. This system as well as many others that utilise the integration of the Internet of Thing (IoT) and smart home security systems indicates the significance of it and therefore is perfect proposition for a reliable, easy-to-operate Arduino based smart home security system solution.

In summary, a lot of research has been conducted around home security systems, using both Raspberry Pi and Arduino along with various sensors, webcams, GSM modules etc. The drawbacks in these home security systems are that they do not consider how the homeowner receives SMS alerts of intrusion in a network problem. The main area of focus with all previous work appears to be proposing an economical, affordable and easy to install system. Many applications have been developed along with these security systems on smartphones that can send warning to the user, however, they cannot view or monitor their residence through this app due to the absence of any sort of camera type alarm system. Thus, the main aim in this project is to convert existing homes into smart homes at a low cost, with an easy to use system and simple design to monitor their home remotely so beginner to advanced level Internet users can use this system with ease. Not only would this system conduct face recognition, but it would also focus on reducing the number of alerts homeowners receive unnecessarily in regards to motion being detected by their own households.

## Methodology

This Arduino Smart Home Security System is based on Agile Development as shown in Figure 5. This model has 6 phases: Planning, Analysis, Design, Implementation, Testing and Maintenance. Agile methodologies key value is the ability to facilitate flexibility, this is in particular very necessary in this project as with an iterative approach many developments can later be implemented to produce a secure, standard and low-cost smart security system for the home using Arduino.



*Figure 5 : Agile Methodology*

Agile methodologies tend to minimise risk during the execution process by developing the project in iterations. Doing so, treats each iteration like a small project of the final project all whilst including the necessary stages of development to implement new features and functionalities (Hidalgo, 2019).

## Analysis

The key objectives of this security system revolve around the research, design, implementation and testing of this smart home security system using Wi-Fi technology, and evaluation of the system in terms of its ability to perform and meeting its aims. In order to achieve this, the following key components are outlined:

- An Android mobile application to monitor the home.
- A Wi-Fi module (ESP32-CAM) facilitating the connection and communication between the mobile application and the actual device.
- A programmable board/microcontroller (Arduino UNO) acting as the brain of the system and executing key functions.
- An internal security system which protects the device from unauthorised users.

Detection of movement: The PIR sensor chosen for this system is responsible for movement detection. The sensors accuracy is high in range of 5 to 7 meters.

Monitoring/Surveillance and Face detection and recognition: The ESP32-CAM has video streaming capabilities that allows the user to not only view/monitor through the camera; but it can also conduct face recognition and detection too. The video stream from the ESP32-CAM can be watched in the user-interface.

Communication: The system is required to communicate to users, so user-interface can notify the user with an alert notification when/if an intruder is detected.

Interface: The user interface is a medium between the user and the system, permitting users to watch a live video stream in the application only by logging in, in order to provide a secure system.

## Hardware Requirements

The hardware components of this proposed smart home security system involve Arduino UNO board, an ESP32-CAM module, SD card and PIR sensor which are outlined in this section. The Android mobile application which can run on a smartphone, tablet or an android compatible computer is also a requirement. The commands are sent to the controlling unit (Arduino UNO) of the smart device via Wi-Fi communication. The Arduino UNO is an embedded microcontroller which acts as the brain of the system. The connection between these components does not require soldering but requires sufficient jumper wires and a breadboard. Also, to provide power to the system, a power source is necessary which is facilitated with the use of an USB A male to male cable.

### Arduino Uno (ATmega 328)

Arduino Uno is a microcontroller board that has 14 digital input/output pins, 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. This contains everything needed to support the microcontroller; can simply connect it to a computer with a USB cable/power it with an AC-to-DC adapter/battery to get started. In addition, the board is inexpensive, freeware and has very active developer's community.

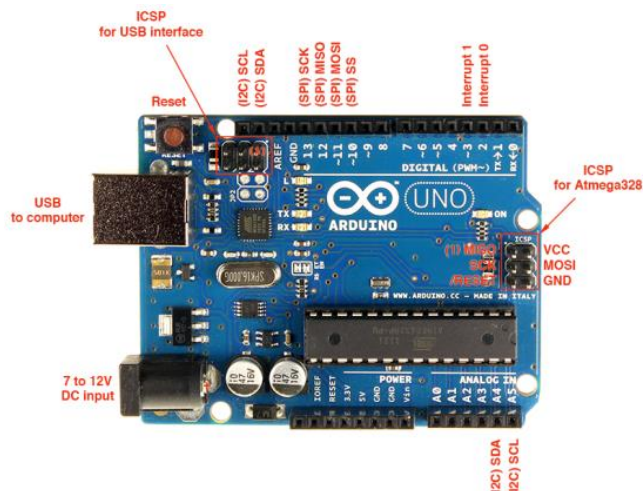


Figure 6: Arduino Uno Rev3

### HC-SR501 Passive Infrared Sensor (PIR)

PIR sensors also known as “Passive Infrared”, “Pyroelectric”, or “IR motion” sensors sense motion whether a human has moved in or out of the sensors range. PIR sensors accuracy is high in range of 5 to 7 meters. It is small, inexpensive, low-power, easy to use and does not wear out. These factors combined made it a suitable choice for this security system. Furthermore, for all these reasons they are commonly found in appliances/gadgets used in homes, schools, businesses etc. (Suresh et al, (2016). This sensor has three output pins Vcc, Output and Ground as shown in Figure 7.

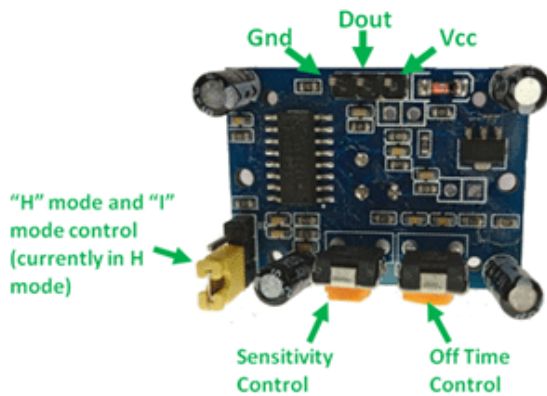


Figure 7: PIR Sensor Pin Diagram

The PIR sensor consists of pyroelectric materials and is suitable for working at frequencies from 0.1 to 100 Hz. As the PIR sensor responds to the rate of change of temperature, it can measure pulse type radiation power (e.g., the change rate of human thermal infrared radiation) (Yan J, 2018). Due to these characteristics of the PIR, it can be used for human detection and therefore relevant to use in this AHSS development. Notably, the sensitivity can be set using the “sensitivity control” potentiometer as shown in the figure above and in Repeatable(H) mode the output pin Dout will go high (3.3V) when a person is detected and goes low after a particular time, time is essentially set by “Off time control” potentiometer.

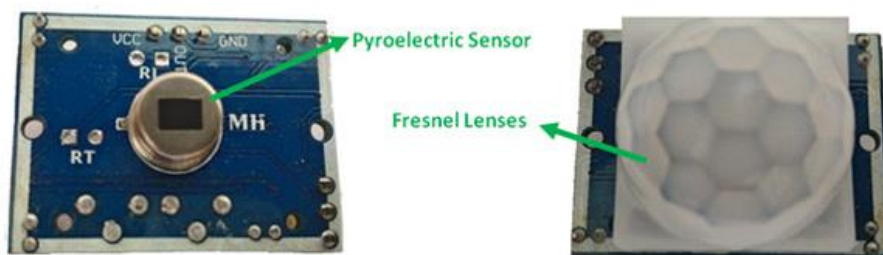


Figure 8 : HC-SR501 PIR Sensor

The Fresnel lens as shown above in Figure 8 condenses light, providing a larger range of IR to the sensor.

### ESP32-CAM Module

ESP32-CAM is a low-cost development board with a Wi-Fi camera (an integrated video camera and microSD card socket). It allows creating IP camera projects for video streaming with different resolutions. ESP32-CAM has a built in PCB antenna. It is perfect for IoT devices requiring a camera with advanced functions like image tracking and recognition.



Figure 9: ESP32-CAM

According to Random Nerd Tutorials (2021), the ESP32-CAM has the following impressive features that contributed to the decision to use this :

- The smallest 802.11b/g/n Wi-Fi BT SoC module
- Low power 32-bit CPU, can also serve the application processor
- Up to 160MHz clock speed, summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Support OV2640 and OV7670 cameras, built-in flash lamp
- Support image WiFi upload
- Support TF card
- Supports multiple sleep modes
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

## Software Requirements

The entire functioning of this Arduino Smart Home Security system uses the Arduino UNO developed through the Arduino IDE which is programmed in C/C++.

The android application set to be developed will use Android Studio 4.1.2 software this uses Java programming language. Android Studio is used to build apps for Android phones, tablets, etc. It was primarily chosen for this project for these reasons as well how the structured code modules facilitates the breakdown of projects into units of functionality that can be independently built, test, and debugged. This AHSS application will facilitate users to control the home security system as well as monitor their home through the camera. The user interface design will aim to keep it an easy to use system where they can monitor their home remotely so beginner to advanced level Internet users can use this system with ease.

The ESP32-CAM has the capability to acquire video and images and using this capability the users of the AHSS will be able to monitor their homes.

### ESP-WHO Development Framework

The ESP32-CAM has support for machine learning with the ESP-WHO framework. With ESP-WHO, you can easily build up face detection- and recognition-featured applications.

ESP-WHO is a face detection and recognition development framework designed for AIoT applications.

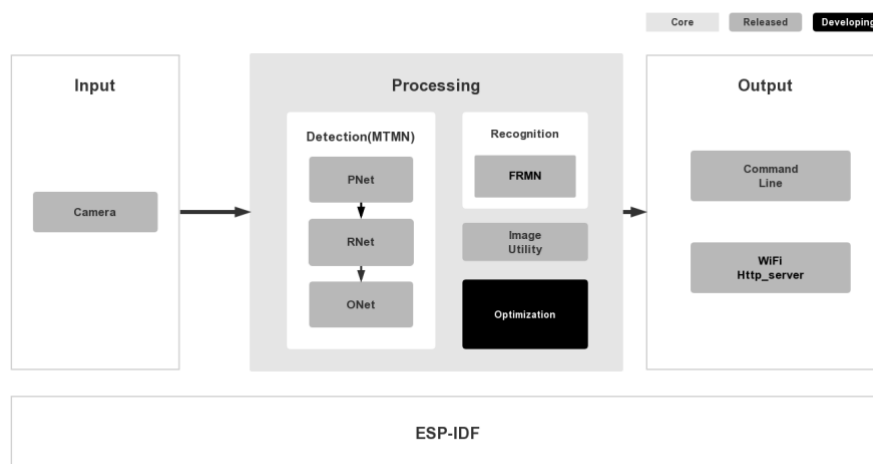


Figure 10: ESP-WHO Development Framework

Face detection is implemented using MTCNN and MobileNet it returns the position of any faces in the image if present. Whilst face recognition is implemented with MobileFace (Allan, 2018).

Firebase Cloud Messaging (FCM) provides a dependable connection between server and devices that allows users to deliver and receive messages and notifications on iOS, Android, and the web at no cost (Firebase, 2020) . In this case, FCM will be needed to send notifications from the Arduino based on the face detection and recognition the ESP32-CAM is conducting and convey a notification to the user's mobile device to alert them of an intruder.

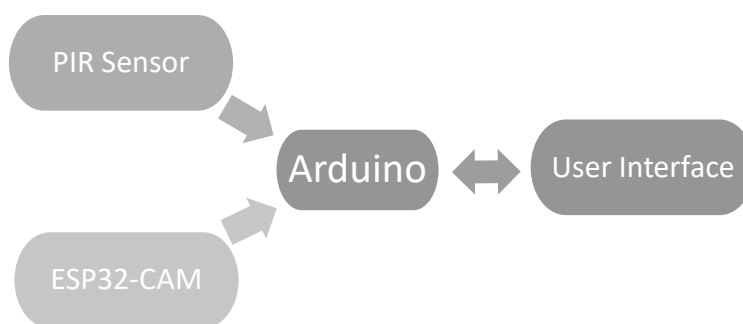
Firebase has the following requirements (Firebase, 2021):

- Install/update Android Studio to its latest version.
- Targets API level 16 (Jelly Bean) or later
- Uses Gradle 4.1 or later

- Uses Jetpack (AndroidX), which includes meeting these version requirements:
- `com.android.tools.build:gradle v3.2.1` or later
- `compileSdkVersion 28` or later
- Set up a physical device or use an emulator to run your app.
- Require the device or emulator to have Google Play services installed.
- Firebase requires a Google account sign in.

## System Design

### Block Diagram



*Figure 11 : Arduino Smart Home Security System Block Design*

The block diagram in Figure 11 depicts the relationship between the components. The PIR sensor and ESP32-CAM essentially provide information to the Arduino Uno. Then through the app, the Arduino communicates with the users, as users will be able to monitor their home whenever they want. The left side of the diagram symbolises the sensors connected to the Arduino Uno and right side represents the user-interface of the android application. The Arduino is the the fundamental component of the system to which the two sensors; the PIR sensor and camera are connected. As it is in charge of the input and output flow in the system.

If the PIR sensor senses any motion and the ESP32-CAM detects an intruder it will trigger an alert to the user that there is an unusual activity in the home. Thus, the Arduino will process data to the application regarding the alert. The user can then monitor the video streaming through the application to observe the unusual activity.

## Flowchart

Flowchart shows the way in which the system will work.

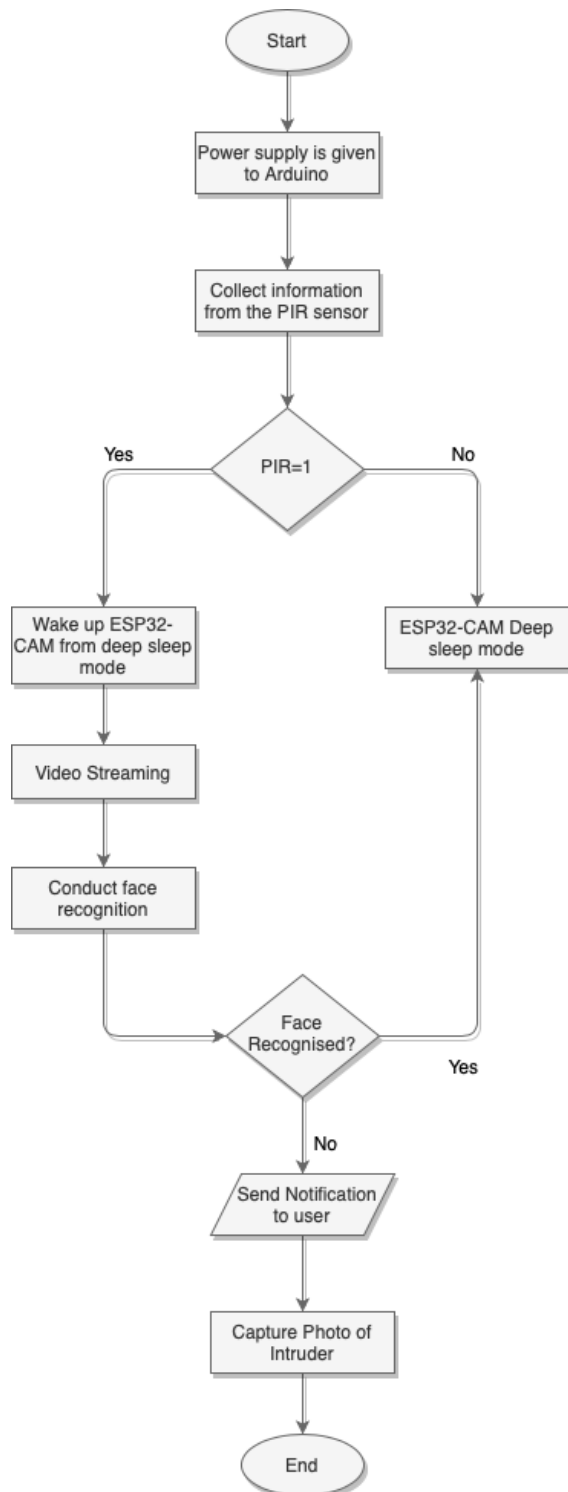


Figure 12: System Flowchart

Figure 12 demonstrates the process in which the system functions. The process begins by supplying power to the Arduino and other components. This then activates the PIR sensor, which can then detect motion. The information from this is passed and then executes in two ways based on the PIR sensor's detection:

If the PIR sensor = 1, meaning motion is detected, the ESP32 camera will be woken up from sleep mode. Waking up from deep sleep mode will result in the ESP32-CAM to be running and thus live video streaming. With the features the ESP32-CAM has it can conduct face detection and recognition and therefore an alert will be sent to the users on the mobile application if the face is not recognised as well as capture a photo of the intruder to save onto the SD card/user's mobile device. However, if the face is recognised, meaning a household member has been detected then the ESP32-CAM will return back to deep sleep mode.

If the PIR sensor = 0, meaning there is no motion detected. Meaning the home is safe from intruders, the ESP32-CAM will remain in deep sleep mode. If the users want to watch the live video streaming, they can do so through the mobile application.

The mobile application function is illustrated below.



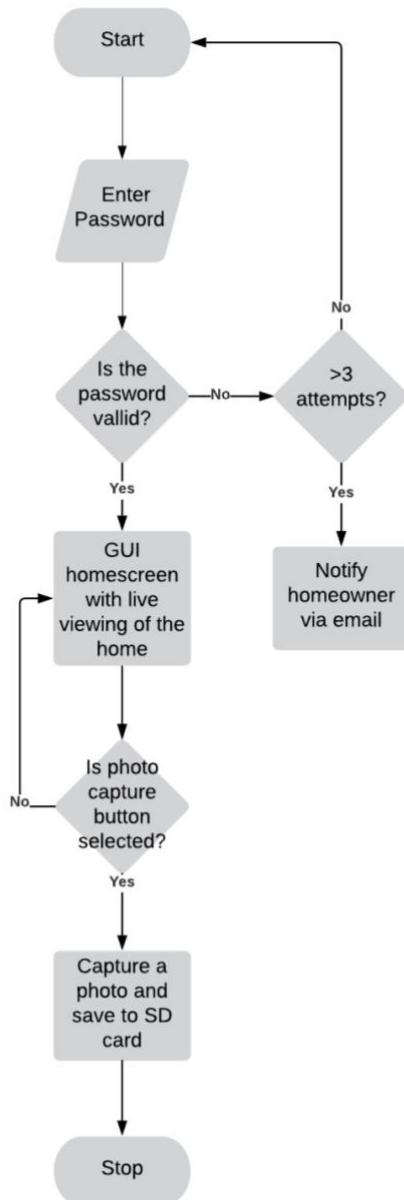


Figure 13 : Mobile Application Flowchart

Figure 13 flowchart illustrates the functionality of the proposed mobile application. It begins with the user facing the login screen, where they enter their password and three failed attempts notifies the homeowner of an intruder, making them aware of someone trying to gain unauthorised access of the home security system. If, however, the password is valid, they enter into the main page. This is the home page that contains the live video stream of the home, it also contains a capture button for the user to capture photos from the live video stream. Captured photos are then saved to the SD card.

### Schematic Diagram

The idea here is the Arduino UNO board will be connected to a computer via USB, where it will connect to the Arduino IDE. With correct programming development of Arduino code in the IDE, when uploading it on to the microcontroller, it should execute the code and be interacting with the other chosen components; the PIR sensor and ESP32-CAM. In order to even upload any code onto

the board, sufficient connections and wiring will need to be done. Below are the schematic diagrams of the connections that are required to give functionality to the components.

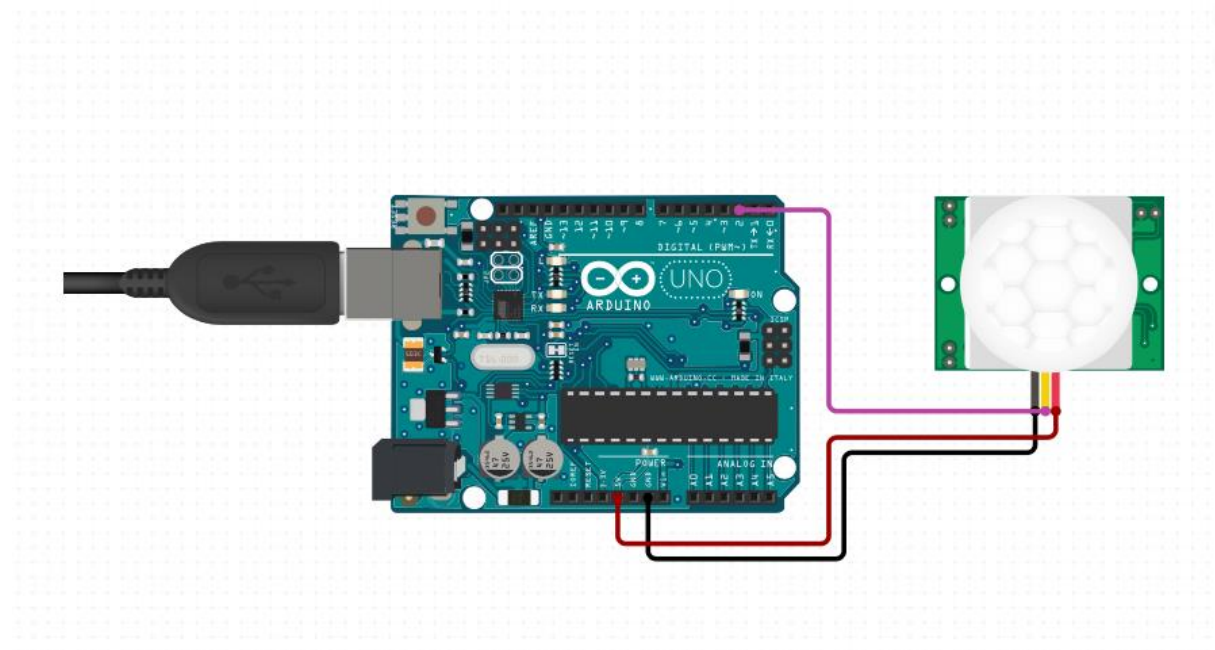


Figure 14: Arduino UNO and PIR Sensor Schematic Diagram

The schematic diagram shown in Figure 14 illustrates the connections between the Arduino UNO with PIR sensor.

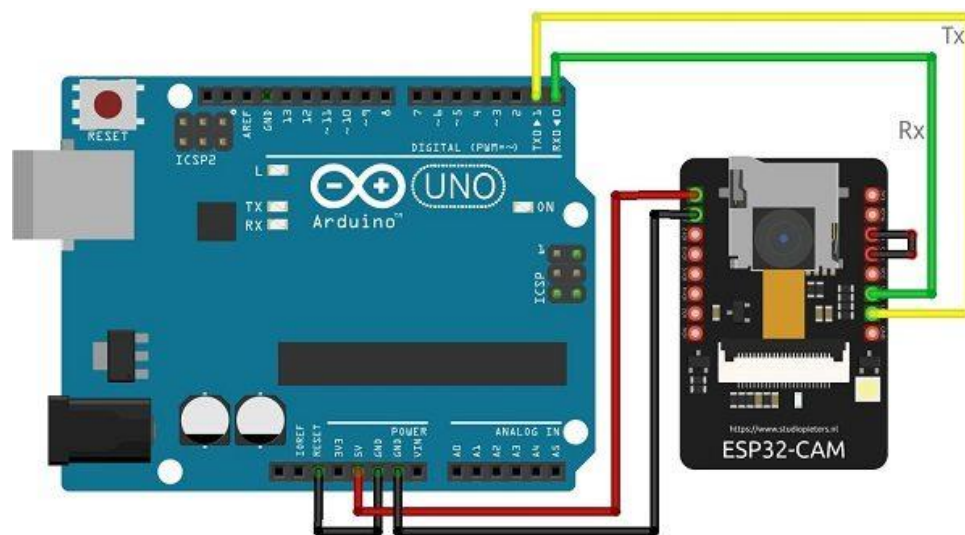


Figure 15: Arduino UNO and ESP32-CAM Schematic Diagram

The schematic diagram presented in Figure 15 shows the connection between Arduino Uno and the ESP-32 CAM.

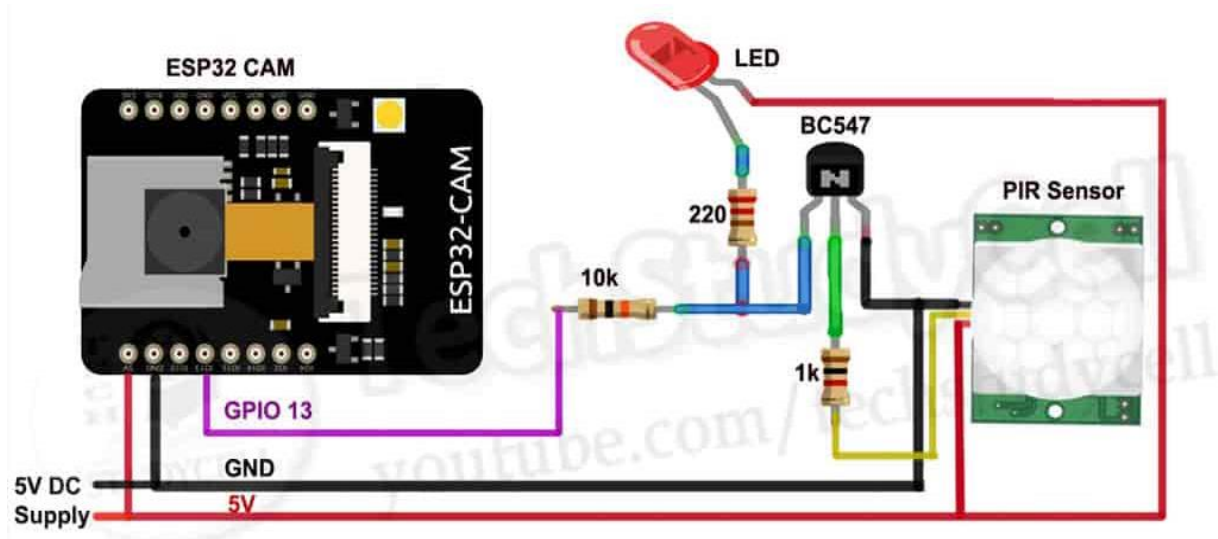


Figure 16: ESP32-CAM and PIR Sensor Schematic Diagram

This circuit diagram in Figure 16 compromises of ESP32-CAM and PIR motion sensor and other extra components such as resistors and LEDs etc. The connections this diagram presents will be followed whilst also using the Arduino UNO to build this AHS System.

## Implementation

The innovative aspect of this project involves the ESP32-CAM. It will recognise the individuals that reside in that home but for any motion that is detected and an unrecognised face is detected - their photo will be captured and it will alert the home owner and store the photo of the intruder for relevant security measures to be conducted. Purpose for this was so that the number of notifications the user receives are kept minimal - that are perhaps just family members and therefore not a security threat. Illustrations of the prototype of this system is presented in this section. Showing the design of the system, the connection between Arduino and other components. Followed by Server-side implementation and then the client-side implementation and a show of the final product.

### Prototype

Firstly, connected the ESP32-CAM to the Arduino UNO to upload code from Arduino IDE. Done so, by doing the following connections:

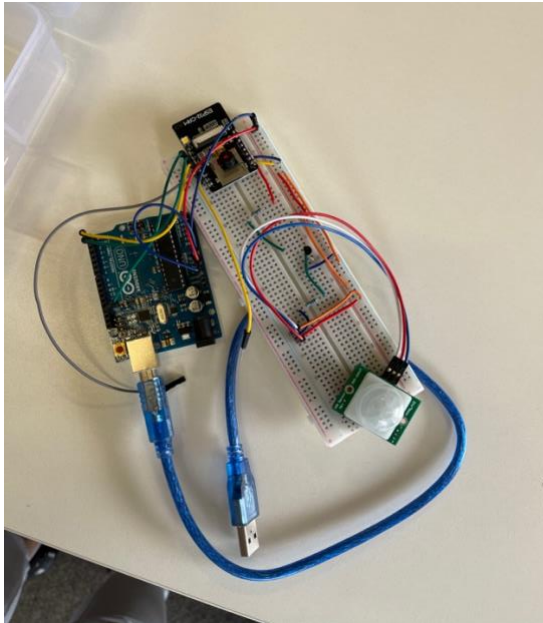
- Connect Arduino 5 volt to ESP32-CAM 5 volt
- Arduino GND to GND
- Arduino RX to ESP32-CAM board RX & TX to TX
- Arduino reset pin to GND
- ESP32-CAM GPIO-0 to GND



Figure 17: ESP32-CAM pins (Seeedstudio.com, 2018)

Connection pins of PIR sensor with the Arduino.

1. An Vcc pin of PIR sensor connected to the 5V pin of the Arduino.
2. A ground pin of PIR sensor connected to the ground pin of the Arduino.
3. The Dout (digital out) pin of PIR sensor connected to the digital 2nd pin of the Arduino.



*Figure 18: AHSS Prototype*

Figure 18 presents the prototype of the AHS system. It shows the connection between the components. It can be easily installed in houses, offices or anywhere that requires a security system.

The essential conditions for installing the prototype are:

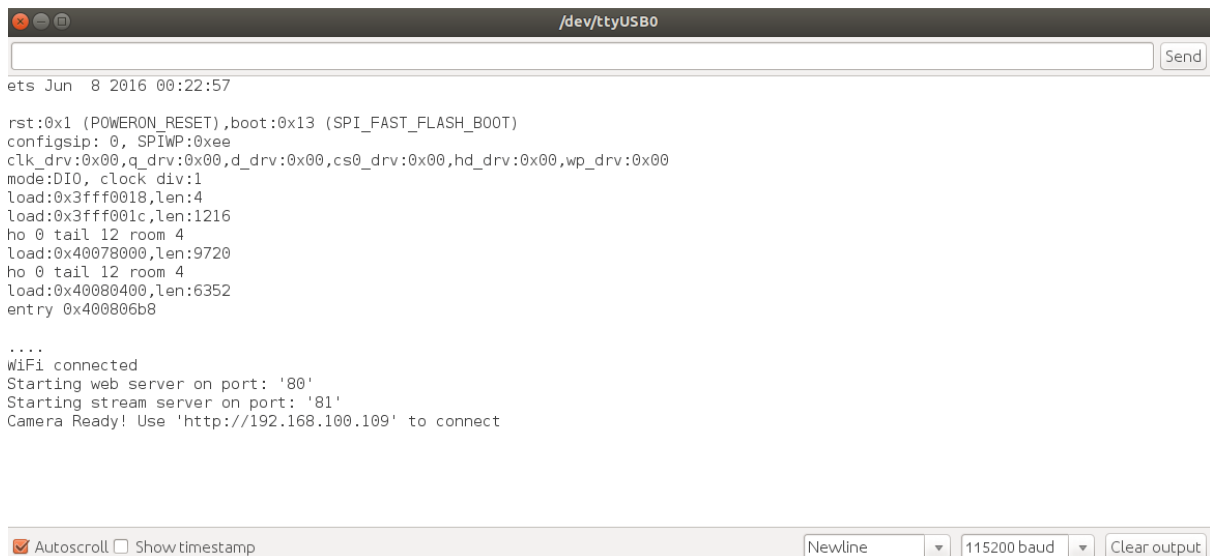
- The user must have Internet access.
- The prototype requires power supply

### Server-side Implementation

After inputting and selecting the relevant settings on the Arduino IDE, following the steps from Random Nerd Tutorial, 2021.

Then it is ready for uploading sketch, so from File > Example > ESP32 > Camera > Camera web server it is possible to upload the sample sketch. Here Wi-Fi credentials are inputted, so name and password as well as choose what camera model in use. In this case, it is AI thinker.

When the sketch has uploaded it is required to disconnect ESP32-CAM IO0 and GND and press reset. In the serial monitor the following message is then shown (Figure 19).



```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8

....
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.100.109' to connect
```

Autoscroll ☐ Show timestamp Newline 115200 baud Clear output

Figure 19: Serial Monitor Output

Figure 19 exemplifies the initialization information outputted in the serial monitor, followed by a statement saying that the board has connected to the network and has obtained an IP address. The IP address will be in the form of a URL. This IP address can then be used to access the web server via a browser to load the ESP32-CAM settings where a live stream can be initiated. This is readily available to do by anyone using the ESP32-CAM – they can stream videos/images over Wi-Fi as well as perform face recognition and detection, all via a demo program included with the drivers.

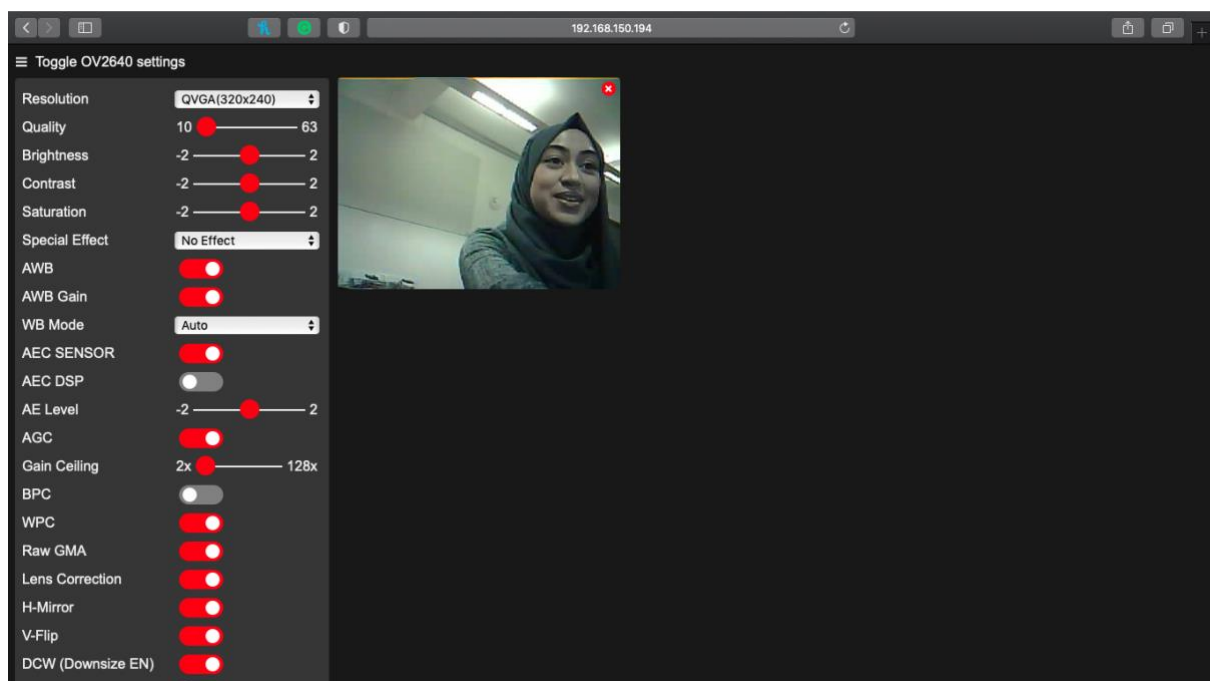


Figure 20: ESP32-CAM Video Streaming Web Server

Here in Figure 20 is an example of what the user sees when accessing the web server via a browser it is a webpage for the camera, complete with a number of controls. There is a Start Stream button that starts to stream video as well as various Camera settings that include allowing the user to change the size and frame rate of the video using the drop-down at the top of the screen.

```

//PIR Definitions
int led = 13;
const int sensor = 13; //Pin of PIR
int state = LOW; //No Motion Detected
int val = 0; //Sensor of status

void startCameraServer();

void setup() {
  pinMode(sensor, INPUT_PULLUP); //Take input from PIR
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  void loop() {
    val = digitalRead(sensor); // read sensor value
    if (val == HIGH) { // check if the sensor is HIGH
      //digitalWrite(led, HIGH); // turn LED ON
      delay(500); // delay 100 milliseconds

      if (state == LOW) {
        Serial.println("Motion stopped!");
        state = HIGH; // update variable state to HIGH
      }
    }
    else {
      //digitalWrite(led, LOW); // turn LED OFF
      delay(500); // delay 200 milliseconds

      if (state == HIGH){
        Serial.println("Motion detected!");
        state = LOW; // update variable state to LOW
      }
    }
  }
}

```

Figure 21: PIR Sensor Code - esp32cam.ino

Connecting the PIR sensor to the Arduino Uno as well was very simple, as shown in Figure 21 above, the PIR sensor code (Adafruit Learning System, 2016) was used to merge into the ESP32-CAM code. To establish connection the PIR is powered with 5V and ground to ground connected. Thenceforward, connected the output to a digital pin as shown in Figure 21 this is Pin 13. This essentially just keeps track of whether the input to pin 13 is high or low. As well as tracking the *state* of the pin, so that outputs the message when motion has started/detected and stopped.

### Client-side Implementation

This section is set to develop an android application which the user is able to easily access. The aim of the application is to provide a software that the average user can operate. The Login screen is first created and then the Home screen, the last section will discuss Firebase Cloud Messaging that was used in the development of this application.

## Login Screen Development

```
package com.nish.ahs;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.view.View;
import android.webkit.WebView;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.messaging.FirebaseMessagingService;

public class MainActivity extends AppCompatActivity {

    private EditText eName;
    private EditText ePassword;
    private Button eLogin;
    private TextView eAttemptsInfo;

    private int counter=3;

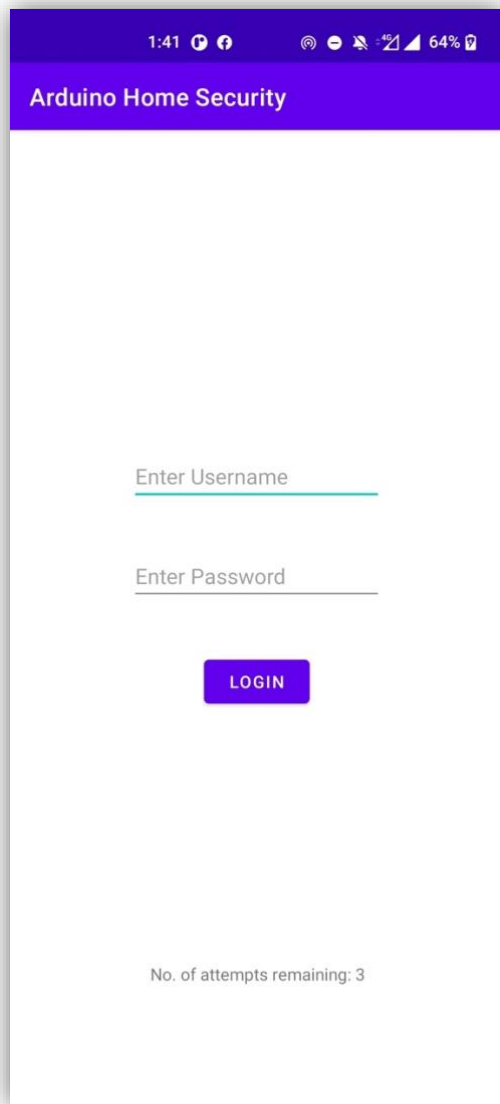
    class Credentials {
        private String Username = "Admin";
        private String Password = "2323";
    }

    Boolean isValid = false;
```

Figure 22: MainActivity.java I

In the Main.Activity.java file, as shown in Figure 22, contains the main functionality for the login system. The Name, Password, Login button and Attempts Info variables were created. When these UI elements are defined, it needs to be bind to the xml layout. There is a particular function called findViewById, so with the variable names that were defined the following is done as shown in Figure 23 to connect the xml layout with the variables created.





*Figure 23: AHSS Application Login Screen*

The Login screen contains EditText which is the standard text entry widget in Android apps. It is used to enter text into an app and in this case is used as shown in Figure 23 for Username and Password to be entered by the user. In Figure 24 id is an attribute used to distinctively identify the Username and Password text. The Login button is located just below where credentials are expected to be inputted. At the bottom of the screen 'No. of attempts remaining:3' is shown as a TextView – an UI element that illustrates text to the user.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main); //set to load the content of activity_main
    Intent intentBackgroundService = new Intent(this, MyFirebaseMessagingService.class);
    startService(intentBackgroundService);
    eName = findViewById(R.id.etName); //connects the xml layout with the variables
    ePassword = findViewById(R.id.etPassword);
    eLogin = findViewById(R.id.btnLogin);
    eAttemptsInfo = findViewById(R.id.tvAttemptsInfo);

    eLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view){

            String inputName = eName.getText().toString();
            String inputPassword = ePassword.getText().toString();

            if(inputName.isEmpty() || inputPassword.isEmpty())
            {
                Toast.makeText(MainActivity.this, "Enter correct username and password",
                Toast.LENGTH_SHORT).show();
            }else{

                isValid = validate(inputName, inputPassword);

                if(!isValid){

                    counter--;

                    eAttemptsInfo.setText("No. of attempts remaining:" + counter);

                    if (counter == 0){
                        eLogin.setEnabled(false); //disables the login button
                        Toast.makeText(MainActivity.this, "You have used all your attempts try
again later", Toast.LENGTH_LONG).show();
                    }
                    else{
                        Toast.makeText(MainActivity.this, "Incorrect username/password",
                        Toast.LENGTH_SHORT).show();
                    }
                }

                else {

                    Toast.makeText(MainActivity.this, "Login success", Toast.LENGTH_SHORT).show(

                    //next activity
                    startActivity(new Intent(MainActivity.this, MainScreen.class));
                }

            }

        }

    });
}

```

Figure 24: MainActivity.java II

Figure 24 shows String input name is used to get an input from the user, this is then converted into a string. The same goes for the password, a new variable string is used to collect the users input of the password, so then to check if the user has inputted the correct credentials, it checks if the user has entered anything at all by using isEmpty() and || or field used, so an error message is outputted using Toast to display simple messages for any events that occur, in this case outputting "Enter correct username and password". See Figure 26 and 27.

```

private boolean validate(String Username, String Password)
{
    /* Get the object of Credentials class */
    Credentials credentials = new Credentials();

    /* Check the credentials */
    if(Username.equals(credentials.Username) && Password.equals(credentials.Password))
    {
        return true;
    }

    return false;
}
}

```

Figure 25: MainActivity.java III

An else statement is used to validate the credentials. The credentials were set as shown in Figure 22. Boolean is used to check if the credentials are correct ie. true or false. This private boolean function is named as 'validate', it takes two inputs - a string name and a password. This compares with the strings that were defined. So, if `name.equals(Username)&&((Password))` will conduct the string match and validate the credentials. In Figure 25 a boolean is created as Boolean **isValid = false**; and then `isValid` equal to the validate function, `isValid = validate(inputName, inputPassword);`, so now having a boolean that is false or true depending on the credentials input.

In case that it is false, `if(!isValid){` (exclamation mark used to invert), then it is expected to decrement the counter and disable the button. Thus, for that a counter is required, in this case, private int `counter=3`. If it is invalid then it decrements by 1 so `counter--`;

It should then display the message that the credentials inputted were incorrect, and another toast message is required with the message stating it is an incorrect username/password. Furthermore, ensured to program that the counter value is 0, if so, then the login button is disabled – linked to variable `eLogin` – `setEnabled` to false and output message

`eAttemptsInfo.setText("No. of attempts remaining:" + counter);`

The text of `eAttemptsInfo` can be set to the number of attempts currently available at that moment in time as evidenced in Figure 26.

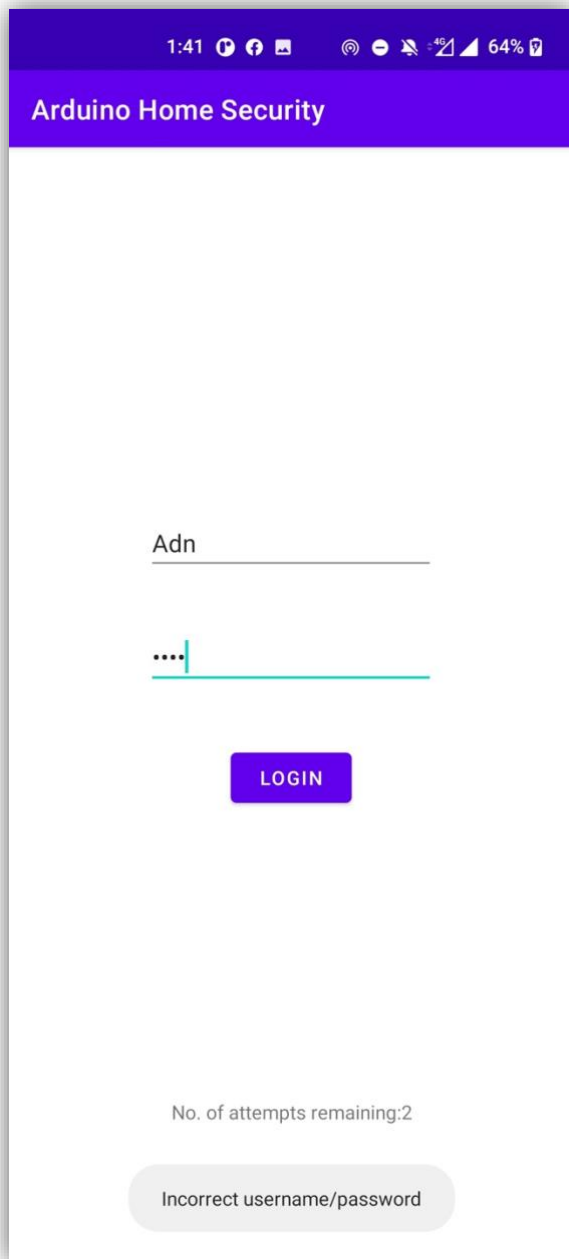


Figure 26: Incorrect Login Credentials

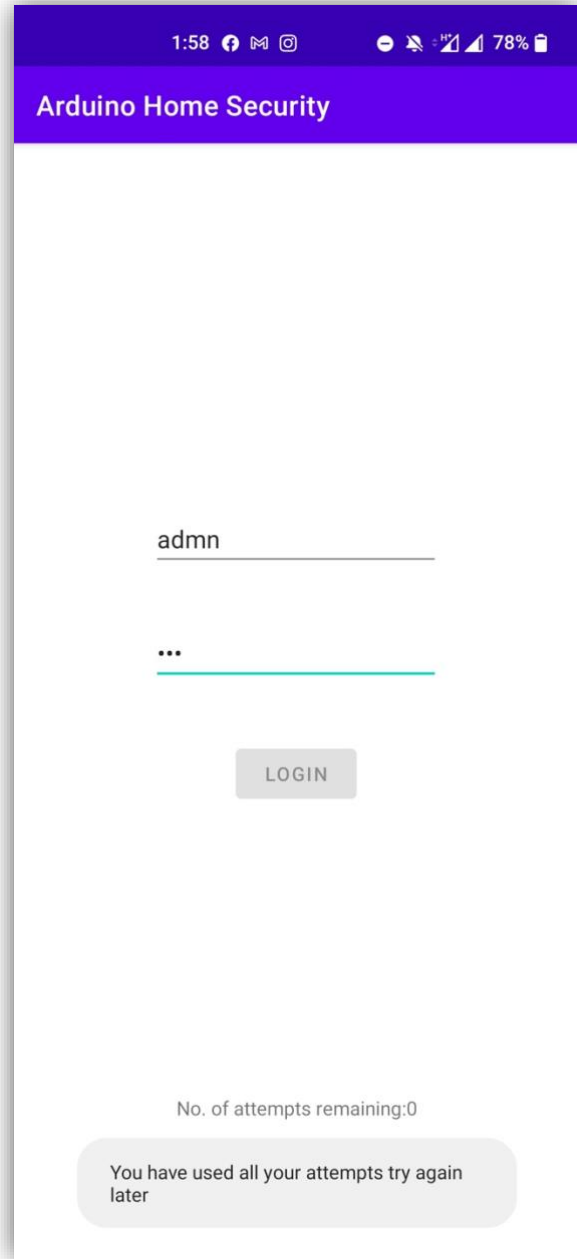


Figure 27: Login Button Disabled

As shown in Figure 25 the app takes the user input and compares it with a string and on correct login it lets the user enter the app else when the user enters incorrect login credentials a Toast message appears on the bottom of the screen to say "Incorrect username/password". This notifies the user of exactly that, that their credentials inputted are wrong and access to the app is denied. The no. of attempts remaining also reduce by 1 and as evident in Figure 26 from 3 the number of attempts remaining has reduced to 2. If, however, the user runs out of attempts (Figure 27) the login button is disabled as well as a Toast message outputting that 'You have used all of your attempts try again later'.

If, however, the correct login credentials are entered, Else statement message outputs the message of login being successful and leads the user onto the next activity (Figure 28 and 29).

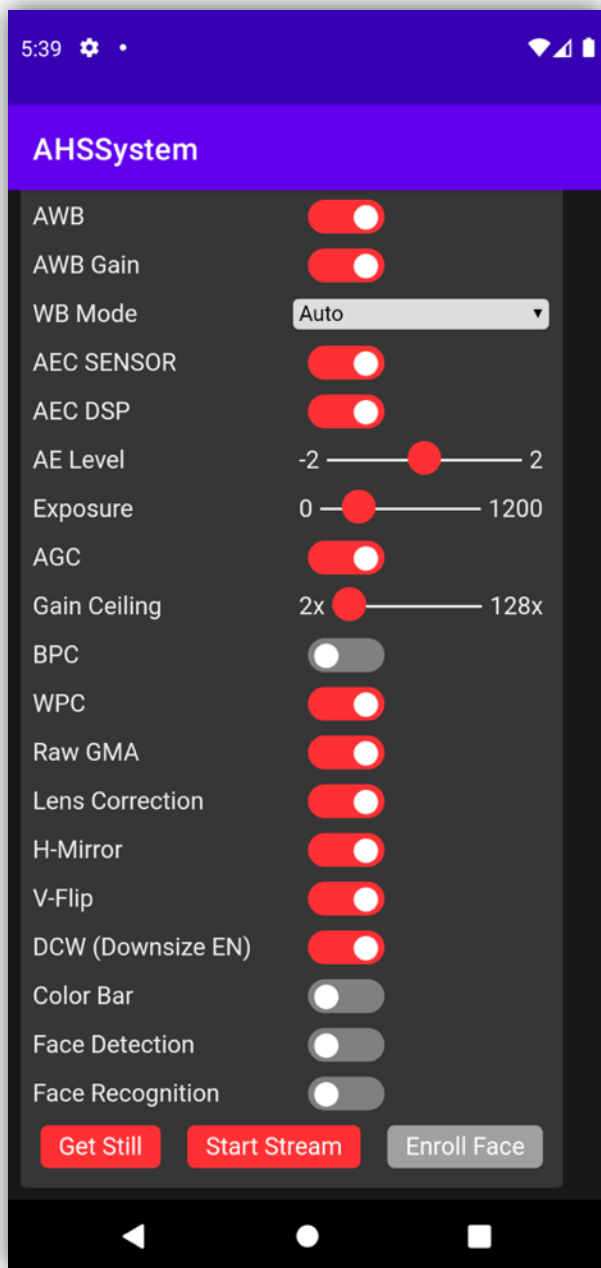


Figure 28: Initial Defective Home Screen

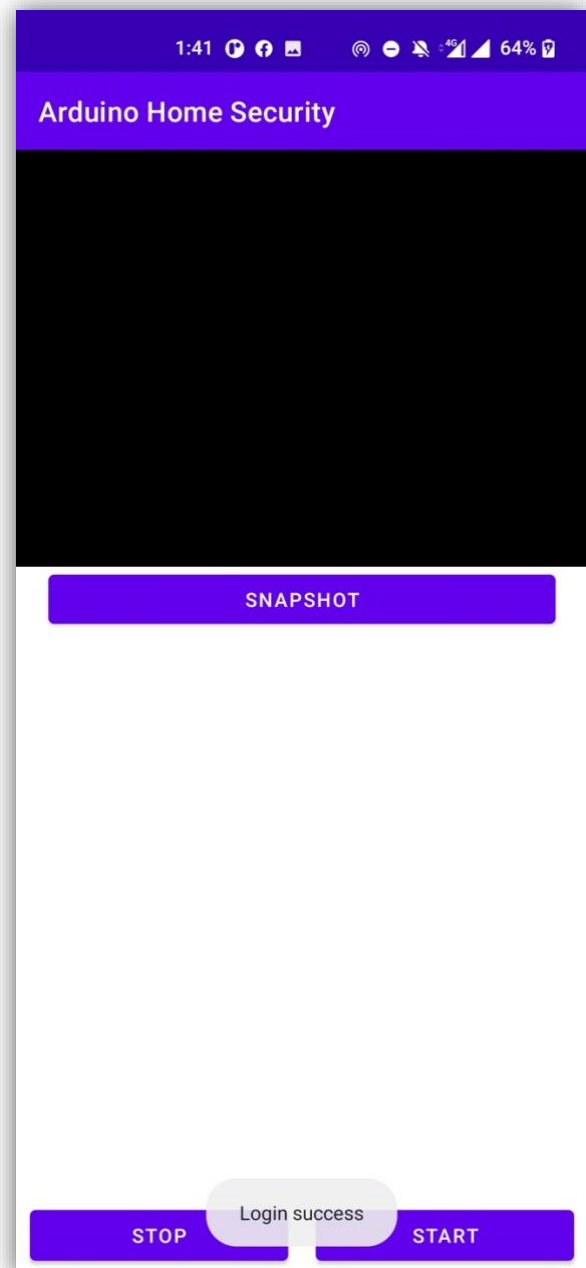


Figure 29: Login Success - Access Granted

At this stage successful login takes the user to the next screen, here a WebView of the ESP32 camera stream is expected to show using the ESP32-CAM IP address, it currently shows the webpage but unable to view the camera. As shown in Figure 28, all of the settings for the video stream are on display, however, the Start stream button does not work.

Therefore, instead of loading the whole webpage and these settings for the camera being on display, decided to instead let the mobile application load the stream URL directly. This would improve the overall user interface looks and aesthetics for the app as well as provide the user with what they want instead of having to go through unnecessary camera settings Figure 28 and 29.

Doing so, led to further issues such as the initial ESP32 web server used – had a max header limit which the Android app did not facilitate for at the beginning due to the package field being too long (HTTP 400 error). Rectified this by making the package smaller and Figure 30 shows the code for the application.

```
public class MainScreen extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_page);

        //IDs
        WebView view = (WebView) this.findViewById(R.id.webView);
        String url = "http://192.168.29.194:81/stream"; //Stream Address
        String blankpage = "http://www.e-try.com/black.htm"; //Address for blank page

        String token = FirebaseInstanceId.getInstance().getToken();
        TextView tokenTextView = (TextView) findViewById(R.id.textView2);
        tokenTextView.setText(token);

        //Stream
        view.getSettings().setLoadWithOverviewMode(true);
        view.getSettings().setUseWideViewPort(true);
        view.loadUrl(blankpage); //Shows a black screen
        //Start Stream Button
        Button start = (Button) findViewById(R.id.start);
        start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                view.loadUrl(url); //Loads the stream url
            }
        });

        //Stop Stream Button
        Button stop = (Button) findViewById(R.id.stop);
        stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                view.stopLoading(); //Stops loading the stream
                view.loadUrl("http://www.e-try.com/black.htm"); //Shows a black screen
            }
        });

        //Snapshot Button
        Button snapshot = (Button) findViewById(R.id.snapshot);
        snapshot.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String token = FirebaseInstanceId.getInstance().getToken();
                tokenTextView.setText(token);

                //Email token to ourselves
                Intent intent = new Intent(Intent.ACTION_SEND);
                intent.putExtra(Intent.EXTRA_EMAIL, new
String[]{"nishathumaira23@gmail.com"});
                intent.putExtra(Intent.EXTRA_SUBJECT, "AHS - Device ID");
                intent.putExtra(Intent.EXTRA_TEXT, token);
                intent.setType("message/rfc822");
                startActivity(Intent.createChooser(intent, "Select Email Sending App :"));
            }
        });
    }
}
```

Figure 30: MainScreen.java

In order to deliver a webpage as part of a client application, it is possible by using WebView. WebView class is an addition of Android's View class that displays web pages as a part of the activity layout. All that WebView does is essentially show a web page thus it does not include features of a

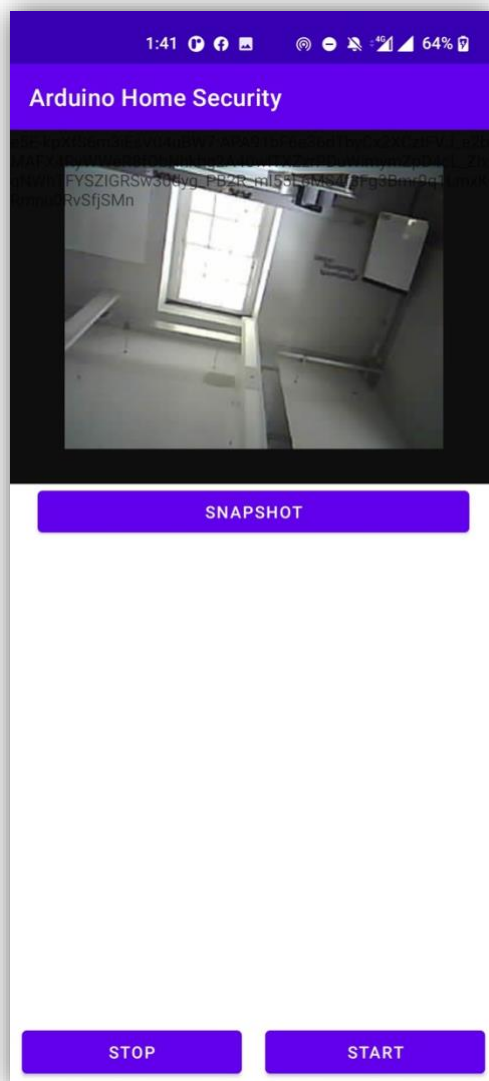
fully developed web browser, such as navigation controls or an address bar. Therefore, buttons will be needed to be created such as for Start Stream, Stop Stream and to capture a still.

```
String url = "http://192.168.29.194:81/stream";
```

Is the URL that this Android Home Security System will load through the stream button and a blank page address is defined for the user to see by default through `view.loadUrl(blankpage);`.

```
view.getSettings().setLoadWithOverviewMode(true);
```

Sets the WebView to load pages in overview mode, which essentially zooms out the content to fit on screen by width. As well as, `view.getSettings().setUseWideViewPort(true);` sets the WebView to support for the "viewport" HTML meta tag/use a wide viewport.



*Figure 31: Live Video Feed on App*

The Start stream button loads the stream URL and Stop stream button loads the URL for the blank page (Figure 31) as mentioned earlier, however, the Snapshot button is expected to capture a still of the live feed and save it to the user's device.

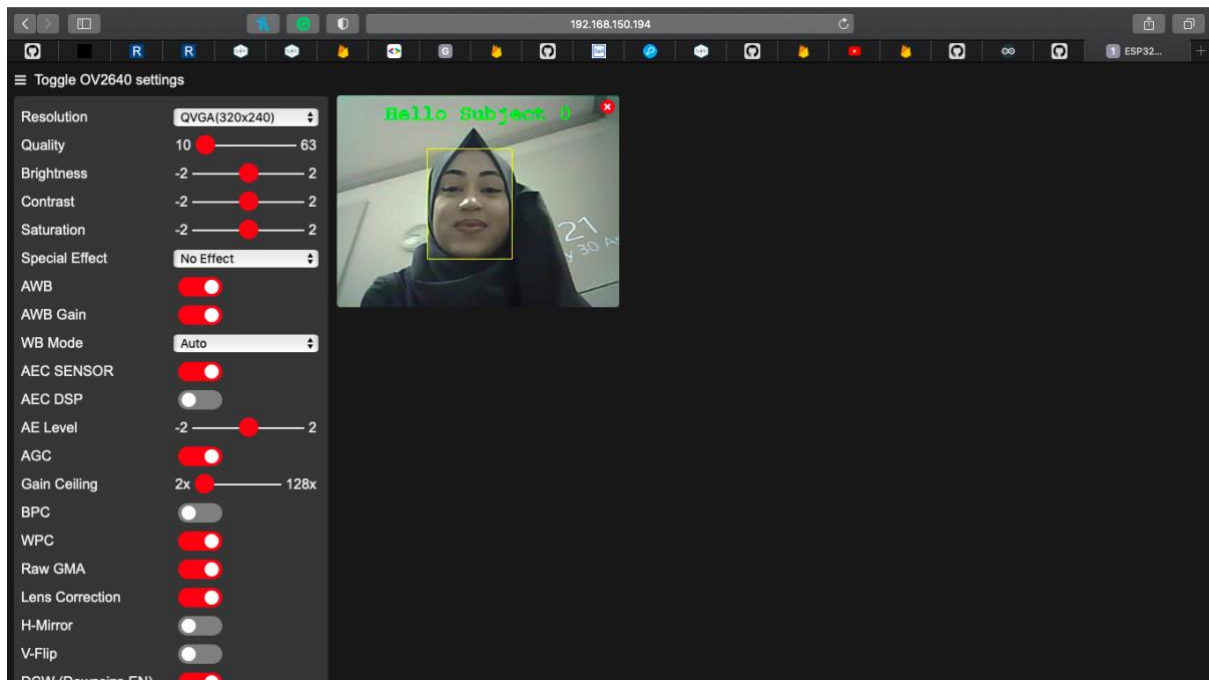


Figure 32: Recognised Face Detected

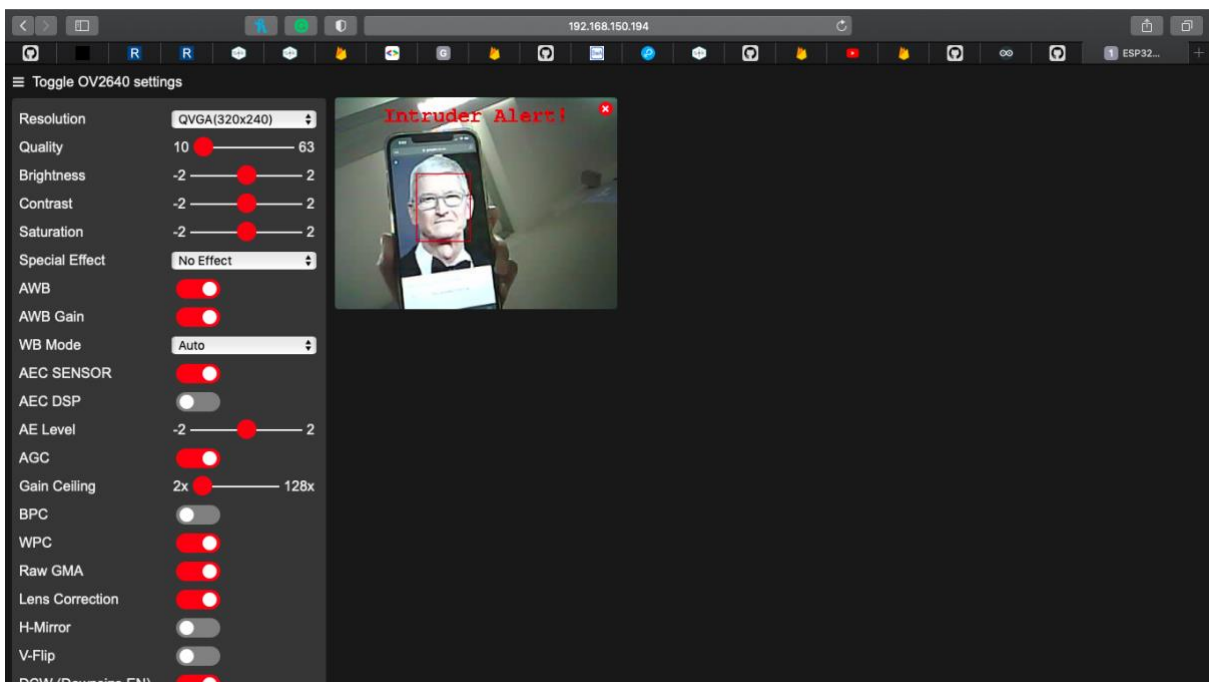


Figure 33: Unrecognised Face Highlights Intruder Alert

Figure 32 and 33 shows the stream from the ESP32-CAM on a webpage. The face detection and face recognition features are enabled and all that the user needs to do is enrol a new face. It will make several attempts to save the face. After enrolling a new user, it should detect the face later on like shown in Figure 32. However, when a new face is shown it will show an intruder alert message on the video stream as evidenced in Figure 33.

The program used to perform face recognition functions is the webserver camera library of the ESP32-CAM board, in `app_httpd.cpp`.



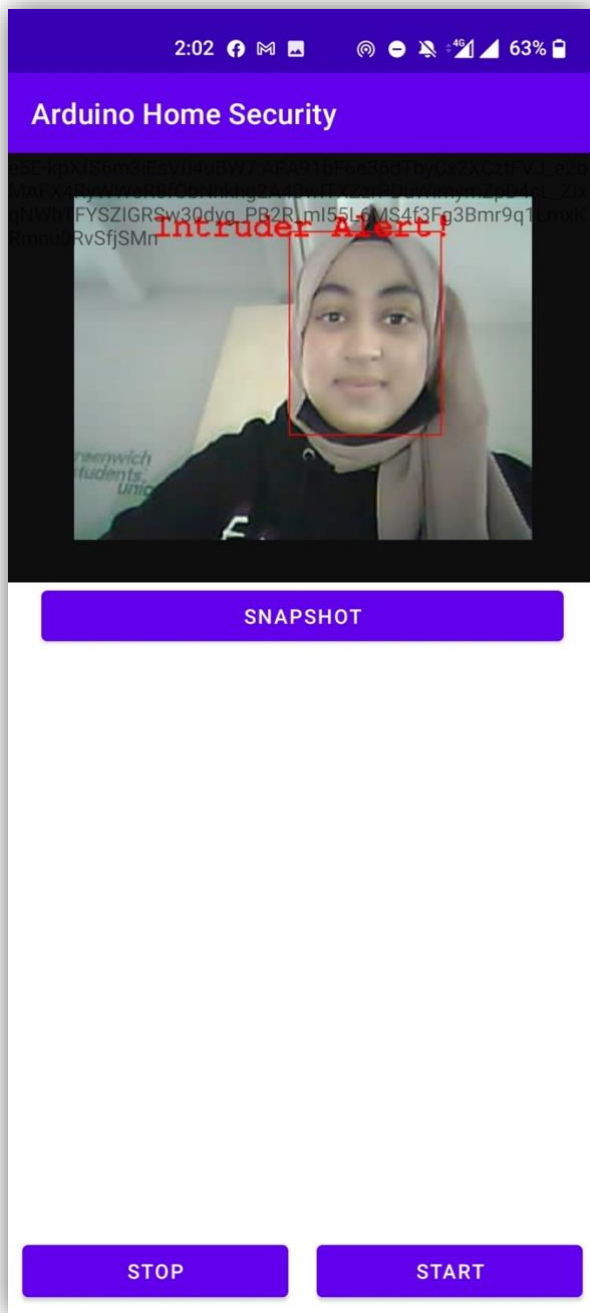


Figure 34: Face Recognition & Detection in App

Figure 34 displays the face detection and recognition feature working on the app. The ESP32-CAM is able to detect a human face and establish whether it is an intruder or not. To further achieve the aims of this project, it will be required to now establish notifications from the mobile application to be sent to the user if an intruder is detected.

With the existing android project for this AHS System and now Firebase developments incorporated, it is expected to produce an Android application that consists of push notification functionality. Following the series on Mobile Development for Arduino blog (Stablekernel.com, (2019)) the initial steps to set up a Firebase Project was conducted. Firstly, a project was set up on Firebase console – it then requires implementing the Cloud Messaging onto the AHS Application. The next step was

adding Firebase to the Android app and so essentially adding the Android configuration file to the AHSS app. This allows the app to talk to the Firebase console, then went on to integrating the Firebase SDK as well as the Google services Gradle plug-in into the AHSS application (in both build.gradle (ahs) and build.gradle (app)). Android Push Notifications + Firebase Cloud Messaging, 2019 video demonstrates the dependencies needed to be incorporated into the gradle file, then the Google services JSON file that Firebase provides has to be added in.

Initially when the app is launched it needs to communicate with the Firebase cloud messaging, Firebase is going to send a token – which is just a string to the app, then that token gets used and plugged it into the Firebase console, and using that token that Firebase console can now communicate with the app because it now knows how to identify the app through that token. So, now need to set up the app to be able to accept that token. Firstly, created a class that extends Firebase messaging service. Created a class called MyFirebaseMessagingService extending to FirebaseMessagingService.

```
public class MyFirebaseMessagingService extends FirebaseMessagingService {

    private static final int NOTIFICATION_ID = 1;

    public MyFirebaseMessagingService() {
        super();
    }

    @Override
    public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);
        Log.d("Msg", "Message received [" + remoteMessage + "]);

        Intent intent = new Intent(this, MainScreen.class);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 1410, intent,
        PendingIntent.FLAG_ONE_SHOT);

        NotificationCompat.Builder notificationBuilder = new
            NotificationCompat.Builder(this)
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle("Message")
                .setContentText(remoteMessage.getNotification().getBody())
                .setAutoCancel(true)
                .setContentIntent(pendingIntent);

        NotificationManager notificationManager = (NotificationManager)
        getSystemService(Context.NOTIFICATION_SERVICE);

        notificationManager.notify(NOTIFICATION_ID, notificationBuilder.build());
    }
}
```

Figure 35: MyFirebaseMessagingService.java

Figure 35 shows the MyFirebaseMessagingService class that was created, the service was registered into the manifest (Figure 36), doing so means whenever Firebase sends out a broadcast in the application to say it has got the token for the app and who is interested. Essentially, what this is doing is telling the system that our app is interested in listening for it. Establishing that by declaring a service in here called MyFirebaseService and with this intent filter for listening for messages. An intent is an abstract description of an operation to be performed. An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities, En.wikipedia.org. (2019).

```

<service
  android:name=".MyFirebaseMessagingService">
  <intent-filter>
    <action android:name="com.google.firebase.MESSAGING_EVENT"/>
  </intent-filter>
</service>

```

Figure 36: AndroidManifest.xml

Tested this by manually sending a push notification from Firebase console to the Android application, has to be in the background to receive messages.

Back on the android app development, had to set up that message has been received and send that message into our activity. To do that, going to implement onMessageReceived, from that it would pull out the remote message and then Log the message that was received and then pass that message to activity, by using a broadcast receiver. Also going to be sending an intent since a broadcast receiver will be used with the action sent message.

Once conducted, Mobizt- GitHub, (2020) implementation of Firebase into Arduino was incorporated. This was used to allow Arduino to send 'tokens' to Firebase which would then return notifications to the app.

```

#define WIFI_SSID "Nishat's OnePlus"
#define WIFI_PASSWORD "A1B2C3D4E5F6"
#define FIREBASE_HOST "ahss-99c86.firebaseio.com"
#define FIREBASE_AUTH "BM_Xu_vUL8M4E1jANuulMEGXPj1MRfUu88tioghLrNnhser005SuoXzLIZgV6A7g9-rs78-SjB1tjy85AhWdWME"
#define FIREBASE_FCM_SERVER_KEY "AAAAaTNTCSzg:APA91bETM6980FyhxvzrnpqWUMnd4uZ40nYS--m8aDDAFrQYkiFWjpkz_PbzBCQlrLZLS9diUoLbLZWJ64uoKZWglYXmiR08XS0AAdD2x7h5vJC1pxxE7MJ8qts8Eo9WfV51"
#define FIREBASE_FCM_DEVICE_TOKEN_1 "eSE-kpXfS6m3iEsV04uBW7:APA91bF6e36dTbyCx2XCztFVJ_e2bMAFX4RyWwER8f0bNkhg2A40wITXZzrPDuWimymZpD4cL_ZTxqNWhTFYSZIGRSw30dyg_PB2R_mIS5L6MS4f3Fg3E"
FirebaseData fbd01;

unsigned long lastTime = 0;
extern int notif_status;
int count = 0;
void sendMessage();
void fb();
void fb()
{
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
  fbd01.fcm.begin(FIREBASE_FCM_SERVER_KEY);
  fbd01.fcm.addDeviceToken(FIREBASE_FCM_DEVICE_TOKEN_1);
  fbd01.fcm.setPriority("high");
  fbd01.fcm.setTimeToLive(1000);
  sendMessage();
}

```

Figure 37: Firebase implmentation in Arduino 'app\_httpd.cpp'

```

void sendMessage()
{
    Serial.println("-----");
    Serial.println("Send Firebase Cloud Messaging...");

    fbdo1.fcm.setNotifyMessage("Arduino Home Security", "An intruder has been detected." + String(count));

    fbdo1.fcm.setDataMessage("{\"myData\": \" " + String(count) + "\"}");

    //if (Firebase.broadcastMessage(fbdo1))
    //if (Firebase.sendTopic(fbdo1))
    if (Firebase.sendMessage(fbdo1, 0))//send message to recipient index 0
    {
        Serial.println("PASSED");
        Serial.println(fbdo1.fcm.getSendResult());
        Serial.println("-----");
        Serial.println();
    }
    else
    {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo1.errorReason());
        Serial.println("-----");
        Serial.println();
    }
}

```

Figure 38: Firebase implentation in Arduino 'app\_httpd.cpp'

What the Arduino program essentially does is get the device token and authentication key, so it sends a notification to that specific device token which is authorised by the authentication key which is sent to Firebase and then sent from there to the app.

```

static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
    dl_matrix3du_t *aligned_face = NULL;
    int matched_id = 0;

    aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
    if(!aligned_face){
        Serial.println("Could not allocate face recognition buffer");
        return matched_id;
    }
    if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
        if (is_enrolling == 1){
            int8_t left_sample_face = enroll_face(&id_list, aligned_face);

            if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
                Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
            }
            Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
            rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
            if (left_sample_face == 0){
                is_enrolling = 0;
                Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
            }
        } else {
            matched_id = recognize_face(&id_list, aligned_face);
            if (matched_id >= 0) {
                Serial.printf("Match Face ID: %u\n", matched_id);
                rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u", matched_id);
            } else {
                Serial.println("No Match Found"); //push here
                rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
                matched_id = -1;
                notif_status=1;
            }
        }
    }
}

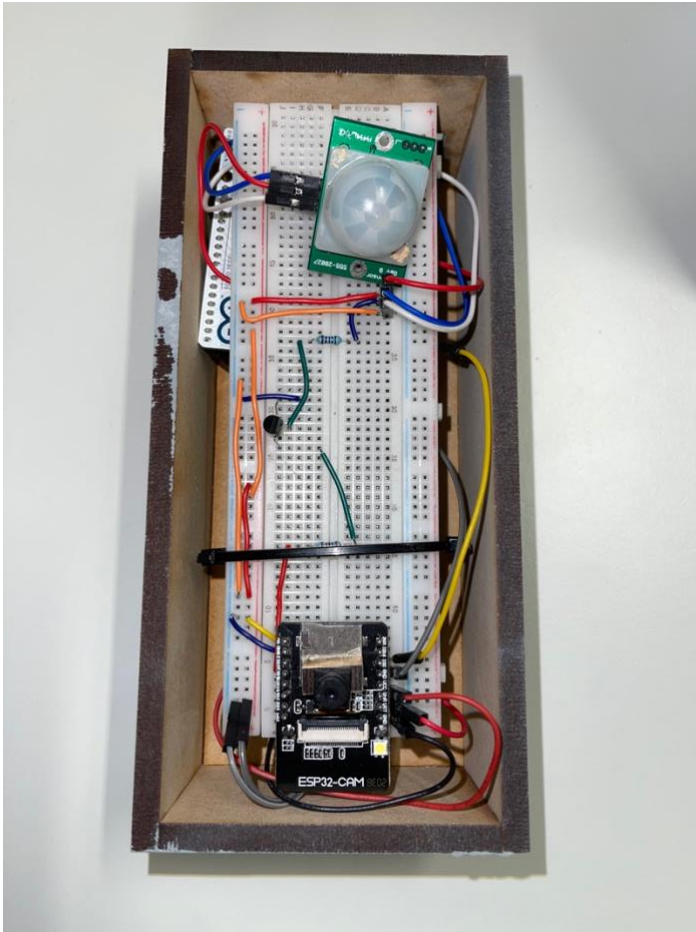
```

Figure 39: Arduino Face Recogniton ESP32-CAM; app\_httpd.cpp

The 'app\_httpd.cpp' file in Arduino also consists of the face recognition capabilities a snippet is shown in Figure 39. Here the code is showing how a face is detected and recognised, checks to see if it is a matched face, meaning a face that has been previously enrolled into the system. Recognised faces are greeted with a Hello message otherwise, it enrolls a new face and sets an ID number for that individual. For an unrecognised face

detected however, the intruder alert message is outputted. This is the message the homeowner will receive through Firebase as an intrusion alert.

### Final Product



*Figure 40: Arduino Home Security System - Final Product*

Figure 40 presents the final product. This is the Arduino Home Security System device enclosed into a casing. This casing was provided by the Stockwell Street Workshop for free, however, it is made up of just pieces of MDF wood and grey spray paint and some glue, therefore, is not costly and meets the aims of keeping this AHSS low-cost. The encased design is minimalistic and aesthetically pleasing as well as the colour grey was chosen for a professional look to match existing devices.

## Testing

Black box testing is a method of software testing which assesses the functionality of software without peering into its inner structure or coding. The primary source of this testing corresponds to each of the **functional requirements** outlined for the AHSS. These are:

Develop a secure, standard and low-cost smart security system for the home using Arduino

With an easy to use system and simple mobile application design to monitor their home remotely so beginner to advanced level Internet users can use this system with ease.

The main aspect of this project involves the ESP32-CAM. As it will recognise the individuals that reside in the home (enrolled faces) but for any motion that is detected and an unrecognised face is detected; their photo will be captured and it will alert the home owner as an app notification and store the photo of the intruder for relevant security measures to be conducted. The reason behind this being - that the number of notifications the user receives are kept minimal instead of notifying each and every time motion is detected as it is perhaps just family members and therefore not a security threat.

Scenario	Expected Outcome	Outcome	Pass/Fail
Incorrect login credentials entered	Access Denied. Should output that incorrect credentials were entered and the number of attempts should reduce by 1	Access is Denied. Outputs that incorrect credentials were entered and the number of attempts reduce by 1	Pass
Incorrect login credentials entered 3 times	Access Denied. Should output that incorrect credentials were entered and the number of attempts are 0 and Login button disables	Access is Denied. Outputs that incorrect credentials were entered and the number of attempts are 0 and Login button is disabled	pass
Correct login credentials entered	Access granted to the home screen	Access is granted to the home screen	pass
Successful login to be able to monitor the home	Home screen should be in view with user able to view live video stream	Home screen in view with user able to view live video stream	pass
Recognised face detected	Face recognised should be greeted with Hello Subject __	Face recognised is greeted with Hello Subject __	pass
Unrecognised face detected	Intruder alert message shown, homeowner notified with an alert	Intruder alert message shown, homeowner notified with an alert	fail
PIR sensor detecting objects correctly	Serial monitor outputs motion detected and motion stopped	Serial monitor outputs motion detected and motion stopped	pass
Intruder detected	Intruder detected sends notif to homeowner	Intruder detected sends notif to homeowner	pass
Start Stream button	When button is clicked live video stream is on show	When button is clicked live video stream is on show	pass

Stop stream button	When button is clicked live video stream stops and stream goes blank	As button is clicked live video stream stops and stream goes blank	pass
Capture still	When capture still button is clicked a photo is saved of the AHSS app home screen	When capture still button is clicked a photo is saved of the AHSS app home screen	fail

### Test Results Summary:

Above is the product testing that was conducted to establish whether the Arduino Home Security System meets its functional requirements and whether a standard, low-cost and secure system was achieved. From the results, it can be gathered that the Login Screen for the mobile application is fully functioning. When a user enters incorrect credentials, they are denied access to viewing the home, furthermore, three failed attempts to login results in the Login Button being disabled and thus preventing unauthorised access to the app.

## Product Evaluation

In order to evaluate the success of this Arduino Smart Security System, the cost, the ability to perform according to the aims as well as user satisfaction needs to be considered. Firstly, in evaluation of the costs of this security system it can be noted it is very inexpensive. Many online sites sell the hardware parts mentioned for very cheap. Damaged modules can easily be replaced and found nearly everywhere around the world. All of these hardware parts can be shipped from China and products then cost less than £5 GBP each. Thus, it can be concluded that this project aims to develop a low-cost security system is met.

In evaluation of the precision of the smart home security system, Black box testing was conducted. The test results reflected the success of this security system as the core objectives were met.

Essentially, the AHSS is capable of detecting motion through the PIR motion sensor, the household members face can be enrolled onto the ESP32-CAM. Meaning if the ESP32-CAM fails to detect any faces it will trigger a notification to the home owner. The iterative process of development and testing highlighted the flaws in the developed system and attempts were made to eradicate them, such as adjusting sensor sensitivity, repositioning device as well as extensive research carried out. It can be evaluated that the AHSS is easy to use as well as the design of the user-interface is simple and aesthetically pleasing. The same can be deducted for the AHS device with the casing made from wood and spray painted grey for aesthetic purposes and for a professional look to match existing systems currently out on the market. The overall performance of the AHSS is satisfactory and can be extensively used in real-life settings whether it is for the home, offices or schools etc. anywhere that would appreciate live video streaming that can be monitored through a mobile application with mobile notification alerts.

## Conclusion

Hereby, can conclude that the system can achieve majority of the objectives and work to a considerable standard. Throughout the development of AHS System managed to overcome various difficulties, many of which took extensive amount of time. Many of the intended features were implemented however others were not due to time constraints for this project. One of which being, the mobile application's inability to automatically save captured photos of an intruder in the background of the AHSS app as highlighted in the testing stages. Following agile methodology which focuses more on responding to change rather than a fixed plan was the best approach for the development of this AHS System as it meant relevant changes could be made to meet the functional requirements. The system design section also aided the development process for the AHSS as the schematic diagrams and flowchart illustrations provided a visual plan and step by step of what was expected to refer back to as well as the schematic diagrams for wiring purposes could potentially help others develop this device themselves.

In conclusion, the AHSS has full capability to monitor the home when needs be as well as monitor the home through the sensor used. The system essentially provides the user with a user-interface that works seamlessly in the way that it lets users monitor their home through the application. When/if there is any motion and the users may wish to watch the live video stream, then it is possible to do so with the user-interface that is developed. This android application that is developed is standard and secure as it consists of a Login screen to prevent unauthorised access.



## Future Work

The future scope of this project is so vast and the possibilities are endless as to what can be achieved with the same hardware or same software. Firstly, future implementation of the AHSS could include In-app ratings as a form of user acceptance testing too as it gives users the ability to offer feedback on their experience with the system and request improvements all in the long-run. These in-app ratings can be developed into the mobile application as a feature that allows users to input their feedback into the developed app itself, perhaps in the form of commonly used stars or a satisfaction rating between 1 to 5.

Currently, the AHSS device has to be set manually for start and stop of the streaming process. Future development could focus on potentially skipping the manual process and having specific times the system is active or inactive as set by the user. A buzzer being incorporated into this AHSS device would be rather relevant as it could be used to alert homeowners too of an intruder in their home, ore specifically a piezo buzzer could be used to play a tune or alarm sounds to not only notify household members but also deter an intruder. The AHSS could also be made to be extended for larger areas, as now it is limited to smaller areas and so with the use of perhaps extra motion sensors in various locations around the house or extra cameras it could provide homeowners added security and ability to monitor their homes and to make the application much more effective.

This device can also further be advanced to be located on the door and consist of locking and unlocking capabilities. The implementation of this would still be in line with the aims of this project of providing a standard, secure and low-cost system. It is rather feasible and again provides homes with that extra security and homeowners reassurance that the chance of burglaries or unauthorised entry will be reduced.

## Legal and Ethical Issues

It is rather necessary to acknowledge possible legal, ethical and social issues as disregarding it could lead to security flaws, user privacy data leakage, social, economic losses as well as a risk of human life as severe cases can be occurred by the attacks of intruders with malicious purpose.

Data confidentiality should be noted for this project, setting a strong password is essential to enhance security. Furthermore, data integrity is also key to prevent data from being changed. Especially in terms of for passwords and recognised faces in the face recognition database. Keeping these factors in check will ensure that the project aim of developing a secure Arduino based smart home security system is met.

## References

S. ur Rehman and V. Gruhn, "An approach to secure smart homes in cyber-physical systems/Internet-of-Things," *2018 Fifth International Conference on Software Defined Systems (SDS)*, 2018, pp. 126-129, doi: 10.1109/SDS.2018.8370433.

S. P. Makhanya, E. M. Dogo, N. I. Nwulu and U. Damisa, "A Smart Switch Control System Using ESP8266 Wi-Fi Module Integrated with an Android Application," *2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*, 2019, pp. 125-128, doi: 10.1109/SEGE.2019.8859904.

Simplisafe.co.uk. 2021. *SimpliSafe / Wireless Smart Home Security Systems & Burglar Alarms*. [online] Available at: <<https://simplisafe.co.uk>> [Accessed 2 February 2021].

UK Store View. 2021. *Yale Smart Living Alarm & Camera Range / YaleHome*. [online] Available at: <[https://yalehome.co.uk/smart-living?gclid=CjwKCAjw1uiEBhBzEiwAO9B\\_HeLs0y--aC1g7JxAaiE37zSGjZVb3GRm10lxnIkMULb2mjpWpIe8FxoCwA8QAvD\\_BwE](https://yalehome.co.uk/smart-living?gclid=CjwKCAjw1uiEBhBzEiwAO9B_HeLs0y--aC1g7JxAaiE37zSGjZVb3GRm10lxnIkMULb2mjpWpIe8FxoCwA8QAvD_BwE)> [Accessed 2 February 2021].

M. ShariqSuhail, G. ViswanathaReddy, G. Rambabu, C. V. R. DharmaSavarni and V. K. Mittal, "Multi-functional secured smart home," *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 2629-2634, doi: 10.1109/ICACCI.2016.7732455.

Suresh S., J. Bhavya, S. Sakshi, K. Varun and G. Debarshi, "Home Monitoring and Security system," *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, 2016, pp. 1-5, doi: 10.1109/ICTBIG.2016.7892665.

S. S. Syazlina Mohd Soleh, M. M. Som, M. H. Abd Wahab, A. Mustapha, N. A. Othman and M. Z. Saringat, "Arduino-Based Wireless Motion Detecting System," *2018 IEEE Conference on Open Systems (ICOS)*, 2018, pp. 71-75, doi: 10.1109/ICOS.2018.8632703.

Hidalgo, E., 2019. Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. *Heliyon*, 5(3), p.e01447.

Yan, J.; Lou, P.; Li, R.; Hu, J.; Xiong, J. Research on the Multiple Factors Influencing Human Identification Based on Pyroelectric Infrared Sensors. *Sensors* **2018**, *18*, 604. <https://doi.org/10.3390/s18020604>

Random Nerd Tutorials. 2021. *ESP32-CAM Video Streaming and Face Recognition with Arduino IDE / Random Nerd Tutorials*. [online] Available at: <<https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/>> [Accessed 7 December 2020].

Medium. 2021. *Face Detection and Recognition on the ESP32*. [online] Available at: <<https://medium.com/hacksters-blog/face-detection-and-recognition-on-the-esp32-3b4b9a35c765>> [Accessed 7 December 2020].

Firebase. 2020. *Firebase Cloud Messaging / Send notifications across platforms for free*. [online] Available at: <<https://firebase.google.com/products/cloud-messaging/>> [Accessed 3 April 2021].

Seedstudio.com. 2018. *ESP32-CAM Development Board(with camera)*. [online] Available at: <<https://www.seedstudio.com/ESP32-CAM-Development-Board-with-camer-p-3153.html>> [Accessed 9 April 2021].

Adafruit Learning System. 2016. *PIR Motion Sensor*. [online] Available at: <<https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/using-a-pir-w-arduino>> [Accessed 5 March 2021].

Youtube.com. 2020. *Simple Login App Tutorial for Beginners E01 - Login Activity Using Android Studio 3.6.3 (NEW)*. [online] Available at: <<https://www.youtube.com/watch?v=LCrhddpsgKU>> [Accessed 7 April 2021].

Stablekernel.com. 2019. *Mobile Development For Arduino Part 7 | Stable Kernel*. [online] Available at: <<https://stablekernel.com/article/mobile-development-arduino-part-7/>> [Accessed 8 April 2021].

Youtube.com. 2019. *Android Push Notifications + Firebase Cloud Messaging* - [online] Available at: <<https://www.youtube.com/watch?v=axX5VGzhboo>> [Accessed 7 April 2021].

En.wikipedia.org. 2019. *Intent (Android)* - Wikipedia. [online] Available at: <[https://en.wikipedia.org/wiki/Intent\\_\(Android\)](https://en.wikipedia.org/wiki/Intent_(Android))> [Accessed 7 April 2021].

(mobizt/Firebase-ESP32, 2020) GitHub. 2020. *mobizt/Firebase-ESP32*. [online] Available at: <<https://github.com/mobizt/Firebase-ESP32>> [Accessed 8 April 2021].

## Appendix

Android Studio

**package** com.nish.ahs;

**import** androidx.appcompat.app.AppCompatActivity;

**import** android.content.Intent;

**import** android.view.View;

**import** android.webkit.WebView;

**import** android.os.Bundle;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.TextView;

**import** android.widget.Toast;

**import** com.google.firebase.messaging.FirebaseMessagingService;

**public class** MainActivity **extends** AppCompatActivity {

**private** EditText eName;

**private** EditText ePassword;

**private** Button eLogin;

**private** TextView eAttemptsInfo;

```
private int counter=3;
```

```
class Credentials {  
    private String Username = "Admin";  
    private String Password = "2323";  
}
```

```
Boolean isValid = false;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main); //set to load the content of activity_main  
    Intent intentBackgroundService = new Intent(this, MyFirebaseMessagingService.class);  
    startService(intentBackgroundService);  
    eName = findViewById(R.id.etName); //connects the xml layout with the variables  
    ePassword = findViewById(R.id.etPassword);  
    eLogin = findViewById(R.id.btnLogin);  
    eAttemptsInfo = findViewById(R.id.tvAttemptsInfo);
```

```
eLogin.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view){
```

```
        String inputName = eName.getText().toString();
```

```
        String inputPassword = ePassword.getText().toString();
```

```
        if(inputName.isEmpty() || inputPassword.isEmpty())
```

```
        {
```

```
            Toast.makeText(MainActivity.this, "Enter correct username and password",  
Toast.LENGTH_SHORT).show();
```

```
        }else{
```

```
            isValid = validate(inputName, inputPassword);
```

```
            if(!isValid){
```

```
                counter--;
```

```

        eAttemptsInfo.setText("No. of attempts remaining:" + counter);

        if (counter == 0){
            eLogin.setEnabled(false); //disables the login button
            Toast.makeText(MainActivity.this, "You have used all your attempts try again later",
Toast.LENGTH_LONG).show();
        }
        else{
            Toast.makeText(MainActivity.this, "Incorrect username/password",
Toast.LENGTH_SHORT).show();
        }
    }

    else {

        Toast.makeText(MainActivity.this, "Login success", Toast.LENGTH_SHORT).show();

        //next activity
        startActivity(new Intent(MainActivity.this, MainScreen.class));
    }

}

}

});

}

private boolean validate(String Username, String Password)
{
    /* Get the object of Credentials class */
    Credentials credentials = new Credentials();

    /* Check the credentials */
    if(Username.equals(credentials.Username) && Password.equals(credentials.Password))
    {
        return true;
    }
}

```

```
        return false;
    }
}
```

```
package com.nish.ahs;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.core.app.NotificationCompat;
```

```
import android.app.NotificationManager;
```

```
import android.app.PendingIntent;
```

```
import android.content.Context;
```

```
import android.content.Intent;
```

```
import android.util.Log;
```

```
import android.view.View;
```

```
import android.webkit.WebView;
```

```
import android.os.Bundle;
```

```
import android.widget.Button;
```

```
import android.widget.TextView;
```

```
import com.google.firebase.messaging.FirebaseMessagingService;
```

```
import com.google.firebase.messaging.RemoteMessage;
```

```
import com.google.firebase.iid.FirebaseInstanceId;
```

```
public class MainScreen extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main_page);
```

```
        //IDs
```

```
        WebView view = (WebView) this.findViewById(R.id.webView);
```

```
        String url = "http://192.168.29.194:81/stream"; //Stream Address
```

```
        String blankpage = "http://www.e-try.com/black.htm"; //Address for blank page
```

```
String token = FirebaseInstanceId.getInstance().getToken();
TextView tokenTextView = (TextView) findViewById(R.id.textView2);
tokenTextView.setText(token);
```

*//Stream*

```
view.getSettings().setLoadWithOverviewMode(true);
view.getSettings().setUseWideViewPort(true);
view.loadUrl(blankpage); //Shows a black screen
```

*//Start Stream Button*

```
Button start = (Button) findViewById(R.id.start);
start.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        view.loadUrl(url); //Loads the stream url
    }
});
```

*//Stop Stream Button*

```
Button stop = (Button) findViewById(R.id.stop);
stop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        view.stopLoading(); //Stops loading the stream
        view.loadUrl("http://www.e-try.com/black.htm"); //Shows a black screen
    }
});
```

*//Snapshot Button*

```
Button snapshot = (Button) findViewById(R.id.snapshot);
snapshot.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String token = FirebaseInstanceId.getInstance().getToken();
        tokenTextView.setText(token);
    }
});
```

*//Email token to ourselves*

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, new String[]{"nishathumaira23@gmail.com"});
intent.putExtra(Intent.EXTRA_SUBJECT, "AHS - Device ID");
```

```

        intent.putExtra(Intent.EXTRA_TEXT, token);
        intent.setType("message/rfc822");
        startActivity(Intent.createChooser(intent, "Select Email Sending App :"));
    }
});

}

package com.nish.ahs;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.core.app.NotificationCompat;

import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFirebaseMessagingService extends FirebaseMessagingService {

    private static final int NOTIFICATION_ID = 1;

    public MyFirebaseMessagingService() {
        super();
    }

    @Override
    public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);
        Log.d("Msg", "Message received [" + remoteMessage + "]");

        Intent intent = new Intent(this, MainScreen.class);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 1410, intent,

```



```
PendingIntent.FLAG_ONE_SHOT);
```

```
NotificationCompat.Builder notificationBuilder = new
    NotificationCompat.Builder(this)
        .setSmallIcon(R.mipmap.ic_launcher)
        .setContentTitle("Message")
        .setContentText(remoteMessage.getNotification().getBody())
        .setAutoCancel(true)
        .setContentIntent(pendingIntent);

NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(NOTIFICATION_ID, notificationBuilder.build());
}
}
```

Arduino IDE

```
#include "esp_camera.h"
#include <WiFi.h>
#include <FirebaseESP32.h>

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"
//#include "PIR.ino"

const char* ssid = "wifi name";
const char* password = "wifi password";

//PIR Definitions
int led = 13;
const int sensor = 13; //Pin of PIR
int state = LOW; //No Motion Detected
int val = 0; //Sensor of status
```

```

int notif_status = 0;
extern int notif_status;

void startCameraServer();
void sendMessage();
void fb();
void setup() {
    pinMode(sensor, INPUT_PULLUP); //Take input from PIR
    Serial.begin(115200);
    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_sscb_sda = SIOD_GPIO_NUM;
    config.pin_sscb_scl = SIOC_GPIO_NUM;
    config.pin_pwdn = PWDN_GPIO_NUM;
    config.pin_reset = RESET_GPIO_NUM;
    config.xclk_freq_hz = 20000000;
    config.pixel_format = PIXFORMAT_JPEG;
    //init with high specs to pre-allocate larger buffers
    if(psramFound()){
        config.frame_size = FRAMESIZE_UXGA;
        config.jpeg_quality = 10;
        config.fb_count = 2;
    } else {
        config.frame_size = FRAMESIZE_SVGA;
        config.jpeg_quality = 12;
        config.fb_count = 1;
    }

```

```

#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("Camera init failed with error 0x%x", err);
    return;
}

sensor_t * s = esp_camera_sensor_get();
//initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
}
//drop down frame size for higher initial frame rate
s->set_framesize(s, FRAMESIZE_QVGA);

#ifdef CAMERA_MODEL_M5STACK_WIDE
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
#endif

WiFi.begin(ssid, password);

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");
//fb();
for (int notif_loop = 1; notif_loop != 0; notif_loop++) {
    if (notif_status == 1) {
        fb();
        notif_status = 0;
    }
    else {
        Serial.println("Listening for intruder");
    }
}
}
}

void loop() {

```

```

val = digitalRead(sensor); // read sensor value
if (val == HIGH) {        // check if the sensor is HIGH
    delay(500);           // delay 100 milliseconds

    if (state == LOW) {
        Serial.println("Motion stopped!");
        state = HIGH;    // update variable state to HIGH
    }
}
else {
    delay(500);           // delay 200 milliseconds

    if (state == HIGH){
        Serial.println("Motion detected!");
        state = LOW;     // update variable state to LOW
    }
}
}

// Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
// http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"
#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE 0x00FFFFFF
#define FACE_COLOR_BLACK 0x00000000
#define FACE_COLOR_RED 0x000000FF

```

```

#define FACE_COLOR_GREEN 0x0000FF00
#define FACE_COLOR_BLUE 0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

#include <WiFi.h>
#include <FirebaseESP32.h>

#define WIFI_SSID "Nishat's OnePlus"
#define WIFI_PASSWORD "A1B2C3D4E5F6"
#define FIREBASE_HOST "ahss-99c86.firebaseio.com"
#define FIREBASE_AUTH
"BM_Xu_vUL8M4E1jANuuUWEGXPj1MRFiu88tioghLrNnhserOO55uaXzLIZgV6A7g9-
rS78-SJBltyB5AhWaWME"
#define FIREBASE_FCM_SERVER_KEY
"AAAAtWTC5zg:APA91bETM698OFyhxvzrnvpqWUMmd4u2A0nYS--
m8aDDAfrQYkiFWjpkz_PbzBCQlrlZlS9diUoLbL2WJ64uoKZWgiYXmiRO8X50AAdD2x
7h5vJCipxxE7MJBqtsBEo9Wfv5lme3Mlrh"
#define FIREBASE_FCM_DEVICE_TOKEN_1 "e5E-
kpXfS6m3iEsV04uBW7:APA91bF6e36dTbyCx2XCztFVJ_e2bMAFX4RyWWeR8fObNhk
hg2A40wITXZrPDuWimymZpD4cL_ZlXqNWhTFYYSZIGRSw30dyg_PB2R_mI55L6MS4f
3Fg3Bmr9q1LmxKRmnu0RvSfjSMn"
FirebaseData fbdo1;

unsigned long lastTime = 0;
extern int notif_status;
int count = 0;
void sendMessage();
void fb();
void fb()
{
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    Firebase.reconnectWiFi(true);
    fbdo1.fcm.begin(FIREBASE_FCM_SERVER_KEY);
    fbdo1.fcm.addDeviceToken(FIREBASE_FCM_DEVICE_TOKEN_1);
    fbdo1.fcm.setPriority("high");
    fbdo1.fcm.setTimeToLive(1000);
    sendMessage();
}

void sendMessage()
{
    Serial.println("-----");
    Serial.println("Send Firebase Cloud Messaging...");

    fbdo1.fcm.setNotifyMessage("Arduino Home Security", "An intruder has been detected."
+ String(count));

```

```

fbdo1.fcm.setDataMessage("{\"myData\":" + String(count) + "}");

//if (Firebase.broadcastMessage(fbdo1))
//if (Firebase.sendTopic(fbdo1))
if (Firebase.sendMessage(fbdo1, 0))//send message to recipient index 0
{

    Serial.println("PASSED");
    Serial.println(fbdo1.fcm.getSendResult());
    Serial.println("-----");
    Serial.println();
}
else
{
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo1.errorReason());
    Serial.println("-----");
    Serial.println();
}

count++;
}

typedef struct {
    size_t size; //number of values used for filtering
    size_t index; //current value index
    size_t count; //value count
    int sum;
    int * values; //array to be filled with values
} ra_filter_t;

typedef struct {
    httpd_req_t *req;
    size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "12345678900000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

static mtmn_config_t mtmn_config = {0};
static int8_t detection_enabled = 0;

```

```

static int8_t recognition_enabled = 0;
static int8_t is_enrolling = 0;
static face_id_list id_list = {0};

static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t sample_size){
    memset(filter, 0, sizeof(ra_filter_t));

    filter->values = (int *)malloc(sample_size * sizeof(int));
    if(!filter->values){
        return NULL;
    }
    memset(filter->values, 0, sample_size * sizeof(int));

    filter->size = sample_size;
    return filter;
}

static int ra_filter_run(ra_filter_t * filter, int value){
    if(!filter->values){
        return value;
    }
    filter->sum -= filter->values[filter->index];
    filter->values[filter->index] = value;
    filter->sum += filter->values[filter->index];
    filter->index++;
    filter->index = filter->index % filter->size;
    if (filter->count < filter->size) {
        filter->count++;
    }
    return filter->sum / filter->count;
}

static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char * str){
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
}

static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char *format, ...){
    char loc_buf[64];
    char * temp = loc_buf;
    int len;
    va_list arg;
    va_list copy;
    va_start(arg, format);
    va_copy(copy, arg);

```

```

len = vsnprintf(loc_buf, sizeof(loc_buf), format, arg);
va_end(copy);
if(len >= sizeof(loc_buf)){
    temp = (char*)malloc(len+1);
    if(temp == NULL) {
        return 0;
    }
}
vsnprintf(temp, len+1, format, arg);
va_end(arg);
rgb_print(image_matrix, color, temp);
if(len > 64){
    free(temp);
}
return len;
}

```

```

static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes, int
face_id){
    int x, y, w, h, i;
    uint32_t color = FACE_COLOR_YELLOW;
    if(face_id < 0){
        color = FACE_COLOR_RED;
    } else if(face_id > 0){
        color = FACE_COLOR_GREEN;
    }
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    for (i = 0; i < boxes->len; i++){
        // rectangle box
        x = (int)boxes->box[i].box_p[0];
        y = (int)boxes->box[i].box_p[1];
        w = (int)boxes->box[i].box_p[2] - x + 1;
        h = (int)boxes->box[i].box_p[3] - y + 1;
        fb_gfx_drawFastHLine(&fb, x, y, w, color);
        fb_gfx_drawFastHLine(&fb, x, y+h-1, w, color);
        fb_gfx_drawFastVLine(&fb, x, y, h, color);
        fb_gfx_drawFastVLine(&fb, x+w-1, y, h, color);
    }
    #if 0
        // landmark
        int x0, y0, j;
        for (j = 0; j < 10; j+=2) {
            x0 = (int)boxes->landmark[i].landmark_p[j];
            y0 = (int)boxes->landmark[i].landmark_p[j+1];
        }
    #endif
}

```



```

        fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
    }
#endif
}
}

static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
    dl_matrix3du_t *aligned_face = NULL;
    int matched_id = 0;

    aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
    if(!aligned_face){
        Serial.println("Could not allocate face recognition buffer");
        return matched_id;
    }
    if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
        if (is_enrolling == 1){
            int8_t left_sample_face = enroll_face(&id_list, aligned_face);

            if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
                Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
            }
            Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
            rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]",
id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
            if (left_sample_face == 0){
                is_enrolling = 0;
                Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
            }
        } else {
            matched_id = recognize_face(&id_list, aligned_face);
            if (matched_id >= 0) {
                Serial.printf("Match Face ID: %u\n", matched_id);
                rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u",
matched_id);

                } else {
                    Serial.println("No Match Found"); //push here
                    rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
                    matched_id = -1;
                    notif_status=1;
                    //fb();

                }
        }
    }
}
}

```

```

    } else {
        Serial.println("Face Not Aligned");
        //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
    }

    dl_matrix3du_free(aligned_face);
    return matched_id;
}

static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){
    jpg_chunking_t *j = (jpg_chunking_t *)arg;
    if(!index){
        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
        return 0;
    }
    j->len += len;
    return len;
}

static esp_err_t capture_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    int64_t fr_start = esp_timer_get_time();

    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");

    size_t out_len, out_width, out_height;
    uint8_t * out_buf;
    bool s;
    bool detected = false;
    int face_id = 0;
    if(!detection_enabled || fb->width > 400){
        size_t fb_len = 0;
        if(fb->format == PIXFORMAT_JPEG){
            fb_len = fb->len;
            res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
        } else {

```

```

        jpg_chunking_t jchunk = {req, 0};
        res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk)?ESP_OK:ESP_FAIL;
        httpd_resp_send_chunk(req, NULL, 0);
        fb_len = jchunk.len;
    }
    esp_camera_fb_return(fb);
    int64_t fr_end = esp_timer_get_time();
    Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end -
fr_start)/1000));
    return res;
}

dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
if (!image_matrix) {
    esp_camera_fb_return(fb);
    Serial.println("dl_matrix3du_alloc failed");
    httpd_resp_send_500(req);
    return ESP_FAIL;
}

out_buf = image_matrix->item;
out_len = fb->width * fb->height * 3;
out_width = fb->width;
out_height = fb->height;

s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
esp_camera_fb_return(fb);
if(!s){
    dl_matrix3du_free(image_matrix);
    Serial.println("to rgb888 failed");
    httpd_resp_send_500(req);
    return ESP_FAIL;
}

box_array_t *net_boxes = face_detect(image_matrix, &mtmn_config);

if (net_boxes){
    detected = true;
    if(recognition_enabled){
        face_id = run_face_recognition(image_matrix, net_boxes);
    }
    draw_face_boxes(image_matrix, net_boxes, face_id);
    free(net_boxes->score);
    free(net_boxes->box);
    free(net_boxes->landmark);
    free(net_boxes);
}

jpg_chunking_t jchunk = {req, 0};

```

```

    s = fmt2jpg_cb(out_buf, out_len, out_width, out_height, PIXFORMAT_RGB888, 90,
jpg_encode_stream, &jchunk);
    dl_matrix3du_free(image_matrix);
    if(!s){
        Serial.println("JPEG compression failed");
        return ESP_FAIL;
    }

    int64_t fr_end = esp_timer_get_time();
    Serial.printf("FACE: %uB %ums %s%d\n", (uint32_t)(jchunk.len), (uint32_t)((fr_end -
fr_start)/1000), detected?"DETECTED ":"", face_id);
    return res;
}

static esp_err_t stream_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    size_t _jpg_buf_len = 0;
    uint8_t * _jpg_buf = NULL;
    char * part_buf[64];
    dl_matrix3du_t *image_matrix = NULL;
    bool detected = false;
    int face_id = 0;
    int64_t fr_start = 0;
    int64_t fr_ready = 0;
    int64_t fr_face = 0;
    int64_t fr_recognize = 0;
    int64_t fr_encode = 0;

    static int64_t last_frame = 0;
    if(!last_frame) {
        last_frame = esp_timer_get_time();
    }

    res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
    if(res != ESP_OK){
        return res;
    }

    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");

    while(true){
        detected = false;
        face_id = 0;
        fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");
            res = ESP_FAIL;
        } else {
            fr_start = esp_timer_get_time();

```

```

fr_ready = fr_start;
fr_face = fr_start;
fr_encode = fr_start;
fr_recognize = fr_start;
if(!detection_enabled || fb->width > 400){
    if(fb->format != PIXFORMAT_JPEG){
        bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
        esp_camera_fb_return(fb);
        fb = NULL;
        if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
        }
    } else {
        _jpg_buf_len = fb->len;
        _jpg_buf = fb->buf;
    }
} else {

image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);

if (!image_matrix) {
    Serial.println("dl_matrix3du_alloc failed");
    res = ESP_FAIL;
} else {
    if(!fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item)){
        Serial.println("fmt2rgb888 failed");
        res = ESP_FAIL;
    } else {
        fr_ready = esp_timer_get_time();
        box_array_t *net_boxes = NULL;
        if(detection_enabled){
            net_boxes = face_detect(image_matrix, &mtmn_config);
        }
        fr_face = esp_timer_get_time();
        fr_recognize = fr_face;
        if (net_boxes || fb->format != PIXFORMAT_JPEG){
            if(net_boxes){
                detected = true;
                if(recognition_enabled){
                    face_id = run_face_recognition(image_matrix, net_boxes);
                }
                fr_recognize = esp_timer_get_time();
                draw_face_boxes(image_matrix, net_boxes, face_id);
                free(net_boxes->score);
                free(net_boxes->box);
                free(net_boxes->landmark);
                free(net_boxes);
            }

```

```

        if(!fmt2jpg(image_matrix->item, fb->width*fb->height*3, fb->width, fb-
>height, PIXFORMAT_RGB888, 90, &_jpg_buf, &_jpg_buf_len)){
            Serial.println("fmt2jpg failed");
            res = ESP_FAIL;
        }
        esp_camera_fb_return(fb);
        fb = NULL;
    } else {
        _jpg_buf = fb->buf;
        _jpg_buf_len = fb->len;
    }
    fr_encode = esp_timer_get_time();
}
dl_matrix3du_free(image_matrix);
}
}
}
if(res == ESP_OK){
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
    res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
}
if(res == ESP_OK){
    res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
}
if(res == ESP_OK){
    res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
}
if(fb){
    esp_camera_fb_return(fb);
    fb = NULL;
    _jpg_buf = NULL;
} else if(_jpg_buf){
    free(_jpg_buf);
    _jpg_buf = NULL;
}
if(res != ESP_OK){
    break;
}
int64_t fr_end = esp_timer_get_time();

int64_t ready_time = (fr_ready - fr_start)/1000;
int64_t face_time = (fr_face - fr_ready)/1000;
int64_t recognize_time = (fr_recognize - fr_face)/1000;
int64_t encode_time = (fr_encode - fr_recognize)/1000;
int64_t process_time = (fr_encode - fr_start)/1000;

int64_t frame_time = fr_end - last_frame;
last_frame = fr_end;
frame_time /= 1000;

```

```

        uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
        Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps),
%u+%u+%u+%u=%u %s%d\n",
        (uint32_t)(_jpg_buf_len),
        (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
        avg_frame_time, 1000.0 / avg_frame_time,
        (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)recognize_time,
        (uint32_t)encode_time, (uint32_t)process_time,
        (detected)?"DETECTED ":"", face_id
        );
    }

    last_frame = 0;
    return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
    char* buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[32] = {0,};

    buf_len = httpd_req_get_url_query_len(req) + 1;
    if (buf_len > 1) {
        buf = (char*)malloc(buf_len);
        if(!buf){
            httpd_resp_send_500(req); //500
            return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
            if (httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK &&
                httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
            } else {
                free(buf);
                httpd_resp_send_404(req);
                return ESP_FAIL;
            }
        } else {
            free(buf);
            httpd_resp_send_404(req);
            return ESP_FAIL;
        }
    }
    free(buf);
    if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
        if (httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK &&
            httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
            int val = atoi(value);
            sensor_t * s = esp_camera_sensor_get();

```

```

int res = 0;

if(!strcmp(variable, "framesize")) {
    if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s, (framesize_t)val);
}
else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s, (gainceiling_t)val);
else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
else if(!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s, val);
else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
else if(!strcmp(variable, "face_detect")) {
    detection_enabled = val;
    if(!detection_enabled) {
        recognition_enabled = 0;
    }
}
else if(!strcmp(variable, "face_enroll")) is_enrolling = val;
else if(!strcmp(variable, "face_recognize")) {
    recognition_enabled = val;
    if(recognition_enabled){
        detection_enabled = val;
    }
}
else {
    res = -1;
}

if(res){
    return httpd_resp_send_500(req);
}

httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");

```



```

    return httpd_resp_send(req, NULL, 0);
}

static esp_err_t status_handler(httpd_req_t *req){
    static char json_response[2048]; //2048

    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';

    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
    p+=sprintf(p, "\"contrast\":%d,", s->status.contrast);
    p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
    p+=sprintf(p, "\"sharpness\":%d,", s->status.sharpness);
    p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
    p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
    p+=sprintf(p, "\"awb\":%u,", s->status.awb);
    p+=sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
    p+=sprintf(p, "\"aec\":%u,", s->status.aec);
    p+=sprintf(p, "\"aec2\":%u,", s->status.aec2);
    p+=sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
    p+=sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
    p+=sprintf(p, "\"agc\":%u,", s->status.agc);
    p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
    p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
    p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);
    p+=sprintf(p, "\"wpc\":%u,", s->status.wpc);
    p+=sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
    p+=sprintf(p, "\"lenc\":%u,", s->status.lenc);
    p+=sprintf(p, "\"vflip\":%u,", s->status.vflip);
    p+=sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
    p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
    p+=sprintf(p, "\"colorbar\":%u,", s->status.colorbar);
    p+=sprintf(p, "\"face_detect\":%u,", detection_enabled);
    p+=sprintf(p, "\"face_enroll\":%u,", is_enrolling);
    p+=sprintf(p, "\"face_recognize\":%u", recognition_enabled);
    *p++ = '}';
    *p++ = 0;
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, json_response, strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t *req){
    httpd_resp_set_type(req, "text/html");
    httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
    sensor_t * s = esp_camera_sensor_get();
    if (s->id.PID == OV3660_PID) {

```

```

        return httpd_resp_send(req, (const char *)index_ov3660_html_gz,
index_ov3660_html_gz_len);
    }
    return httpd_resp_send(req, (const char *)index_ov2640_html_gz,
index_ov2640_html_gz_len);
}

void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t index_uri = {
        .uri      = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t status_uri = {
        .uri      = "/status",
        .method    = HTTP_GET,
        .handler   = status_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t cmd_uri = {
        .uri      = "/control",
        .method    = HTTP_GET,
        .handler   = cmd_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t capture_uri = {
        .uri      = "/capture",
        .method    = HTTP_GET,
        .handler   = capture_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t stream_uri = {
        .uri      = "/stream",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };

    ra_filter_init(&ra_filter, 20);

    mtmn_config.type = FAST;
    mtmn_config.min_face = 80;

```

```

mtmn_config.pyramid = 0.707;
mtmn_config.pyramid_times = 4;
mtmn_config.p_threshold.score = 0.6;
mtmn_config.p_threshold.nms = 0.7;
mtmn_config.p_threshold.candidate_number = 20;
mtmn_config.r_threshold.score = 0.7;
mtmn_config.r_threshold.nms = 0.7;
mtmn_config.r_threshold.candidate_number = 10;
mtmn_config.o_threshold.score = 0.7;
mtmn_config.o_threshold.nms = 0.7;
mtmn_config.o_threshold.candidate_number = 1;

face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);

Serial.printf("Starting web server on port: '%d'\n", config.server_port);
if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
    httpd_register_uri_handler(camera_httpd, &status_uri);
    httpd_register_uri_handler(camera_httpd, &capture_uri);
}

config.server_port += 1;
config.ctrl_port += 1;
Serial.printf("Starting stream server on port: '%d'\n", config.server_port);
if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
}
}

```