

# **COVguard**

## **(AUTO TEMPERATURE AND MASK SCANNING ENTRY SYSTEM) MAJOR PROJECT REPORT**

Submitted in partial fulfilment of the requirements  
for the award of the Degree of

### **BACHELOR OF TECHNOLOGY**

In

### **MECHATRONICS ENGINEERING**

2018-2022

Submitted By:

Pratishtha	03570211218
Abhishek Kumar	41270211218
Isha Kushwaha	02070211218



**Under the Guidance of:**

Dr. R.K. Dhammi

Dr. Arun Gupta

Dr. Charu Gaur

**DEPARTMENT OF MECHATRONICS ENGINEERING**

**DELHI SKILL AND ENTREPRENEURSHIP UNIVERSITY**

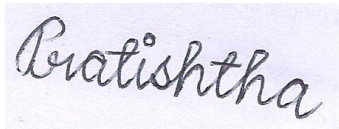
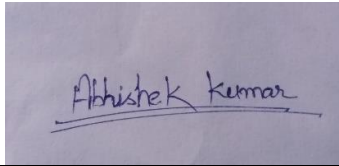
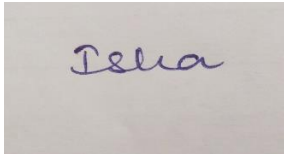
OKHLA – II CAMPUS, NEW DELHI-110020

(Formerly DELHI INSTITUTE OF TOOL ENGINEERING, DELHI)

## DECLARATION

It is hereby certified that the work which is being presented in the Major Project report entitled “**COVguard**” in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in **MECHATRONICS** and is submitted to the Department of Mechatronics Engineering of **DSEU Okhla Campus - II ( Formerly DELHI INSTITUTE OF TOOL ENGINEERING ,** New Delhi affiliated to **Guru Gobind Singh Indraprastha University )** is an authentic record of our own work carried out during the period of 8<sup>th</sup> semester under the guidance of Dr. R.K Dhammi, Dr. Arun Gupta & Dr. Charu Gaur.

The matter presented in the Major Project Report has not been submitted by us for the award of any other degree of this or any other Institute.

S.No.	Enrollment No.	Name	Signature
1	03570211218	Pratishtha	
2	41270211218	Abhishek Kumar	
3	02070211218	Isha Kushwaha	

## CERTIFICATE

This is to certify that the project entitled “**COVguard**” is being submitted in partial fulfilment for the award of degree of Bachelor of Technology in **MECHATRONIC ENGINEERING** of the **DSEU Campus – II ( Formerly DELHI INSTITUTE OF TOOL ENGINEERING )**.

It is a record of their own work, carried out under the supervision of respective guide and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title. This Project Report is the record of authentic work carried out by the following below mentioned students during the period of Eight (8th) Semester.

NAME	ENROLLMENT NO:
PRATISHTHA	03570211218
ABHISHEK KUMAR	41270211218
ISHA KUSHWAHA	02070211218

\_\_\_\_\_  
INTERNAL EXAMINER

\_\_\_\_\_  
EXTERNAL EXAMINER

## ACKNOWLEDGEMENT

An endeavour of a long period can be successful only with the advice of many well-wishers. We take this opportunity to express our heartfelt gratitude and appreciation to all those who encouraged us for successfully completion of the project work.

Our special thanks to **Prof. M A Khan (Director cum Principal)** and **Dr. Arun Gupta (H.O.D), (Department of Mechatronics)** for providing us with the opportunity. Their valuable support and encouragement for project work.

We would like to express our sincere gratitude to **Dr. R.K. Dhammi** (Our Guide) and **Dr. Charu Gaur** (Assistant Professor, as our Co-Guide) during the progress of project work, for their timely suggestions and help in spite of their busy schedule.

We wish to express our sincere thanks to Management of **Delhi Institute of tool engineering**, New Delhi for their consistent encouragement to complete the project work.

Finally, we would like to express our sincere thanks to project coordinators, one and all who have helped us to complete the project work successfully.

## CONTENTS

❖ DECLARATION .....	ii
❖ CERTIFICATE .....	iii
❖ ACKNOWLEDGMENT .....	iv
❖ LIST OF FIGURES.....	vi
<u>Chapter -1</u> Introduction .....	1
1.1 Face Mask Detector with Temperature Sensor Kit.....	1
<u>Chapter -2</u> Problem Identification And Literature Review .....	2
2.1 Problem Statement.....	2
2.2 Literature Review .....	3
2.3 The Compact Solution for Any Access Control Environment.....	5
2.3.1 Key Benefits.....	5
2.3.2 Features And Functions.....	5
<u>Chapter -3</u> Methodology.....	7
3.1 Circuitry.....	7
3.2 Components Used- Hardware.....	7
3.3 Other Components.....	14
3.4 Software Used .....	15
3.5 Work Flow.....	16
<u>Chapter -4</u> Result And Discussion .....	17
4.1 Kit Image.....	17
4.2 Final Output Images.....	18
<u>Chapeter -5</u> Conclusion.....	25
➤ References .....	26
➤ Appendix .....	27

## LIST OF FIGURES

❖ Fig. 2.1	SHOWING BASIC IDEA BEHIND THE WORKING OF COVGUARD
❖ Fig. 3.1	CIRCUITRY
❖ Fig. 3.2.1	ARDUINO UNO
❖ Fig. 3.2.2	ESP32 CAM WIFI
❖ Fig. 3.2.3	ULTRASONIC SENSOR
❖ Fig. 3.2.4	IR TEMPERATURE SENSOR
❖ Fig. 3.2.5	LCD I2C MODULE
❖ Fig. 3.2.6	SERVO MOTOR
❖ Fig. 3.2.7	PUSH BUTTON MODULE
❖ Fig. 3.2.8	LEDS
❖ Fig. 3.3.1	USB TO TTL UART SERIAL CONVERTER MODULE
❖ Fig. 3.3.2	DC-DC STEP DOWN POWER MODULE
❖ Fig. 3.3.3	JUMPER WIRES
❖ Fig. 3.5	WORK FLOW
❖ Fig. 4.1	KIT IMAGE
❖ Fig. 4.2	FINAL OUTPUT IMAGES

# **CHAPTER - 1**

## **INTRODUCTION**

SARS-COV2, alternatively known as covid-19 has become a serious public health issue around the globe. Because of its contact-transparent characteristics, it is rapidly spreading. The use of a face mask is among the most efficient methods for preventing the transmission of the Covid-19 virus. Wearing the face mask alone can cut the chance of catching the virus by over 70%. Consequently, World Health Organization (WHO) advised wearing masks in crowded places as precautionary measures. To solve this challenge, we needed a reliable mask and temperature monitoring system.

### **1.1 Face Mask Detector with Temperature Sensor Kit:**

In this kit, user need to scan their body temperature first before scanning their face mask. It senses temperature contactless rather than need a contact to read temperature. In this project, your body **temperature** should be between *33 - 39-degree Celsius* to pass the scan. Other than that, it shows error. After passing the temperature scan, you need to scan your face. You need to **stand between 35-60cm** from the camera to scan your face. If not, the camera will not scan your face and ask you to come closer or take a step back. You need to pass the **threshold value**, which is *80% of your face covered* to pass the scan. There is also a **pushbutton** to *open the door manually* from inside; in case of any emergency situation or stampede. **Green LED** will glow while passing the scan and **Red LED** will glow when the parameters are not satisfied and entry is barred. **Live Stream** can also be accessed by the IP Address shown on the LCD Screen, basically working as a CCTV Camera.

Any person will not be provided entry without temperature and mask scan. **Only person having both conditions satisfied is instantly allowed inside.**

The system uses temperature sensor and camera connected with an Arduino and ESP32 system to control the entire operation. The ESP32 Cam is used to scan for mask and temperature sensor for forehead temperature. The Arduino processes the sensor inputs and decides whether the person is to be allowed. In this case the system operates a motor to open the barrier allowing the person to enter the premises. If a person is flagged by system for high temperature or no Mask the system glows the red light and bars the person from entry. Also, the face and temperature of person is transmitted over IOT to server for authorities to take action and test the person for covid. We can also view the livestream through the shown IP Address and it basically works as our CCTV Camera.

## **CHAPTER - 2**

### **PROBLEM IDENTIFICATION AND LITERATURE REVIEW**

**2.1 Problem Statement:** In the present scenario due to Covid-19, the need for face mask detection applications, temperature detection and hand sanitizing are now in high demand for Railway Entrance, Airport Entrance, Office Entrance, Museums and Amusement Parks, Other Public Places and enterprises to ensure safety. These steps are now done in manual way by which the personnel may get in contact with the other personnel while sanitizing and checking temperature might not be accurate. Now that many shops, offices and institutions are re-opening again after the Corona lockdown, many businesses are faced with the need to provide the best possible protection for their staff and customers. Face masks and body temperature checks play an important part in the protection effort.

While this is already done routinely and at a large scale at airports or railway stations, many businesses and institutions are struggling to meet the challenge. Face mask monitoring often requires additional staff resources. At the same time, body temperature checks by staff come with certain risks in terms of hygiene and data privacy.

The first step to detect COVID is by scanning for fever. Also, we need to monitor every person for a mask. We have temperature checking systems for every entrance for scanning but **manual temperature scanning has a lot of disadvantages.**

- The personnel are not well trained on using temperature scanner devices.
- There is human error in reading values.
- Many times, people are not barred from entry even after higher temperature readings or no masks.
- The scanning is skipped by the personnel if supervisors are not watching.
- Manual scanning system is not suitable for large crowds.
- The person scanning the temperature is at risk of being infected.

To mitigate the problem, aiming to increase Covid-19 entrance safety, covering several relevant aspects: Contactless temperature sensing and Mask detection.

Contactless temperature sensing subsystem relies on Arduino Uno using temperature sensor, while mask detection performed by leveraging computer vision techniques on camera-equipped ESP32 Cam. Any person without temperature check and mask scan will not be provided entry. Only person having the conditions satisfied by the system is instantly allowed inside. From the simulation results, it is clearly observed that the proposed method has high accuracy compare to the existing methods. Thus, the system provides a 100% automated system to prevent the spread of Covid-19.

**To solve this problem, we here propose a fully automated temperature scanner and entry provider system.** It is a multipurpose system that has a wide range of applications. The system makes use of a contactless temperature scanner and a mask monitor. The scanner is connected directly with a human barrier to bar entry if high temperature or no



mask is detected. Any person will not be provided entry without temperature and mask scan. Only person having both conditions is instantly allowed inside. The system uses temperature sensor and camera connected with Arduino Uno system to control the entire operation. The camera is used to scan for mask and temperature sensor for forehead temperature. The Arduino processes the sensor inputs and decides whether the person is to be allowed. In this case the system operates a motor to open the barrier allowing the person to enter the premises. If a person is flagged by system for high temperature or no Mask the system glows the red light and bars the person from entry. Also, the face and temperature of person is transmitted over IOT to server for authorities to take action and test the person for COVID. Thus, the system provides automated system to prevent the spread of COVID. The system makes use of a contactless temperature scanner and a mask monitor. The scanner is connected directly with a human barrier to bar entry if high temperature or no mask is detected. Any person will not be provided entry without temperature and mask scan. Only person having both conditions is instantly allowed inside.

**2.2 Literature Review:** The coronavirus disease, or COVID-19, which originated primarily in Wuhan, China, has rapidly spread to several countries, including India, the world's second-most populous country with a population of more than 134 billion people. With such a large population, India would have trouble preventing the spread of the coronavirus. Face masks and sanitizers are the most effective ways to minimize transmission. When it comes to reducing disease transmission, this has shown good results.

Fever, sore throat, tiredness, loss of taste and smell, and nasal congestion are all common symptoms of coronavirus infection. The majority of the time, it is transmitted indirectly through surfaces. The incubation period can be very long, ranging from 10 to 14 days in extreme cases, and the virus can attack directly (from one individual to other individuals) by respiratory droplets. Governments implemented a variety of protection and safety initiatives to reduce disease transmission, including social distancing, mandatory mask-wearing, quarantine, restricting citizens' traveling within state boundaries and abroad, self-isolation, and the exclusion and cancellation of big social occasions and meetings. From work activities to social relationships, all kinds of sports activities, as well as off-screen and on-screen entertainment have all been affected due to this COVID-19 pandemic. Individuals with high body temperature are not to be permitted to enter public places because they are at a high risk of infection and spreading the virus; wearing a mask is essential. At the entrances to any city, workplaces, malls, and hospital gates, temperature and mask checks are also necessary.

COVID 19 pandemic is causing a global health epidemic. The most powerful safety tool is wearing a face mask in public places and everywhere else. The COVID 19 outbreak forced governments around the world to implement lockdowns to deter virus transmission. According to survey reports, wearing a face mask at public places reduces the risk of transmission significantly.

COVID 19 has made a huge impact on the society, the new restriction has been imposed as in the number of users allowed in a particular room in offices, shops, etc. To maintain social

distancing, along with social distancing regular temperature check at entrances of malls, the office is mandatory.

In this work, IoT based Arduino is used. Temperature monitoring using Arduino is an exciting and secure process. Furthermore, this flexible system obtains more values in calculating the actuator from the data saved on the internet. IoT is used for connecting the electronic devices with the internet. The devices may vary from the temperature measuring equipment and vehicles SOS system to other electronic devices such as sensors, software's, and network connectivity facilities, which sanction collecting and exchanging data. The twenty-first century has witnessed a massive paradigm shift to and focusing on global attention onto IoT as a burgeoning discipline with multiple possibilities and diverse opportunities for growth and development with soap or a hand sanitizer. The next best thing is social distancing.

To avoid getting infected or spreading it, it is essential to wear a face mask while going out from home especially to public places such as markets or hospitals As Countries around the Globe are Reopening, living with the Novel Coronavirus is becoming the new way of life. But to Stop the Spread of the Virus we need to separate people having the Coronavirus from the Rest. According to the CDC, fever is the leading symptom of the Coronavirus with up to 83% of Symptomatic Patients showing some signs of fever. Many Countries are making Temperature Check-up's and Masks mandatory for Schools, Colleges, Offices, and other Workplaces. Currently, Temperature check-ups are done manually using Contactless Thermometer. Manual check-ups can be Inefficient, Impractical (in places with a large crowd), and Risky. A non- contact infrared sensor thermometer is useful for measuring temperature under circumstance where thermocouple or other probe type sensors cannot be used or do not produce accurate data for a variety of reasons. IoT based devices in homes and industries are used for controlling all the electrical or electronic devices which are present. Additionally, the saved information of the IoT devices can be controlled from anywhere. The sensor analyses the graphical representation of the observed data in every user-defined format wherever in the world.

In this project, an IoT-enabled smart door that uses a machine learning model for monitoring body temperature and face mask detection. The proposed model can be used for any shopping mall, hotel, apartment entrance, etc. As an outcome a cost-effective and reliable method of using AI and sensors to build a healthy environment. Besides, the body temperature of the individual is monitored using a non-contact temperature sensor. This proposed system can detect the users from COVID 19.

As a result, a smart entry device that automatically monitors human body temperature and detects a mask at the door opening system is developed. An advanced idea is used in this system approach, which is a combination of both including temperature detection and mask detection. Unlike, handheld thermometers that need an individual to take a person's body temperature, COVguard provides non-contact and efficient temperature check and facial mask detection to allow or deny the entry of people in buildings and event venues. With artificial intelligence (AI) technology, it does not require human observation or intervention to assist frontline workers to perform epidemic prevention and control.

## **2.3 The Compact Solution for Any Access Control Environment**

Designed for lobbies and indoor entrances, the solution package consists of a freely placeable kit with built-in screen display unit, complete with integrated IR-thermometer (thermoscan) and video camera. The solution interfaces seamlessly with existing door opener mechanisms to provide fully automated controlled access.

User interaction is very easy and straightforward: graphical instructions shown on the screen guide users through the scanning process, which takes only a few seconds. The result is instantly shown on the screen. If both requirements are met, access is granted. Persons who are not wearing a face mask are requested to do so by visual prompts. In case the scanned temperature is in the fever range, the device automatically trigger. In case of problems, users can also initiate a support call and the door will be opened manually.

### **2.3.1 Key Benefits:**

- Fully automated monitoring of compliance with face mask and body temperature requirements
- Reliable detection within seconds
- Contactless surfaces
- Customisable system response and video communication in case of non-compliance
- LCD Screen with self-explanatory video interaction for guiding users through the entire scanning process
- Saves staff resources
- Avoids potential hygiene and data privacy risks, thanks to fully automated, data protection compliant technology

### **2.3.2 Features and Functions:**

- Mobile design for flexible placement in or in front of buildings
- Handy Kit with built-in thermal sensor and video cameras
- Easy plug-and-play installation using regular power and network connections
- All components are IP capable
- Additional visual prompts to ensure correct face position (lower display unit)
- Face mask recognition can be disabled as needed
- Integrated control contact for automated actuation of turnstile gates, doors, barriers, etc
- Package includes IP-based query terminal with LCD screen for user interaction and display
- Live streaming through given IP address, acting as a CCTV camera



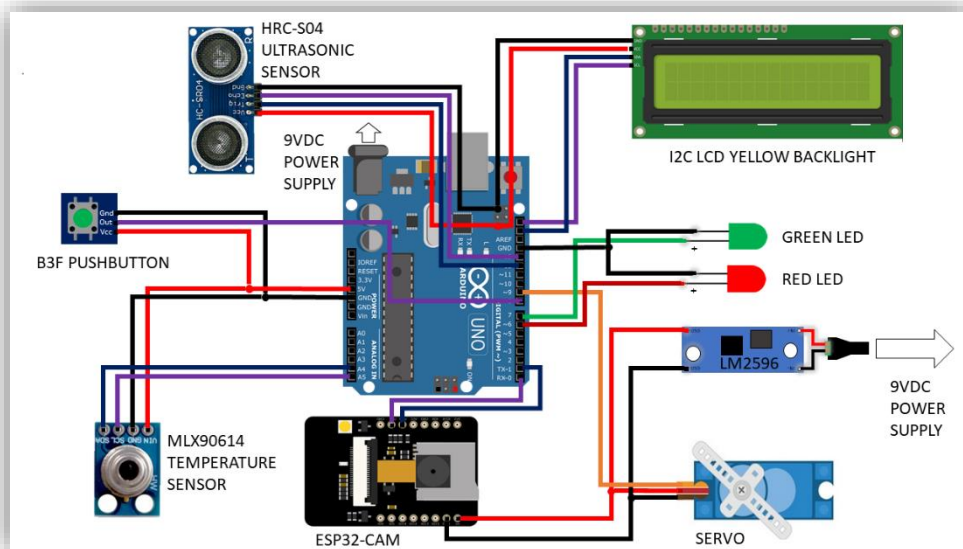
*Fig.2.1: Showing basic idea behind the working of COVguard*

## CHAPTER - 3

### METHODOLOGY

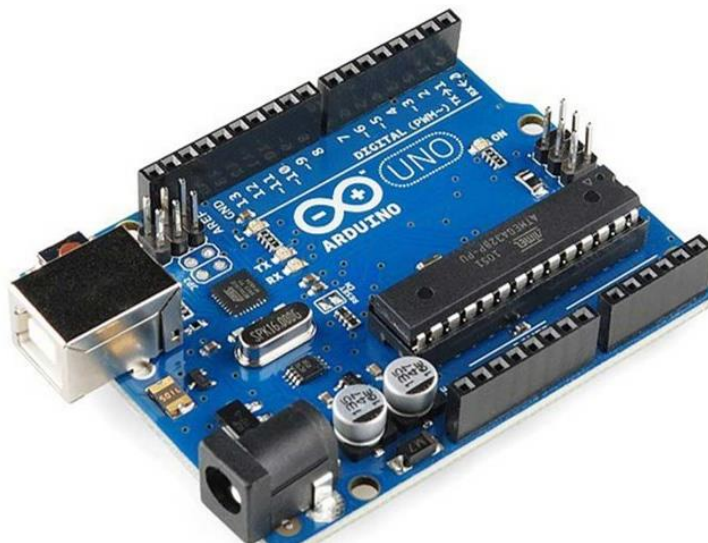
#### 3.1 CIRCUITRY

The following schematic diagram shows the various components used in the project along with their connections.



#### 3.2 COMPONENTS USED- HARDWARE

##### 3.2.1 ARDUINO UNO



Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller. The Arduino UNO board is mostly used in electronics projects and the user can program this board keeping in mind the requirements of the project. The board has regular innovation and a bug fix in the design of the board to make the board suitable for the project's use. The Arduino UNO board is considered as the most used board. The Arduino UNO board is primarily used over other Arduino products because of the following reasons.

1. As the board can be easily connected to the other computer system via USB port. The USB port fixed in the board serves two purposes. It can be used to supply the power supply to the board and can act as a serial device to connect the board to a computer system.
2. The board is capable to get the power supply from DC adaptor having a voltage of 12 V. The board can be charged from this external power supply.
3. The microcontroller used in the board i.e., ATmega328 has the flexibility provided to the board. It means the controller chip can be replaced, removed from the board in case of damage or improper functioning of the chip. This flexibility functionality is not provided in other Arduino boards.
4. The board pins are capable of functioning for constant power supply of 5 v. The digital and analog pins are used to adjust the voltage supply in the board.
5. As the board design is simple it can be used by multiple users and the community support for the Arduino UNO board.
6. The Arduino UNO board has a list of several hardware components and has the capability to interact with those devices. The device includes Bluetooth, internet, motor control, and many more.
7. The main use of the Arduino UNO board over other Arduino board is the price factor. The price of this board is lowest compared to other Arduino products. This is the main reason to prefer this board over other boards.

- ❖ The table that follows enlists the technical specifications of the Arduino uno board.

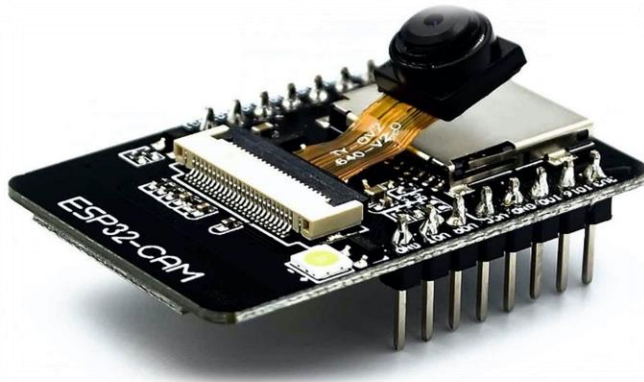
BOARD	NAME	ARDUINO UNO
	SKU	A000066
MICROCONTROLLER	AT mega 328p	
USB CONNECTOR	USB-B	
PINS	BUILT-IN LED PIN	13
	DIGITAL I/O PINS	14
	ANALOG INPUT PINS	6
	PWM PINS	6
COMMUNICATION	UART	YES
	I2C	YES
	SPI	YES
POWER	I/O VOLTAGE	5V
	INPUT VOLTAGE(NOMINAL)	7-12 V
	DC INPUT PER I/O PIN	20 mA
	POWER SUPPLY CONNECTOR	BARREL PLUG
CLOCK SPEED	MAIN PROCESSOR	ATmega 328p 16 MHz
	USB SERIAL PROCESSOR	ATmega 16U2 16 MHz
MEMORY	ATmega 328p	2KB SRAM, 32 KB FLASH MEMORY, 1 KB EEPROM
DIMENSIONS	WEIGHT	25 g
	WIDTH	53.4 mm
	LENGTH	68.6 mm

TABLE: TECHNICAL SPECIFICATIONS OF ARDUINO BOARD

### 3.2.2 ESP32 CAM WIFI

The ESP32-CAM is a full-featured microcontroller that also has an integrated video camera and microSD card socket. It's inexpensive and easy to use, and is perfect for IoT devices requiring a camera with advanced functions like image tracking and recognition. The sample software distributed by *Espressif* includes a sketch that allows you to build a web-based camera with a sophisticated control panel. A square white LED on the top of the module, this can act as a “flash” for illuminating the subject you are trying to view with the camera. The ESP32-CAM is based upon the ESP32-S module, so it shares the same specifications. It has the following features:

- 802.11b/g/n Wi-Fi
- Bluetooth 4.2 with BLE
- UART, SPI, I2C and PWM interfaces
- Clock speed up to 160 MHz
- Computing power up to 600 DMIPS
- 520 KB SRAM plus 4 MB PSRAM
- Supports Wi-Fi Image Upload
- Multiple Sleep modes
- Firmware Over the Air (FOTA) upgrades possible
- 9 GPIO ports
- Built-in Flash LED



The ESP32-CAM includes an OV2640 camera module. The device also supports OV7670 cameras. The OV2640 has the following specifications:

- 2 Megapixel sensor
- Array size UXGA 1622×1200
- Output formats include YUV422, YUV420, RGB565, RGB555 and 8-bit compressed data
- Image transfer rate of 15 to 60 fps

Using the ESP32-CAM is similar to using the ESP32 modules we looked at previously, with one major difference. The ESP32-CAM board has no USB port, so you can't just connect it up to your computer and start loading programs.

Instead, we will need to add an external FTDI adapter. This is the same adapter you would use when programming an Arduino board.



Module Model	ESP32 CAM
Package	DiP 16
Size	27*40.5*4.5 (±0.2) mm
SPI flash	Default 32 MBIT
Ram	520KB SRAM +4M PSRAM
Bluetooth	Bluetooth 4.2 BR/EDR and BLE standards
Wi-Fi	Wi-Fi 802.11 b/g/n/
Support interface	SPI, UART, I2C, PWM
Support TF card	Maximum support 4G
Io port	9
UART baud rate	Default 115200 bps
Image output format	Image Output Format JPEG (OV2640 support only), BMP, GRAYSCALE
Spectrum range	2412 ~2484MHz
Antenna	Onboard PCB antenna, gain 2dBi
Transmit power	802.11b: 17±2 dBm (@11Mbps) 802.11g: 14±2 dBm (@54Mbps) 802.11n: 13±2 dBm (@MCS7)
Receiving sensitivity	CCK, 1 Mbps: -90dBm CCK, 11 Mbps: -85dBm 6 Mbps (1/2 BPSK): -88dBm 54 Mbps (3/4 64-QAM): -70dBm MCS7 (65 Mbps, 72.2 Mbps): -67dBm
Power dissipation	Turn off the flash lamp:180mA@5V Turn on the flash lamp and turn on the brightness to the maximum:310mA@5V Deep-sleep: Minimum power consumption can be achieved 6mA@5V Modem-sleep: Minimum up to 20mA@5V Light-sleep: Minimum up to 6.7mA@5V
Security	WPA/WPA2/WPA2-Enterprise/WPS
Power supply range	5V
Operating temperature	20°C- 85 °C
Storage environment	40°C- 90 °C, < 90%RH
Weight	10 g

### 3.2.3 ULTRASONIC SENSOR

Ultrasonic sensors are devices that generate or sense ultrasound energy. They can be divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert **electrical signals** into **ultrasound**, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

This technology, as well, can detect approaching objects and track their positions. Ultrasound can also be used to make point-to-point distance measurements by transmitting and receiving discrete bursts of ultrasound between transducers. This technique is known as **Sonomicrometry** where the transit-time of the ultrasound signal is measured electronically (i.e. digitally) and converted mathematically to the distance between transducers assuming the speed of sound of the medium between the transducers is known. This method can be very precise in terms of temporal and spatial resolution because the time-of-flight measurement can be derived from tracking the same incident (received) waveform either by reference level or zero crossing. This enables the measurement resolution to far exceed the wavelength of the sound frequency generated by the transducers.



### 3.2.4 IR TEMPERATURE SENSOR

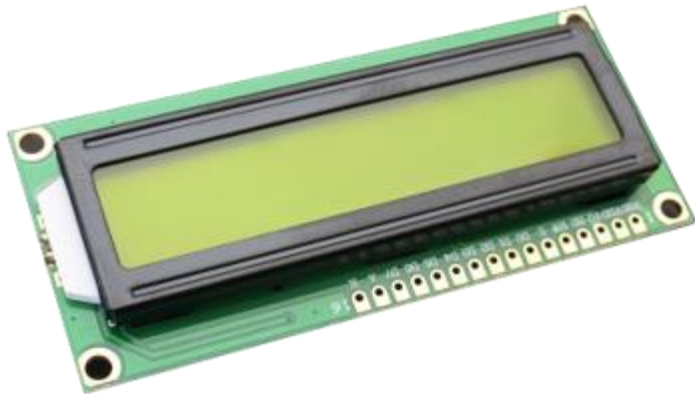
Infrared sensors are used to make non-contact temperature measurements. They are available in many configurations to cover a broad range of applications. They are available for temperatures from below freezing up to the temperature range of heat-treating ovens. They can be used for close or far distances, for small or large objects, some have very fast response times, and some can even see-through certain windows.

- Both the IR sensitive thermopile detector chip and the signal conditioning ASIC are integrated in the same TO-39 can.
- Integrated into the MLX90614 are a low noise amplifier, 17-bit ADC and powerful DSP unit thus achieving high accuracy and resolution of the thermometer.
- The thermometer comes factory calibrated with a digital SM Bus output giving full access to the measured temperature in the complete temperature range(s) with a resolution of 0.02°C.
- The user can configure the digital output to be pulse width modulation (PWM). As a standard, the 10-bit PWM is configured to continuously transmit the measured temperature in range of -20 to 120°C, with an output resolution of 0.14°C.



### 3.2.5 LCD I2C MODULE

The wiring between a normal LCD and Arduino is complex. Therefore, LCD I2C has been created keeping this in mind. LCD I2C is composed of a normal LCD, LCD I2C module and a potentiometer.

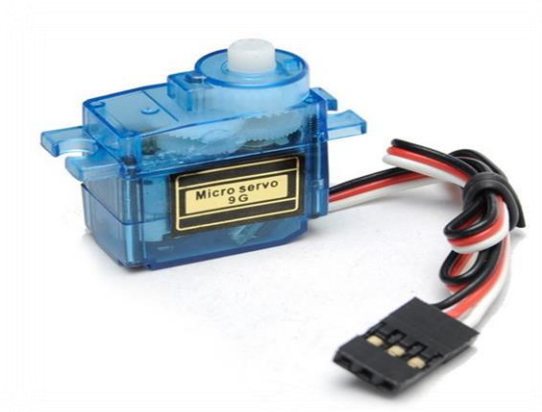


I2C LCD is ideal for displaying text and numbers.

The LCD screens we carry, however, have a little “backpack” on them which does a few convenient things:

- a potentiometer for adjusting the screen contrast, allow you to switch on/off LCD screen’s backlight, and
- most significantly: condense all of the communication between the LCD screen and the Arduino into two little wires (SDA and SCL) using the I<sup>2</sup>C communication protocol.

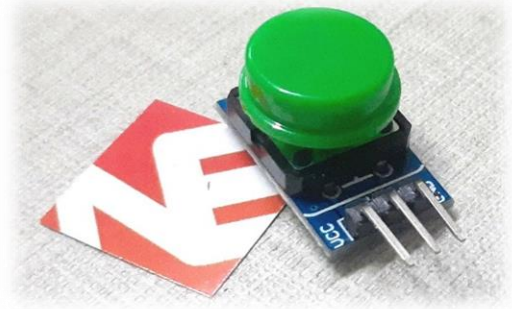
### 3.2.6 SERVO MOTOR



A servo motor is used to demonstrate the opening and closing of the main door. A Servo Motor produces velocity and torque based on the voltage and the amount of current supplied. It also works as a part of a closed-loop system providing velocity and torque as commanded from the servo controller with a feedback device to close the loop.

### 3.2.7 PUSH BUTTON MODULE

It is a simple switch which only on/conduct when the button is pressed. Most of the push-buttons are designed to operate with human hand. So, the top of the push-button always a flat structure. Here we have used a PCB mount type push-button.



### 3.2.8 LEDS

LEDs have many advantages over incandescent light sources, including lower power consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. In exchange for these generally favorable attributes, disadvantages of LEDs include electrical limitations to low voltage and generally to DC (not AC) power, inability to provide steady illumination from a pulsing DC or an AC electrical supply source, and lesser maximum operating temperature and storage temperature. In contrast to LEDs, incandescent lamps can be made to intrinsically run at virtually any supply voltage, can utilize either AC or DC current interchangeably, and will provide steady illumination when powered by AC or pulsing DC even at a frequency as low as 50 Hz. LEDs usually need electronic support components to function, while an incandescent bulb can and usually does operate directly from an unregulated DC or AC power source.



## 3.3 OTHER COMPONENTS

### 3.3.1 USB TO TTL UART SERIAL CONVERTER MODULE

With this USB to TTL Converter module, we can establish communication with your MCU through the computer. It gives maximum transmission speed up to 3Mbps. The main chip is FT232RL from FTDI, with high stability. LED indicator flashes when the communication is being done. It is very useful when you are debugging or downloading.



### 3.3.2 DC-DC STEP DOWN POWER MODULE

LM2596 DC-DC buck converter step-down power module with a high-precision potentiometer for adjusting output voltage, capable of driving a load up to 3A with high efficiency. When the output current required is greater than 2.5A(10W).



### 3.3.3 JUMPER WIRES



A jump wire is an **electrical wire**, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a **breadboard** or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the **header connector** of a circuit board, or a piece of test equipment. We have made use of Male-to-male, Female-to-female and Female-to-male jumper wires in making connections with various components.

## 3.4 SOFTWARE USED

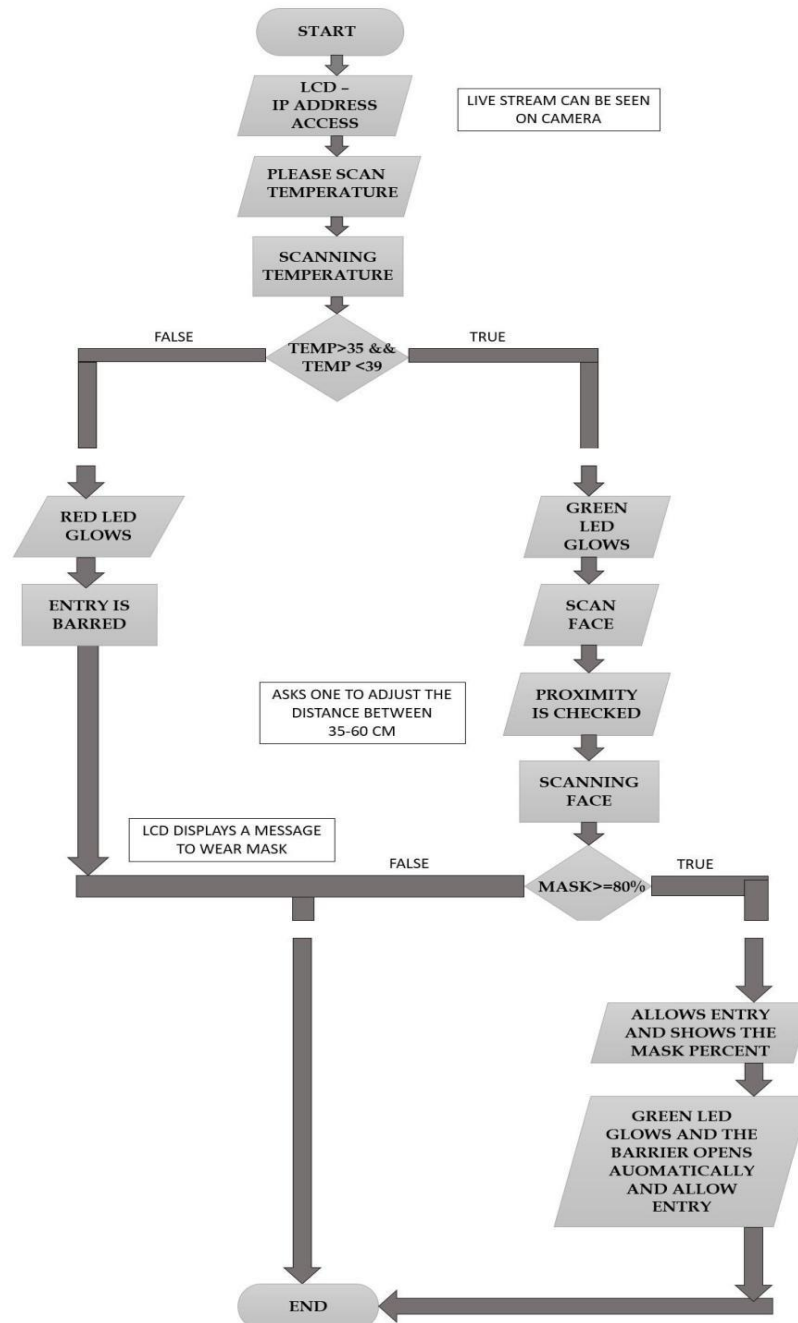
### 3.4.1 ARDUINO SOFTWARE (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.



### 3.5 WORK FLOW

The flow chart that follows is a diagrammatic representation of the process. Process when arranged in a sequential manner helps in better understanding of the flow of the process.

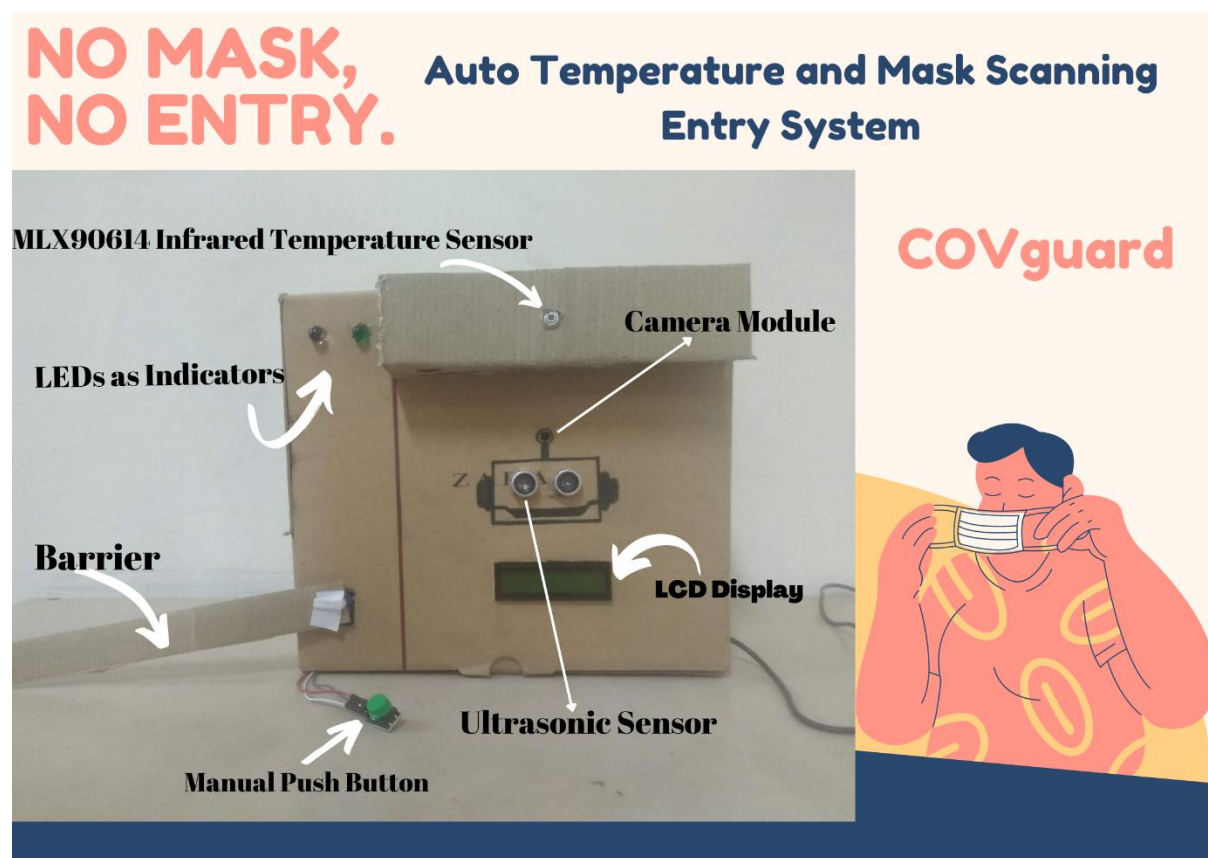


*Fig: Flow chart illustrating the work flow*

## CHAPTER - 4

### RESULT AND DISCUSSION

#### 4.1 Auto Temperature And Mask Scanning Entry System Kit



## 4.2 FINAL OUTPUT IMAGES

The first step of the process is to provide power supply to the circuit through a 9V DC Adapter, the LCD switches on and has to be connected to WiFi to proceed further. It then displays the IP address.

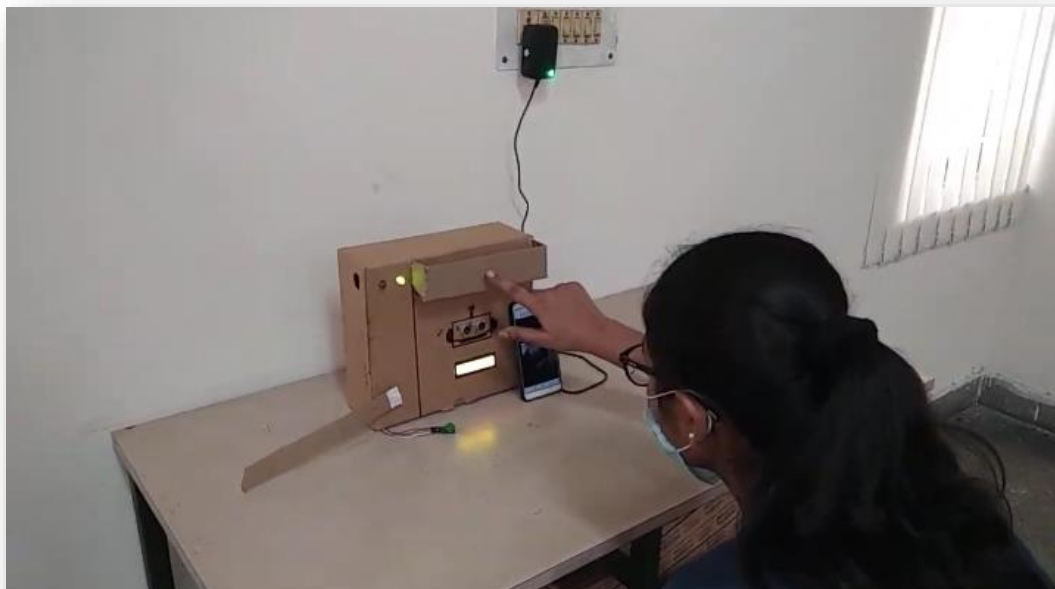
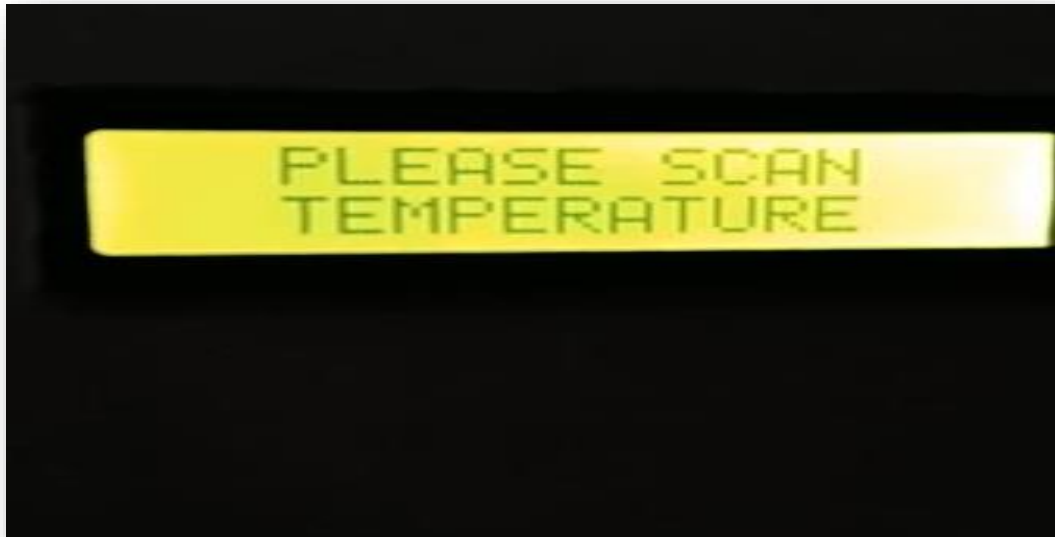


The IP address can be typed in any web browser on which live streaming can be seen.





The LCD then displays a message to scan the temperature. The IR temperature sensor is deployed for this purpose.



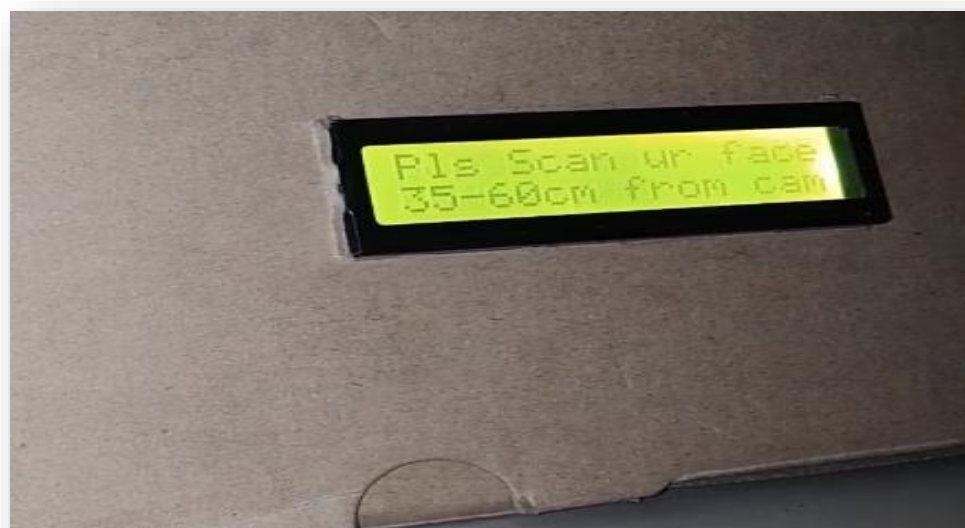
The acceptable values for temperature are 35°C- 39°C. If the measured temperature is not in this range, it will show an error and the white LED glows and the scanning is complete with the barrier remaining closed. The process ends then and there.

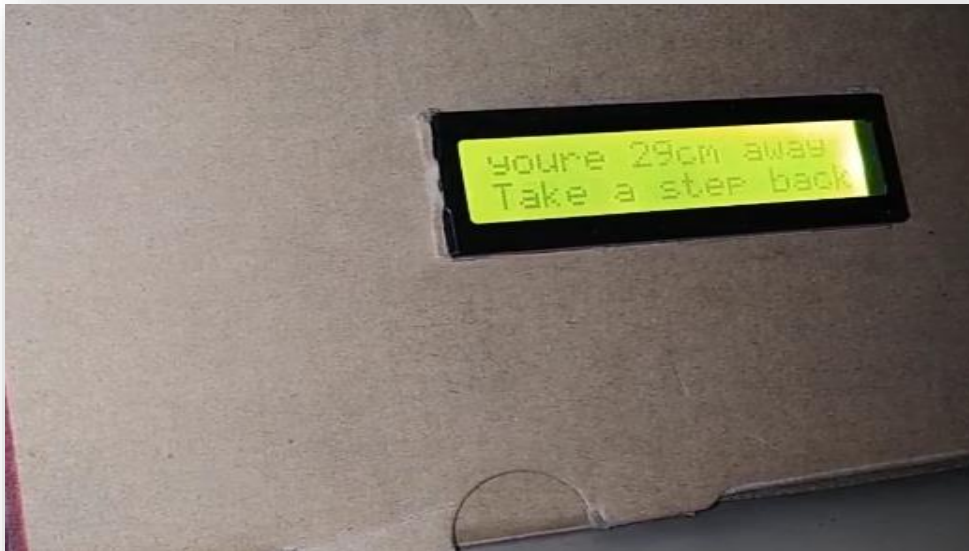
If the temperature measured is within the permissible range, it will proceed to the next step.



**The next step is to detect the mask.**

Meanwhile it also checks the proximity of the person from the scanner. You have to maintain appropriate distance from the scanner. The subject should be standing between 35 cm and 60 cm. Otherwise, it displays a message to move accordingly.





The threshold value for the mask is set to 80%. If the percentage of mask detected is less than the set value, it will simply show the percentage of mask along with an advisory to wear one.

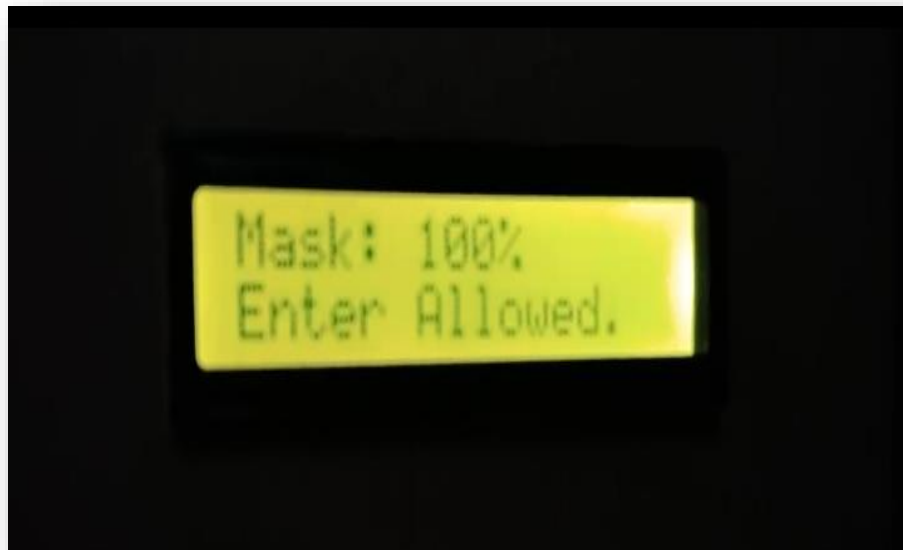




White LED glows which indicate that the entry is barred and the scanning is complete.



If the percentage of mask detected is more than 80%, the green LED glows.



The human barrier opens automatically afterwards, allowing the entry.



In case of an emergency, we may require to open the door manually. For this purpose, we have a push button module which is accessible to the authorities only.



## **CHAPTER – 5**

### **CONCLUSION**

According to the achieved results, the proposed solution is usable for its purpose. To contribute towards communal health, our project aims to devise a highly accurate and automated temperature and mask scanning entry system that can efficiently detect temperature and non-mask faces and thus, enforcing to wear mask and ensuring safety checks. This protection is provided thanks to the complete barrier between proximities of individuals, nasal and oral cavities with the outside world.

Finally, the ultimate goal is to integrate the kit “**COVguard**” presented in this paper with our social and official frameworks (such as Railways Entries, Airport Entries, Hospitals Entries, Offices Entries, Educational Institutions, Museums & Amusement Parks and Other Public Places) for efficient resource planning during pandemic and upcoming crisis in order to enable efficient safety and security scheduling.

Our project presents a chance to be more ready for the next crisis or to evaluate the effects of huge scope social change in respecting sanitary protection rules.

## **REFERENCES**

Hunt, K. (2020, June 1). Social distancing and masks reduce risk of getting covid-19. CNN.

Retrieved from

<https://edition.cnn.com/2020/06/01/health/review-masks-social-distancing-covid-19-wellness/index.html>

Khan, R. (2021, June). Temperature and mask scanning system.

ResearchGate.

Retrieved from

[https://www.researchgate.net/publication/352393974\\_Auto\\_Temperature\\_and\\_Mask\\_Scanning\\_Entry\\_System](https://www.researchgate.net/publication/352393974_Auto_Temperature_and_Mask_Scanning_Entry_System)

Mukhtar, A. (2019, May). Smart Health Care Monitoring System based on IOT. Arduino Project Hub.

Retrieved from

<https://create.arduino.cc/projecthub/156471/smart-health-care-monitoring-system-based-on-iot-f559b3>

Akinwale, O. (2020, March). Designing embedded systems with Arduino microcontrollers: A way forward -ResearchGate.

Retrieved from

[https://www.researchgate.net/publication/342416323\\_Designing\\_Embedded\\_Systems\\_with\\_Arduino\\_Microcontrollers\\_A\\_Way\\_Forward\\_for\\_Technological\\_Advancement\\_in\\_Nigeria](https://www.researchgate.net/publication/342416323_Designing_Embedded_Systems_with_Arduino_Microcontrollers_A_Way_Forward_for_Technological_Advancement_in_Nigeria)



## APPENDIX

### Program code

#### Arduino code:

```
//Libraries
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Ultrasonic.h>
#include <Adafruit_MLX90614.h>
//Define constant values
#define SERVO_PIN 9
#define LED_GREEN 7
#define LED_RED 6
#define PUSH_BUTTON_PIN 8
#define USTRIG_PIN 12
#define USECHO_PIN 13

#define START_RUN 'a'
#define RESTART_ESP32 'r'
#define RESTART_SCAN 'n'

#define CLOSE_DOOR 180
#define OPEN_DOOR 90
#define DOOR_LOCK 0
#define OPEN_DOOR_TIME 4000
#define CLOSE_DOOR_TIME 1000
#define DELAY_TEMP_SUCCESS 2000
#define RESTART_TIME 5000

#define ON 0
#define OFF 1

#define MASK_DETECT_TH 80 //in percent %
#define START_SCAN_COUNT 15
#define DISTANCE_ERROR_COUNT 10
#define ERROR_COUNT 5
#define TEMP_MAX_ALLOWABLE 39
#define TEMP_THRESHOLD 35

#define MINIMUM_DISTANCE 35
#define MAXIMUM_DISTANCE 60

#define NO_OBJECT -1
#define TEMP_LOW 0
#define TEMP_HIGH 1
#define TEMP_ALLOWED 2
#define SCAN_NO_OBJECT 0
#define SCAN_PASS 1
```

```

#define SCAN_ERROR                2
#define DISP_IP_INDEX             0
#define DISP_DISTANCE_ERROR_INDEX 1

// set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27,16,2);
Servo myServo;
Ultrasonic ultrasonic(USTRIG_PIN, USECHO_PIN);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

char gszIP_Add[20];
unsigned short gusMask_Detect = 0;
unsigned short gusLCD_Index = 0;
unsigned short gusDisp_Index = 0;
unsigned short gusIsNeedDisp = 1;
unsigned short gusIsNeed_Restart = 0;
unsigned short gusIsSend_Request = 0;

unsigned long gulStart_Timer_LCD = 0;
unsigned long gulRestart_Timer = 0;
unsigned long gulDistance_Timer = 0;

void setup()
{
    char szData[30];
    unsigned short usExit = 0;
    unsigned short usData_Len = 0;
    short a = 0;

    Serial.begin(9600);
    mlx.begin();

    lcd.init();                // initialize the lcd
    lcd.backlight();

    pinMode(PUSH_BUTTON_PIN,INPUT_PULLUP);
    pinMode(LED_GREEN,OUTPUT);
    pinMode(LED_RED,OUTPUT);

    vLED_Control(SCAN_NO_OBJECT);
    vServo_Control(DOOR_LOCK);

    lcd.clear();
    lcd.print("Wifi");
    lcd.setCursor(0,1);
    lcd.print("connecting...");
    memset(szData, '\0', sizeof(szData));
    memset(gszIP_Add, '\0', sizeof(gszIP_Add));

```

```

do
{
  usData_Len = usRead_Serial_Data(szData, sizeof(szData));

  if(usData_Len > 0)
  {
    for(short i=0; i<usData_Len; i++)
    {
      if(szData[i] == '#')
      {
        i++;
        while(szData[i] != ',')
        {
          gszIP_Add[a] = szData[i++];
          a++;
        }
        usExit = 1;
        break;
      }
    }
  }
  else
  {
    if(gusIsNeed_Restart == 0)
    {
      Serial.println(RESTART_ESP32);

      gusIsNeed_Restart = 1;
    }

    if((millis() - gulRestart_Timer) > RESTART_TIME)
    {
      gusIsNeed_Restart = 0;
      gulRestart_Timer = millis();
    }
  }
} while(usExit == 0);

vLCD_Disp_Ip(gszIP_Add);
gusDisp_Index = DISP_IP_INDEX;
gulStart_Timer_LCD = millis();
}

void loop()
{
  char szData[30];
  unsigned short usExit = 0;
  unsigned short usTempExit = 0;
  unsigned short usData_Len = 0;
  unsigned short usTemp_Allowed = NO_OBJECT;

```

```

unsigned short usIsNeed_Rescan = 0;
unsigned short usFace_Distance = 0;
unsigned short usStart_Scanning_Count = 0;
unsigned short usDistance_Error = 0;
unsigned short usError_Count = 0;
short sMask_Percent = 0;
float fTemperature_Object = 0;

while(usTempExit == 0)
{
    fTemperature_Object = mlx.readObjectTempC();

    if(fTemperature_Object > TEMP_MAX_ALLOWABLE)
    {
        // temp high
        vLED_Control(SCAN_ERROR);
        usTemp_Allowed = TEMP_HIGH;
        gusIsNeedDisp = 1;
    }
    else if(fTemperature_Object > TEMP_THRESHOLD)
    {
        vLED_Control(SCAN_PASS);
        usTemp_Allowed = TEMP_ALLOWED;

        gusIsNeedDisp = 1;
        usTempExit = 1;
    }
    else
    {
        vLED_Control(SCAN_NO_OBJECT);
        usTemp_Allowed = NO_OBJECT;
    }
    if(gusIsNeedDisp == 1)
    {
        if(usTemp_Allowed != NO_OBJECT)
        {
            vDisp_Temp_Sensor(usTemp_Allowed, fTemperature_Object);
        }
        else if(usTemp_Allowed == NO_OBJECT)
        {
            vLCD_Disp_Ip(gszIP_Add);
            gusDisp_Index = DISP_IP_INDEX;
        }
        gusIsNeedDisp = 0;
    }
    vLCD_Disp_Timer_Index();
    sRead_But();
}
delay(DELAY_TEMP_SUCCESS);
vLED_Control(SCAN_NO_OBJECT);

```

```

while(usTempExit == 1)
{
    usExit = 0;
    usFace_Distance = usRead_Distance();

    if((usFace_Distance > MINIMUM_DISTANCE) && (usFace_Distance <
MAXIMUM_DISTANCE))
    {
        usStart_Scanning_Count++;
        vDisp_Scanning();
    }
    else
    {
        usStart_Scanning_Count = 0;
    }

    if(usStart_Scanning_Count > START_SCAN_COUNT)
    {
        //optimal distance to scan face
        memset(szData, '\0', sizeof(szData));

        do
        {
            usFace_Distance = usRead_Distance();

            if((usFace_Distance > MINIMUM_DISTANCE) && (usFace_Distance <
MAXIMUM_DISTANCE))
            {
                if(gusIsSend_Request == 0)
                {
                    Serial.println(START_RUN);
                    gusIsSend_Request = 1;
                }

                //Read data from ESP32-CAM
                usData_Len = usRead_Serial_Data(szData, sizeof(szData));
                if(usData_Len > 0)
                {
                    if(szData[0] == '*')
                    {

                        // ESP32-CAM will return mask percent to Arduino
                        sscanf(szData, "%d", &sMask_Percent);

                        usIsNeed_Rescan = 0;
                        gusIsSend_Request = 0;
                        usExit = 1;
                    }
                }
            }
        }
    }
}

```

```

else
{
    usDistance_Error++;
}

if(usDistance_Error > DISTANCE_ERROR_COUNT)
{
    usDistance_Error = 0;
    usIsNeed_Rescan = 1;
    usExit = 1;
}
}while(usExit == 0);

if(usIsNeed_Rescan == 0)
{
    vLCD_Disp_Status(sMask_Percent);

    //if the percentage is higher than MASK_DETECT_TH (80%), door will open
    if(sMask_Percent >= MASK_DETECT_TH)
    {
        usTempExit = 0;
        vDoor_Control(ON);
    }
    else
    {
        vDoor_Control(OFF);
    }
}
else if(usIsNeed_Rescan == 1)
{
    usError_Count++;
    if(usError_Count > ERROR_COUNT)
    {
        usTempExit = 0;
    }
    else
    {
        short a = 0;
        vLCD_Disp_Error_Scan();
        vLED_Control(SCAN_ERROR);
        memset(szData, '\0', sizeof(szData));
        Serial.println(RESTART_ESP32);

        do
        {

            //Read data from ESP32-CAM, and dump previous data.
            usData_Len = usRead_Serial_Data(szData, sizeof(szData));
            if(usData_Len > 0)

```

```

        {
            gusIsSend_Request = 0;
            for(short i=0; i<usData_Len; i++)
            {
                if(szData[i] == '#')
                {
                    i++;
                    while(szData[i] != ',')
                    {
                        gszIP_Add[a] = szData[i++];
                        a++;
                    }
                    usExit = 0;
                    break;
                }
            }
        }while(usExit == 1);
    }
}
else
{
    vLED_Control(SCAN_NO_OBJECT);

    if(gusIsNeedDisp == 1)
    {
        vLCD_Disp_Scan_Face(usFace_Distance);
        gusDisp_Index = DISP_DISTANCE_ERROR_INDEX;
        gusIsNeedDisp = 0;
    }
}
vLCD_Disp_Timer_Index();
sRead_But();
}
}

void vDisp_Scanning()
{
    lcd.clear();
    lcd.print("SCANNING...");
    lcd.setCursor(0,1);
    lcd.print("Pls wait & hold.");
}
//PERCENTAGE OF FACE DETECTED, MORE FACE DETECTED, LOW
PERCENTAGE, NO MASK
void vLCD_Disp_Status(short sMask_Percent)
{
    lcd.clear();
    lcd.print("Mask: ");

```

```

lcd.print(sMask_Percent);
lcd.print("%");
lcd.setCursor(0,1);

if(sMask_Percent >= MASK_DETECT_TH)
{
    lcd.print("Enter Allowed.");
}
else
{
    lcd.print("PLEASE WEAR MASK");
}
}
void vDisp_Temp_Sensor(short usTemp_Allowed, float fTemperature_Object)
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Body Temp: ");
    lcd.print(fTemperature_Object);
    lcd.print(char(223));
    lcd.print("C");
    lcd.setCursor(0,1);

    if(usTemp_Allowed == TEMP_ALLOWED)
    {
        lcd.print("Body Temp OK");
    }
    else if(usTemp_Allowed == TEMP_HIGH)
    {
        lcd.print("Body Temp HIGH");
    }
    else if(usTemp_Allowed == TEMP_LOW)
    {
        lcd.print("Body Temp LOW");
    }
}
void vLCD_Disp_Ip(char *szIp)
{
    lcd.clear();
    if(gusLCD_Index == 0)
    {
        lcd.setCursor(2,0);
        lcd.print("PLEASE SCAN");
        lcd.setCursor(2,1);
        lcd.print("TEMPERATURE");
    }
    else
    {
        lcd.setCursor(1,0);
        lcd.print("CONNECTED TO:");
    }
}

```



```

        lcd.setCursor(1,1);
        lcd.print(szIp);
    }
}
void vLCD_Displ_Scan_Face(unsigned short usFace_Distance)
{
    lcd.clear();
    if(gusLCD_Index == 0)
    {
        lcd.setCursor(0,0);
        lcd.print("Pls Scan ur face");
        lcd.setCursor(0,1);
        lcd.print(MINIMUM_DISTANCE);
        lcd.print("-");
        lcd.print(MAXIMUM_DISTANCE);
        lcd.print("cm from cam");
    }
    else if(gusLCD_Index == 1)
    {
        lcd.setCursor(0,0);
        lcd.print("youre ");
        lcd.print(usFace_Distance);
        lcd.print("cm away");

        lcd.setCursor(0,1);
        if(usFace_Distance > MAXIMUM_DISTANCE)
        {
            lcd.print("Move Closer.. ");
        }
        else if(usFace_Distance < MINIMUM_DISTANCE)
        {
            lcd.print("Take a step back..");
        }
    }
}

void vDisp_Face_Distance_Error(unsigned short usFace_Distance)
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("youre ");
    lcd.print(usFace_Distance);
    lcd.print("cm away");

    lcd.setCursor(0,1);
    if(usFace_Distance > MAXIMUM_DISTANCE)
    {
        lcd.print("Move Closer.. ");
    }
}

```

```

else if(usFace_Distance < MINIMUM_DISTANCE)
{
    lcd.print("Take a step back..");
}
}
void vLCD_Dispatch_Error_Scan()
{
    lcd.clear();
    lcd.print("Take a step back");
    lcd.setCursor(0,1);
    lcd.print("and rescan again");
}
void vLED_Control(short sScan_Status)
{
    if(sScan_Status == SCAN_PASS)
    {
        digitalWrite(LED_GREEN,HIGH);
        digitalWrite(LED_RED,LOW);
    }
    else if(sScan_Status == SCAN_ERROR)
    {
        digitalWrite(LED_GREEN,LOW);
        digitalWrite(LED_RED,HIGH);
    }
    else
    {
        digitalWrite(LED_GREEN,LOW);
        digitalWrite(LED_RED,LOW);
    }
}
void vLCD_Dispatch_Timer_Index()
{
    if((millis() - gulStart_Timer_LCD) >= 1000)
    {
        gusLCD_Index++;

        if(gusDisp_Index == DISP_IP_INDEX)
        {
            if(gusLCD_Index > 1)
            {
                gusLCD_Index = 0;
            }
        }
        else if(gusDisp_Index == DISP_DISTANCE_ERROR_INDEX)
        {
            if(gusLCD_Index > 1)
            {
                gusLCD_Index = 0;
            }
        }
    }
}

```

```

    gusIsNeedDisp = 1;
    gulStart_Timer_LCD = millis();
}
}
short sRead_But()
{
    short sBut_Status = 0;
    sBut_Status = digitalRead(PUSH_BUTTON_PIN);

    if(sBut_Status == HIGH)
    {
        vDoor_Control(ON);
        gusLCD_Index = 0;
        gulStart_Timer_LCD = millis();
        gusIsNeedDisp = 1;
    }
}
void vDoor_Control(short sOnOff)
{
    if(sOnOff == ON)
    {
        vLED_Control(SCAN_PASS);
        vServo_Control(OPEN_DOOR);
        delay(OPEN_DOOR_TIME);

        vServo_Control(CLOSE_DOOR);
        delay(CLOSE_DOOR_TIME);
    }
    else
    {
        vServo_Control(DOOR_LOCK);
        vLED_Control(SCAN_ERROR);
        delay(OPEN_DOOR_TIME);
    }
}
unsigned short usRead_Distance()
{
    unsigned short usFace_Distance = 0;
    usFace_Distance = ultrasonic.read();
    delay(50);
    return usFace_Distance;
}

//CONTROL SERVO BASED ON RESULTS
void vServo_Control(int sDoor_Status)
{
    myServo.attach(SERVO_PIN);
    if(sDoor_Status == OPEN_DOOR) //open door
    {

```

```

for(int i = CLOSE_DOOR; i > OPEN_DOOR; i--)
{
    myServo.write(i);
    delay(20);
}
}
else if(sDoor_Status == CLOSE_DOOR)
{
    for(int i = OPEN_DOOR; i < CLOSE_DOOR; i++)
    {
        myServo.write(i);
        delay(20);
    }
}
else if(sDoor_Status == DOOR_LOCK)
{
    myServo.write(CLOSE_DOOR);
}

myServo.detach();
}

//READ SERIAL DATA FROM ESP32-CAM
unsigned short usRead_Serial_Data(char *szData, short sDataSize)
{
    short i=0;
    if(Serial.available())
    {
        *(szData+i) = Serial.read();
        i++;
        delay(2);

        while(Serial.available())
        {
            *(szData+i) = Serial.read();
            i++;

            if(i >= sDataSize)
            {
                break;
            }
            delay(2);
        }
    }
    Serial.flush();

    return i;
}

```

## **Program code**

### **ESP32 cam code:**

```
//Libraries
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h" //disable brownout problems
#include "esp_http_server.h"
#include "fd_forward.h"
#include "fr_forward.h"

//Replace with your network credentials
const char* ssid = "Abhi"; // wifi network name
const char* password = "wifi9245124"; // wifi password

//Define constant values
#define PART_BOUNDARY "1234567890000000000000987654321"
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27

#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22

#define FACE_COLOR_RED 0x000000FF
#define FACE_COLOR_GREEN 0x0000FF00
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)

#define DETECT_FACE_TIME 2000
#define NUM_FRAME 10
#define PERCENT_WEAR_MASK_TH 80
```

```

#define START_RUN 'a'
#define RESTART_ESP32 'r'
#define RESTART_SCAN 'n'

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;

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";

httpd_handle_t stream_httpd = NULL;
static mtmn_config_t mtmn_config = {0};
static int8_t detection_enabled = 1;
static ra_filter_t ra_filter;

short gsFace_Detected = 0;
short gsIsInStream = 0;
short gusRestart_Scan = 0;
unsigned long gulStart_IsInStream_Reset_Timer = 0;
short gsArrayIsFaceDetect[NUM_FRAME] = {0,0,0,0,0,0,0,0,0,0};
unsigned short gusIsMask_Detect = 0;
unsigned short gusFrame_Count = 0;
unsigned short gusIsStart = 0;

void setup()
{
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);    //disable brownout
    detector

    Serial.begin(9600);
    Serial.setDebugOutput(false);

    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;
config.frame_size = FRAMESIZE_QVGA;
config.jpeg_quality = 10;
config.fb_count = 1;

// 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();
s->set_vflip(s, 1);//flip camera
//s->set_brightness(s, 1);//up the blightness just a bit
//s->set_saturation(s, -2);//lower the saturation

// Wi-Fi connection
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

delay(100);
digitalWrite(4, LOW);

Serial.print("#");
Serial.print(WiFi.localIP());
Serial.println(",");

mtmn_config = mtmn_init_config();
// Start streaming web server
startCameraServer();
}

```

```

void loop()
{
    // put your main code here, to run repeatedly:
    short sNum_Detect = 0;
    short sPercent_Wear_Mask = 0;
    char szData[10];
    char szTemp[10];

    memset(szData, '\0', sizeof(szData));
    if(usRead_Serial_Data(szData, sizeof(szData)) > 0)           //read data request from
    Arduino
    {
        if(szData[0] == START_RUN) // start face_detection algorithm
        {
            gusIsStart = 1;
            gusRestart_Scan = 0;
            gusFrame_Count = 0;
        }
        else if(szData[0] == RESTART_ESP32)    // restart ESP32-CAM in case of ESP32-
        CAM hung up
        {
            ESP.restart();
        }
        else if(szData[0] == RESTART_SCAN)
        {
            gusRestart_Scan = 1;
        }
    }
    if(gsIsInStream == 1)
    {
        if(millis() - gulStart_IsInStream_Reset_Timer > DETECT_FACE_TIME)
        {
            gsIsInStream = 0;
        }
    }
    else if(gsIsInStream == 0)
    {
        if(gusIsStart == 1)
        {
            vFace_Detection_Offline();
        }
    }
    if(gusFrame_Count >= NUM_FRAME) // scanning for 10 times
    {
        for(short i=0; i<NUM_FRAME; i++)
        {
            if(gsArrayIsFaceDetect[i] == 0)
            {
                sNum_Detect++; // number of face not detected
            }
        }
    }
}

```



```

//calculate the percentage of face not detected
sPercent_Wear_Mask = map(sNum_Detect, 0, NUM_FRAME, 0, 100);

if(gusRestart_Scan == 1)
{
    Serial.println("\0");
}
else
{
    memset(szTemp, '\0', sizeof(szTemp));
    sprintf(szTemp, "%d", sPercent_Wear_Mask);
    Serial.println(szTemp); //send percentage number to
Arduino
}
gusFrame_Count = 0;
gusIsStart = 0;
}
}

void startCameraServer()
{
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    config.server_port = 80;
    httpd_uri_t index_uri = {
        .uri      = "/",
        .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;

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

```

```

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;

    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)
    {
        gsIsInStream = 1;
        gulStart_IsInStream_Reset_Timer = millis();                //restart timer

        detected = false;
        face_id = 0;
        fb = esp_camera_fb_get();

        if (!fb)
        {
            Serial.println("Camera capture failed");
            res = ESP_FAIL;
        }
        else
        {
            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
        {
            box_array_t *net_boxes = NULL;

            if(gusIsStart == 1)
            {
                net_boxes = face_detect(image_matrix, &mtmn_config);
                //gusFrame_Count++;
            }

            if (net_boxes || fb->format != PIXFORMAT_JPEG)
            {
                if(net_boxes)
                {
                    detected = true;
                    draw_face_boxes(image_matrix, net_boxes);
                    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;
}
}
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;
}

if(detected == 1)

```

```

    {
        gsFace_Detected = 1; //detect face
    }
    else
    {
        gsFace_Detected = 0;
    }

    if(gusIsStart == 1)
    {
        if(gusFrame_Count < NUM_FRAME)
        {
            gsArrayIsFaceDetect[gusFrame_Count] = gsFace_Detected;
        }
        else
        {
            gusIsStart = 0; //Reset
        }

        gusFrame_Count++;

    }
    delay(1);
}

last_frame = 0;
return res;
}
void vFace_Detection_Offline()
{
    camera_fb_t * frame;
    frame = esp_camera_fb_get();

    dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, frame->width, frame->height,
3);
    fmt2rgb888(frame->buf, frame->len, frame->format, image_matrix->item);

    esp_camera_fb_return(frame);

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

    if(boxes)
    {
        gsFace_Detected = 1;
        free(boxes->score);
        free(boxes->box);
        free(boxes->landmark);
        free(boxes);
    }
}

```

```

else
{
    gsFace_Detected = 0;
}

dl_matrix3du_free(image_matrix);
if(gusFrame_Count < NUM_FRAME)
{
    gsArrayIsFaceDetect[gusFrame_Count] = gsFace_Detected;
}
gusFrame_Count++;
}
static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes)
{
    int x, y, w, h, i;
    uint32_t color = FACE_COLOR_YELLOW;
    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];
            fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
        }
    #endif
}

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;
}

unsigned short usRead_Serial_Data(char *szData, short sDataSize)
{
    short i=0;

    if(Serial.available())
    {
        *(szData+i) = Serial.read();
        i++;
        delay(2);

        while(Serial.available())
        {
            *(szData+i) = Serial.read();
            i++;

            if(i >= sDataSize)
            {
                break;
            }
            delay(2);
        }
    }
    return i;
}

```