```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Ultrasonic.h>
#include <Adafruit_MLX90614.h>

#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 BUZZER                     3
#define PLAY                       5

#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         7000
#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                3
#define TEMP_MAX_ALLOWABLE         37
#define TEMP_THRESHOLD             32

#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

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display
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(PLAY, OUTPUT);
 pinMode(BUZZER, OUTPUT);
 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((millis() - gulRestart_Timer) > RESTART_TIME)
   {
    Serial.println(RESTART_SCAN);
```

```
      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;
    usTempExit = 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;
     }

     usData_Len = usRead_Serial_Data(szData, sizeof(szData)); //Read data from ESP32-CAM
     if(usData_Len > 0)
     {
      if(szData[0] == '*')
      {
       sscanf(szData, "*%d,", &sMask_Percent);// ESP32-CAM will return mask percent to Arduino

       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(sMask_Percent >= MASK_DETECT_TH) //if the percentage is higher than MASK_DETECT_TH
(80%), door will open
       {
        usTempExit = 0;
        vDoor_Control(ON);
       }
       else
       {
        vDoor_Control(OFF);
        usError_Count++;

        if(usError_Count >= ERROR_COUNT)
        {
         usTempExit = 0;
        }
       }
      }
      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);
        gulRestart_Timer = millis();

        do
        {
         usData_Len = usRead_Serial_Data(szData, sizeof(szData)); //Read data from ESP32-CAM, and
dump previous data.
         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;
              }
            }
          }
          else
          {
            if((millis() - gulRestart_Timer) > RESTART_TIME)
            {
              Serial.println(RESTART_SCAN);
              gulRestart_Timer = millis();
            }
          }
        }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");
  digitalWrite(PLAY, HIGH);
  delay(500);
  digitalWrite(PLAY, LOW);
  delay(500);
 }
}
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");
  delay(7000);
 }
 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_Disp_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_Disp_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);
    digitalWrite(BUZZER,HIGH);
    delay(1000);
    digitalWrite(BUZZER,LOW);
    delay(1000);
  }
  else if(sScan_Status == SCAN_ERROR)
  {
    digitalWrite(LED_GREEN,LOW);
    digitalWrite(LED_RED,HIGH);
    digitalWrite(BUZZER,HIGH);
    delay(3000);
    digitalWrite(BUZZER,LOW);
    delay(3000);
  }
  else
  {
    digitalWrite(LED_GREEN,LOW);
    digitalWrite(LED_RED,LOW);
  }
}
void vLCD_Disp_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;
}
```

```
#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 = "dudud";
const char* password = "12345678";

#define PART_BOUNDARY     "123456789000000000000987654321"
#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)
    {
      Serial.print("#");
      Serial.print(WiFi.localIP());
      Serial.println(",");
    }
  }

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

    if(gusRestart_Scan == 1)
    {
      Serial.print("#");
      Serial.print(WiFi.localIP());
      Serial.println(",");
    }
    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;
}
```

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji : Slamet Widodo, M.Kom.
Nama Mahasiswa : Andika Wijaya Pradarma
NIM : 062030701727
Jurusan /Program Studi : D3 Teknik Komputer
Judul LA/ Skripsi : Rancang Bangun Alat Pendeteksi Masker dan Pemindai Suhu Tubuh Menggunakkan Arduino Uno dan Esp 32 CAM

No	Uraian	Paraf
1)	Sensor suhu dijauhi dulam	
	Sensor ponag lagi	
2)	Sensor pendeteksi masker	
	dulam jelas	
3)	Revisi Alat sesuai fungsi	

Palembang,
Dosen Penguji,

Slamet Widodo, M.Kom.
NIP. 197305162002121001

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji : Arsia Rini, S.Kom., M.Kom.
Nama Mahasiswa : Andika Wijaya Pradarma
NIM : 062030701727
Jurusan /Program Studi : D3 Teknik Komputer
Judul LA/ Skripsi : Rancang Bangun Alat Pendeteksi Masker dan Pemindai Suhu Tubuh Menggunakkan Arduino Uno dan Esp 32 CAM

No	Uraian	Paraf
	tata tulis Format penulisan	
1.	Penelitian terdahulu	
2.	Permasalahan dilatar belakang	
3	flowthart	
4.	langkah pembuatan Alat	

Palembang,
Dosen Penguji,

Arsia Rini, S.Kom., M.Kom.
NIP.198809222020122014

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji : Isnainy Azro, M.Kom.
Nama Mahasiswa : Andika Wijaya Pradarma
NIM : 062030701727
Jurusan /Program Studi : D3 Teknik Komputer
Judul LA/ Skripsi : Rancang Bangun Alat Pendeteksi Masker dan Pemindai Suhu Tubuh Menggunakkan Arduino Uno dan Esp 32 CAM

No	Uraian	Paraf
1	Idem dng penguji lain	

Palembang,
Dosen Penguji,

Isnainy Azro, M.Kom.
NIP. 197310012002122007

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji	: Ikhthison Mekongga, S.T., M.Kom.
Nama Mahasiswa	: Andika Wijaya Pradarma
NIM	: 062030701727
Jurusan /Program Studi	: D3 Teknik Komputer
Judul LA/ Skripsi	: Rancang Bangun Alat Pendeteksi Masker dan Pemindai Suhu Tubuh Menggunakkan Arduino Uno dan Esp 32 CAM

No	Uraian	Paraf
	Tata tulis format penulisan	

Palembang,
Dosen Penguji,

Ikhthison Mekongga, S.T., M.Kom.
NIP. 197705242000031002

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI POLITEKNIK
NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN PROPOSAL LAPORAN TUGAS AKHIR

Nama Mahasiswa	:	Andika Wijaya Pradarma
NIM	:	062030701727
Jurusan/Program Studi	:	Teknik Komputer/D3 Teknik Komputer
Dosen Pembimbing	:	Herlambang Saputra,Ph.D
Judul	:	Rancang bangun alat pendeteksi masker dengan pemindai temperatur tubuh dan servo menggunakkan ESP32-CAM dan Arduino Uno

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	1-3-24	Acc Judul.	
2.	8-3-24	Revisi Bab I.	
3.	14-3-23	Revisi Bab I.	
4.	16-3-24	Revisi Bab I	
5.	17-3-23	Acc Bab I.	
6.	21-3-24	Revisi Bab II.	
7.	24-3-24	Revisi Bab II	
8.	28-3-24	Revisi Bab II.	
9.	31-3-24	Acc Bab II.	
10.	4-4-24.	Revisi Bab III.	
11.	8-4-24.	Revisi Bab III	
12.	11-4-23	Revisi Bab IV	
13.	14-4-23.	Revisi Bab III.	
14.	28-4-24	Acc Bab III & IV.	

Palembang, 2023
Mengetahui,
Ketua Jurusan

Azwardi,S.T.,M.T
NIP. 19700523200501004

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI POLITEKNIK
NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN PROPOSAL LAPORAN TUGAS AKHIR

Nama Mahasiswa	:	Andika Wijaya Pradarma
NIM	:	062030701727
Jurusan/Program Studi	:	Teknik Komputer/D3 Teknik Komputer
Dosen Pembimbing	:	Herlambang Saputra,Ph.D
Judul	:	Rancang bangun alat pendeteksi masker dengan pemindai temperatur tubuh dan servo menggunakkan ESP32-CAM dan Arduino Uno

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	5-5-25	Revisi Daftar Pustaka.	
2.	10-5-25	Acc Proposal.	

Palembang, 2023
Mengetahui,
Ketua Jurusan

Azwardi,S.T.,M.T
NIP. 19700523200501004

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN TUGAS AKHIR

Nama Mahasiswa	:	Andika Wijaya Pradarma
NIM	:	062030701727
Jurusan/Program Studi	:	Teknik Komputer/D3 Teknik Komputer
Dosen Pembimbing	:	Herlambang Saputra,Ph.D
Judul	:	Rancang bangun alat pendeteksi masker dan pemindai suhu tubuh menggunakkan ESP32-CAM dan Arduino Uno

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
15	20-7-23	Revisi Bab IV.	
16.	25-7-23	Revisi Bab IV.	
17.	26-7-23	Revisi Bab IV & V.	
18.	8-8-23	Acc Bab IV & V Acc Ujian (Rekomendasi).	

Palembang, 2023
Mengetahui,
Ketua Jurusan

Azwardi,S.T.,M.T
NIP. 19700523200501004

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN TUGAS AKHIR

Nama Mahasiswa	:	Andika Wijaya Pradarma
NIM	:	062030701727
Jurusan/Program Studi	:	Teknik Komputer/D3 Teknik Komputer
Dosen Pembimbing	:	Ica Admirani,S.Kom,M.Kom
Judul	:	Rancang bangun alat pendeteksi masker dan pemindai suhu tubuh menggunakkan ESP32-CAM dan Arduino Uno

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	10/7-2023	- Perbaiki gambar	
2.	31/7-2023	- Acc Alat - Acc Laporan	

Palembang, 2023
Mengetahui,
Ketua Jurusan

Azwardi,S.T.,M.T
NIP. 19700523200501004

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

PELAKSANAAN REVISI UJIAN TUGAS AKHIR

Nama Mahasiswa : Andika Wijaya Pradarma
NIM : 062030701727
Jurusan /Program Studi : D3 Teknik Komputer
Judul LA/ Skripsi : Rancang Bangun Alat Pendeteksi Masker dan Pemindai Suhu Tubuh Menggunakkan Arduino Uno dan Esp 32 CAM

Telah melaksanakan revisi terhadap Laporan Tugas Akhir yang diujikan pada hari ..... tanggal ..... bulan ..... tahun ..... Pelaksanaan revisi terhadap Laporan Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No	Komentar	Nama Dosen Penguji	Tanggal/ bulan	Tanda Tangan
1.	Acc	Slamet Widodo, M.Kom.	23/08/'23	
2.	Acc	Ikhthison Mekongga, S.T., M.Kom.	29/08/23	
3.	Acc	Isnainy Azro, M.Kom.	29/08/23	
4.	Acc	Arsia Rini, S.Kom., M.Kom.	22/8/23	

Palembang,
Ketua Penguji

Slamet Widodo, M.Kom.
NIP. 197305162002121001

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REKOMENDASI UJIAN TUGAS AKHIR

Pembimbing Laporan Tugas Akhir, memberikan rekomendasi ujian laporan tugas akhir kepada,

Nama Mahasiswa	:	Andika Wijaya Pradarma
NIM	:	062030701727
Jurusan/Program Studi	:	Teknik Komputer/D3 Teknik Komputer
Judul Tugas Akhir	:	Rancang bangun alat pendeteksi masker dan suhu tubuh menggunakkan esp 32 cam dan Arduino Uno

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Laporan Tugas Akhir, pada Tahun Akademik 2023/2024

Palembang, 2023

Disetujui oleh,

Pembimbing I	Pembimbing II
Herlambang Saputra,Ph.D	**Ica Admirani,S.Kom,M.Kom**
NIP.198103182008121002	**NIP.197903282005012001**