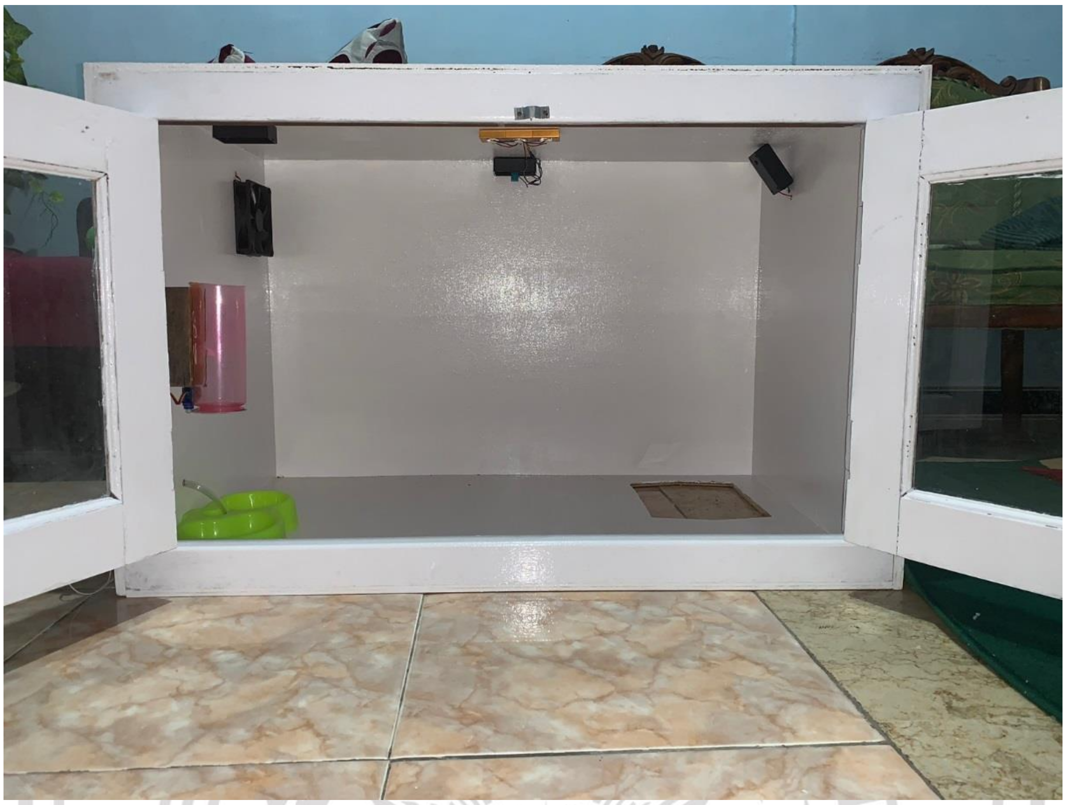
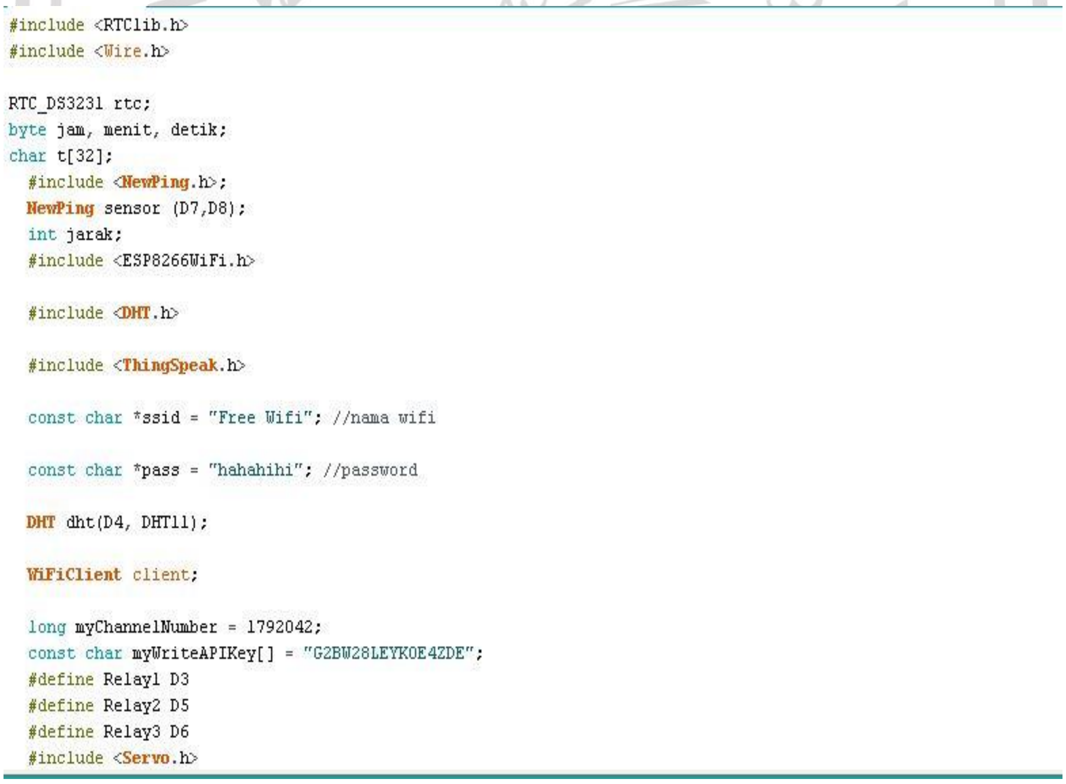
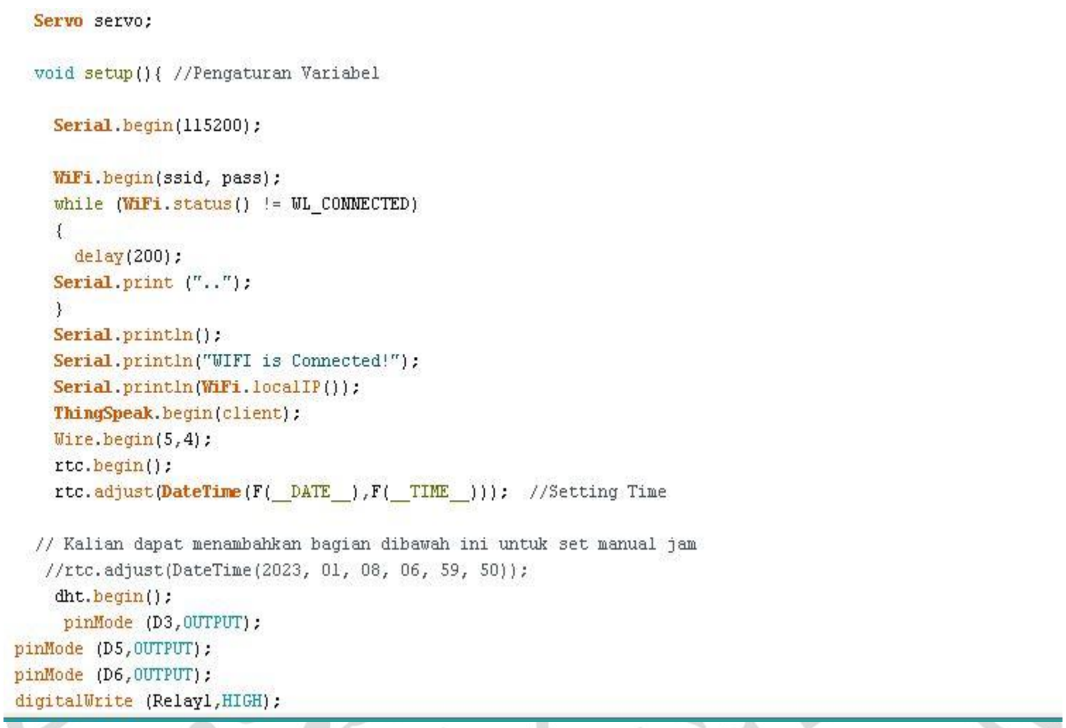
LAMPIRAN

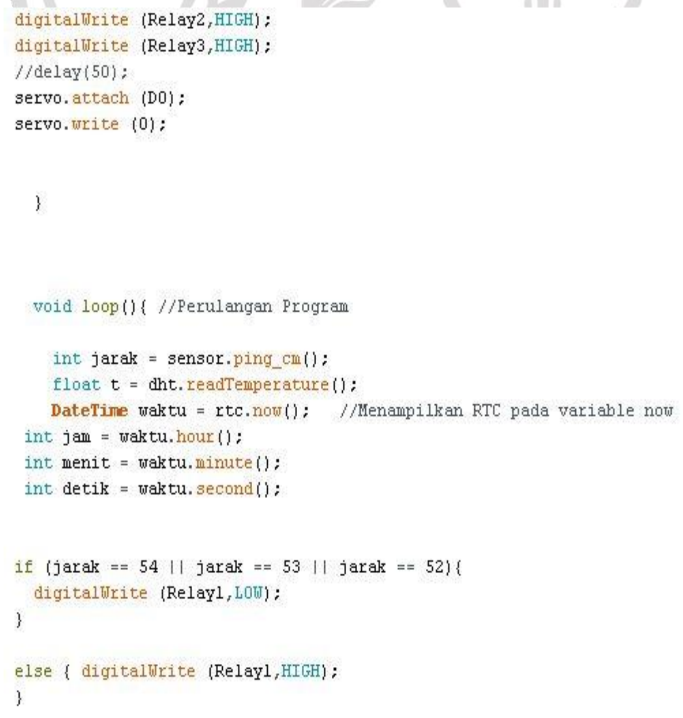




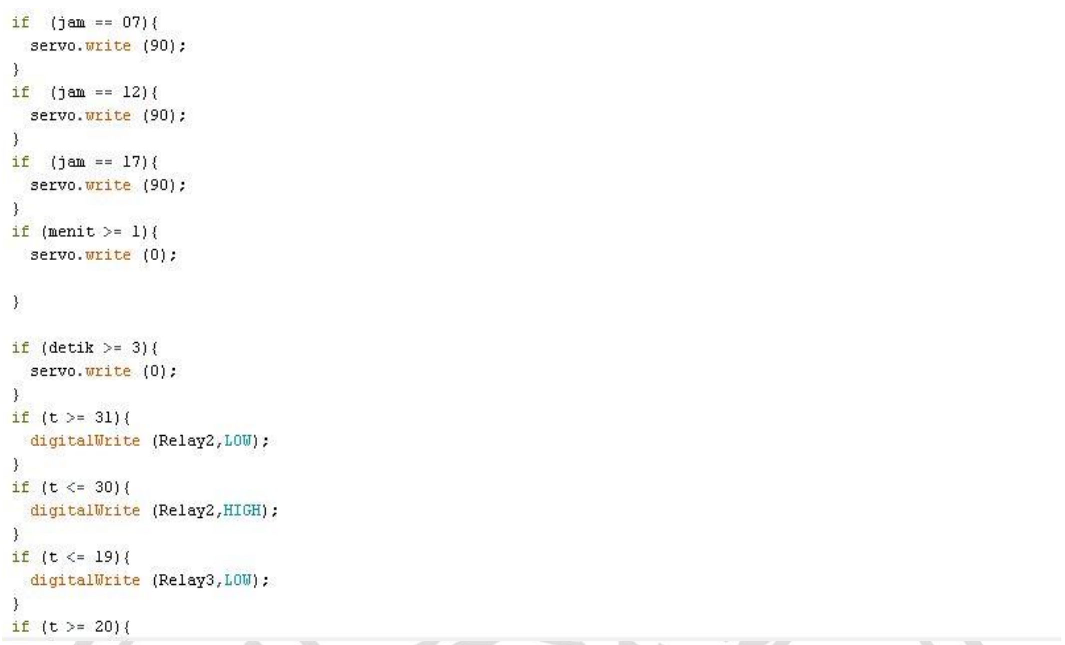
```
#include <RTClib.h>
#include <Wire.h>

RTC_DS3231 rtc;
byte jam, menit, detik;
char t[32];
  #include <NewPing.h>;
  NewPing sensor (D7,D8);
  int jarak;
  #include <ESP8266WiFi.h>

  #include <DHT.h>

  #include <ThingSpeak.h>

  const char *ssid = "Free Wifi"; //nama wifi

  const char *pass = "hahahihi"; //password

  DHT dht(D4, DHT11);

  WiFiClient client;

  long myChannelNumber = 1792042;
  const char myWriteAPIKey[] = "G2BW28LEYKOE4ZDE";
  #define Relay1 D3
  #define Relay2 D5
  #define Relay3 D6
  #include <Servo.h>
```



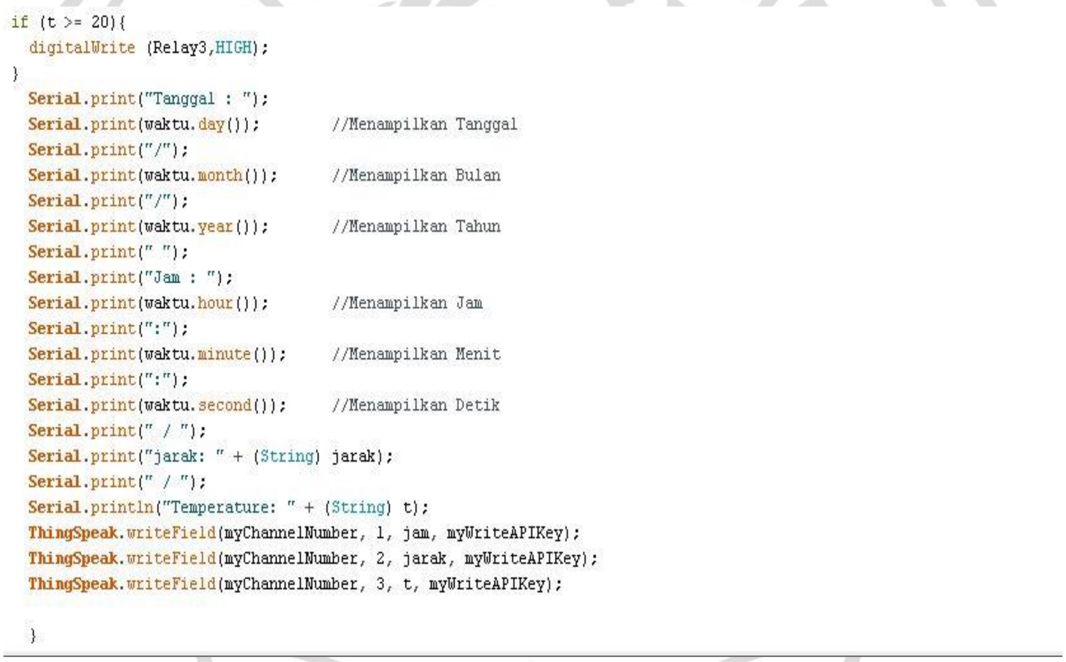
```
  Servo servo;

  void setup(){ //Pengaturan Variabel

    Serial.begin(115200);

    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
      delay(200);
    Serial.print ("..");
    }
    Serial.println();
    Serial.println("WIFI is Connected!");
    Serial.println(WiFi.localIP());
    ThingSpeak.begin(client);
    Wire.begin(5,4);
    rtc.begin();
    rtc.adjust(DateTime(F(__DATE__),F(__TIME__)));  //Setting Time

  // Kalian dapat menambahkan bagian dibawah ini untuk set manual jam
   //rtc.adjust(DateTime(2023, 01, 08, 06, 59, 50));
    dht.begin();
     pinMode (D3,OUTPUT);
pinMode (D5,OUTPUT);
pinMode (D6,OUTPUT);
digitalWrite (Relay1,HIGH);
```



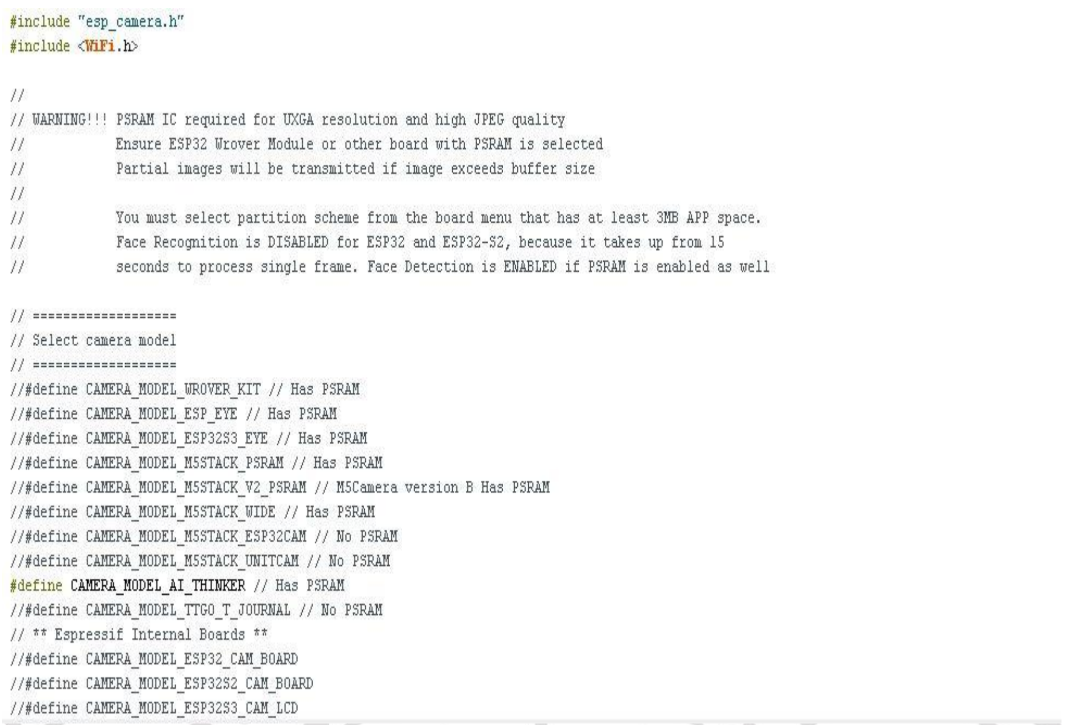
```
digitalWrite (Relay2,HIGH);
digitalWrite (Relay3,HIGH);
//delay(50);
servo.attach (D0);
servo.write (0);


  }



  void loop(){ //Perulangan Program

    int jarak = sensor.ping_cm();
    float t = dht.readTemperature();
    DateTime waktu = rtc.now();   //Menampilkan RTC pada variable now
 int jam = waktu.hour();
 int menit = waktu.minute();
 int detik = waktu.second();


if (jarak == 54 || jarak == 53 || jarak == 52){
  digitalWrite (Relay1,LOW);
}

else { digitalWrite (Relay1,HIGH);
}
```



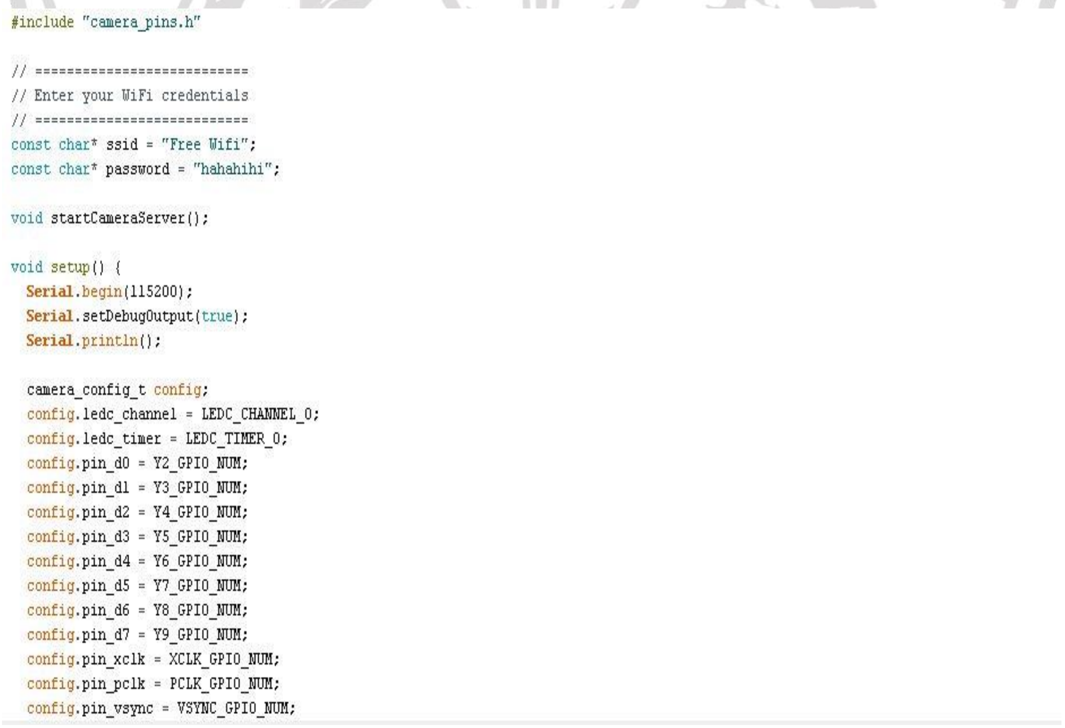
```
if  (jam == 07){
  servo.write (90);
}
if  (jam == 12){
  servo.write (90);
}
if  (jam == 17){
  servo.write (90);
}
if (menit >= 1){
  servo.write (0);

}

if (detik >= 3){
  servo.write (0);
}
if (t >= 31){
  digitalWrite (Relay2,LOW);
}
if (t <= 30){
  digitalWrite (Relay2,HIGH);
}
if (t <= 19){
  digitalWrite (Relay3,LOW);
}
if (t >= 20){
```



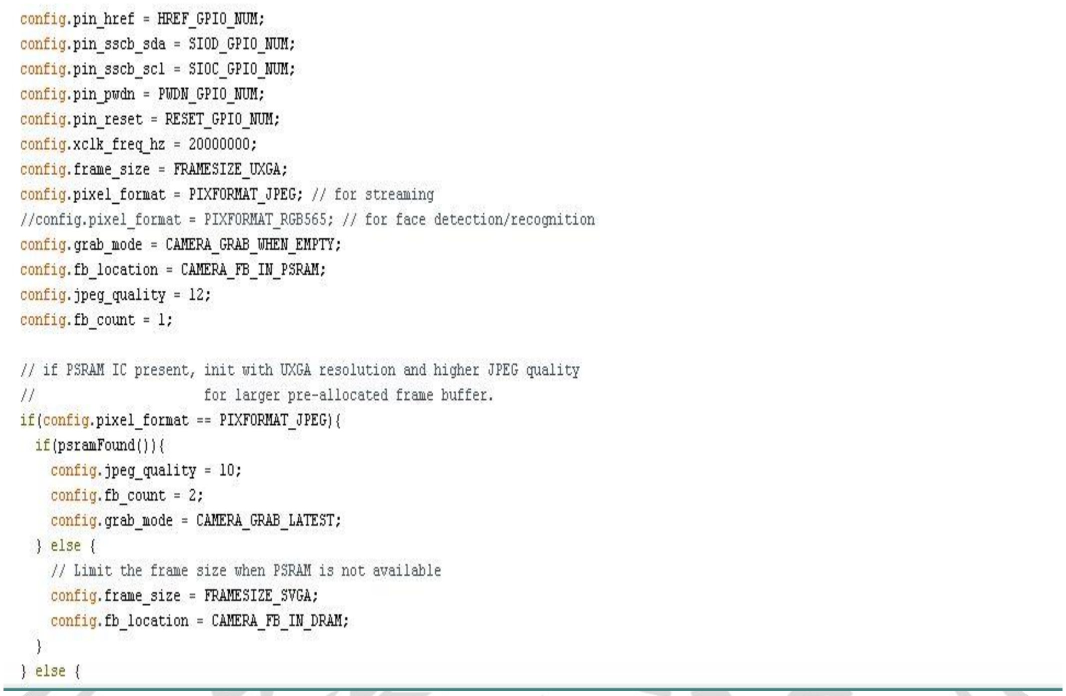
```
if (t >= 20){
  digitalWrite (Relay3,HIGH);
}
  Serial.print("Tanggal : ");
  Serial.print(waktu.day());        //Menampilkan Tanggal
  Serial.print("/");
  Serial.print(waktu.month());      //Menampilkan Bulan
  Serial.print("/");
  Serial.print(waktu.year());       //Menampilkan Tahun
  Serial.print(" ");
  Serial.print("Jam : ");
  Serial.print(waktu.hour());       //Menampilkan Jam
  Serial.print(":");
  Serial.print(waktu.minute());     //Menampilkan Menit
  Serial.print(":");
  Serial.print(waktu.second());     //Menampilkan Detik
  Serial.print(" / ");
  Serial.print("jarak: " + (String) jarak);
  Serial.print(" / ");
  Serial.println("Temperature: " + (String) t);
  ThingSpeak.writeField(myChannelNumber, 1, jam, myWriteAPIKey);
  ThingSpeak.writeField(myChannelNumber, 2, jarak, myWriteAPIKey);
  ThingSpeak.writeField(myChannelNumber, 3, t, myWriteAPIKey);

  }
```



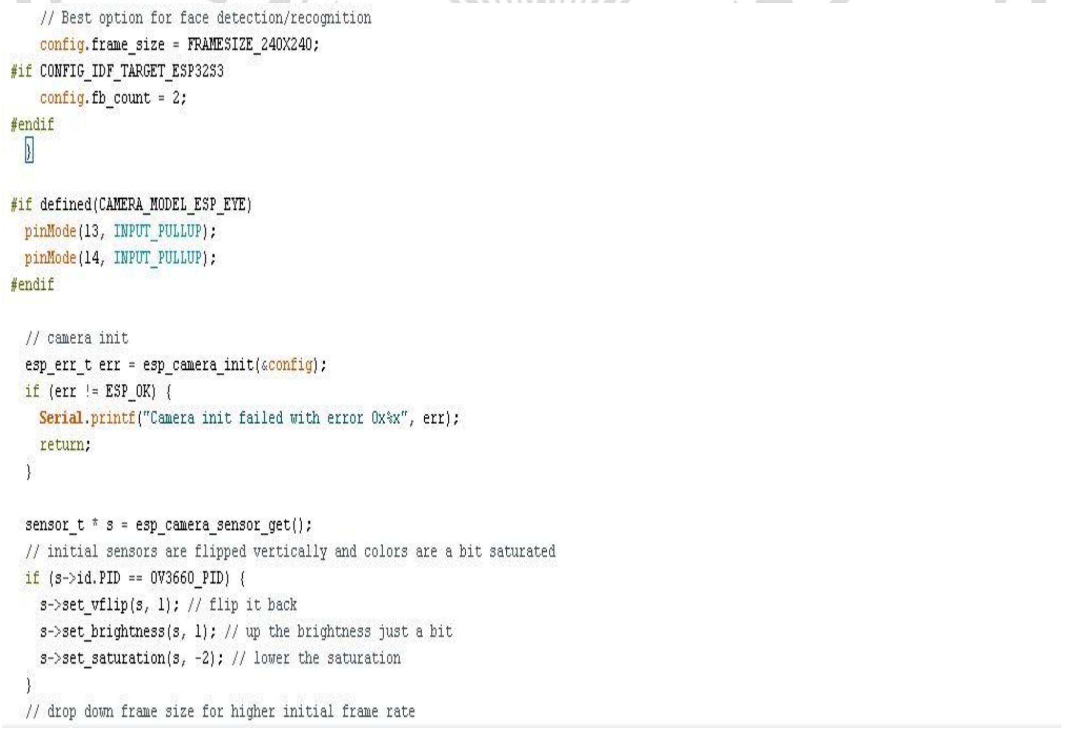
```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//
//            You must select partition scheme from the board menu that has at least 3MB APP space.
//            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes up from 15
//            seconds to process single frame. Face Detection is ENABLED if PSRAM is enabled as well

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
```



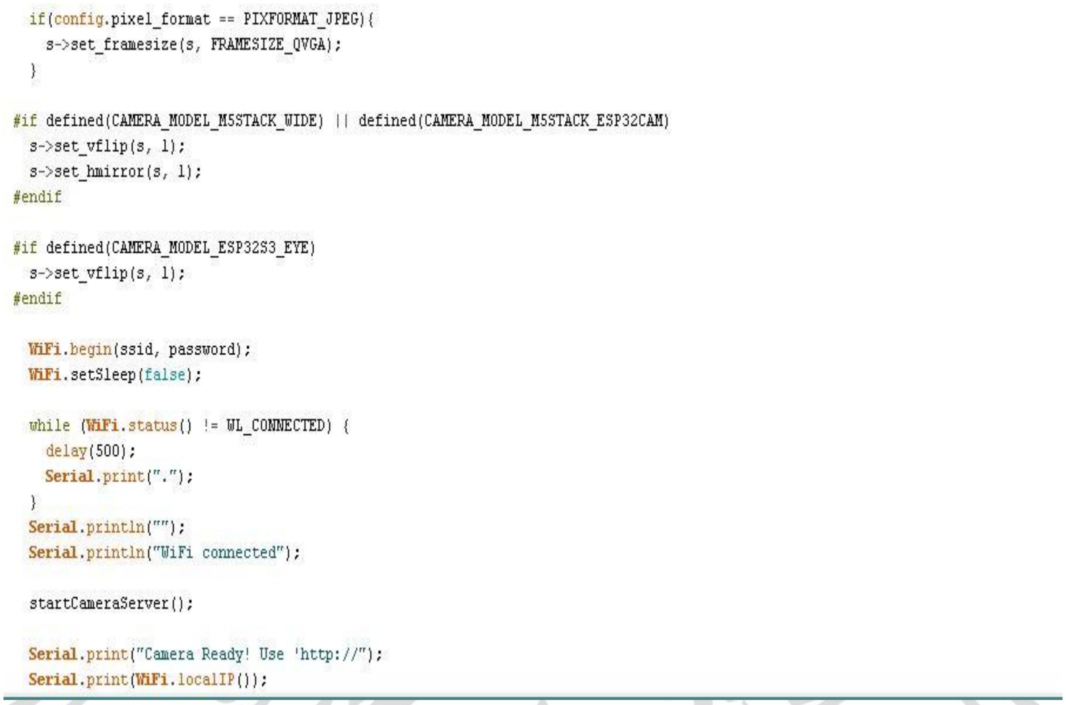
```
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid = "Free Wifi";
const char* password = "hahahihi";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
```



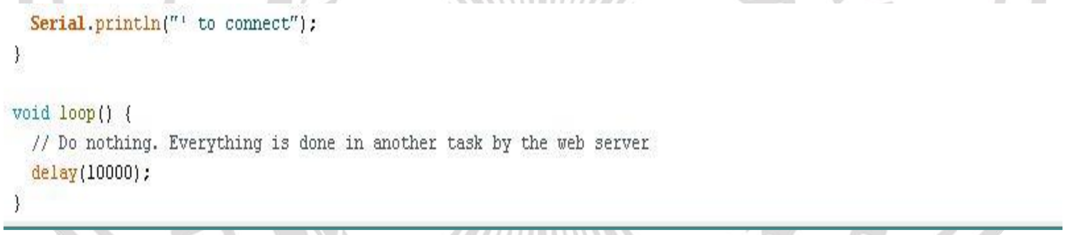
```
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

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



```
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

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



```
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
```



```
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```