به نام خدا

. پروژه درس ریز پردازنده و زبان اسمبلی

عنوان پروژه: خانه هوشمند

شماره پروژه : سوم

دانشجو : پارسا کاویان پور

استاد : دکتر سجاد نژاد حسن

تاریخ تحویل: 1404/05/15

بهار 1404

عنوان پروژه: خانه هوشمند

مقدمه :

پروژه خانه هوشمند دارای امکانات متفاوت و کاربردی نظیر حالت اتوماتیک و دستی، باز و بسته نمودن درب، روشن و خاموش کردن کولر در سالن پذیرایی و دو اتاق مجزا، روشن و خاموش نمودن چراغ ها در سالن پذیرایی و دو اتاق مجزا، حالت پارتی مود جهت مهمانی ها و جشن ها، حالات مختلف حفاظتی نظیر حفاظت از بچه که در صورت گریه ی بچه به والدین اطلاع میدهد. حالت حفاظتی از گاز کربن‌مونوکسید و همچنین پنل وب و پنل اپلیکیشن اندروید امن و گزارشات امنیتی.

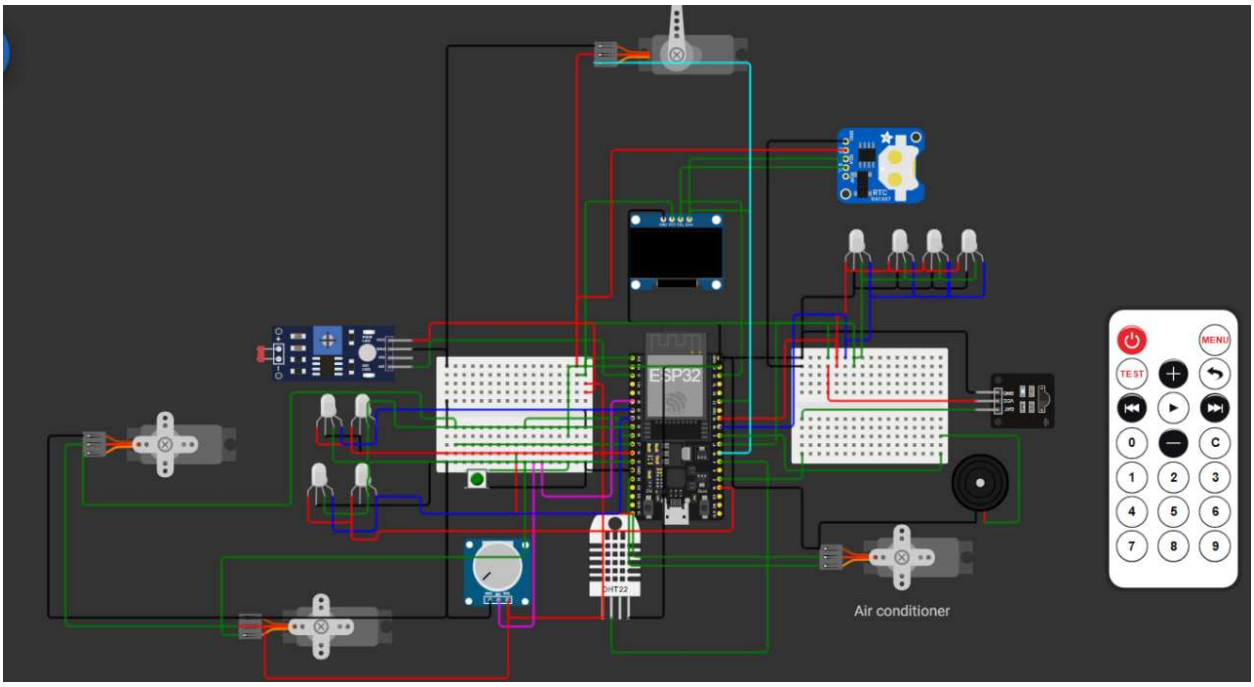
شرح پروژه:

این پروژه دارای یک ریموت کنترل است که فرمان از طریق آن توسط ESP دریافت میشود. دلیل وجود ریموت کنترل دسترسی اسان برای کودکان و بزرگسالان است. خیلی از خانواده ها کودکان زیر 15 سال دارند و شاید والدین‌شان برای آن‌ها موبایل هوشمند را مجاز ندانند اما در این پروژه این گروه هدف نیز در نظر گرفته شده. همچنین پنل وب و اندروید نیز با زبان فلاتر پیاده سازی شده است که امکان کنترل از راه دور را نیز به ارمغان می آورد. لازم به ذکر است تمام تسک ها بصورت همروند اجرا میشوند و در این امر از سیستم عامل بلادرنگ FreeRTOS بهره برده شده است.در ادامه توضیحاتی در خصوص قابلیت ها، نحوه استفاده و توابع مهم ارائه خواهد شد.

قابلیت ها :

1. حالت دستی :
   - روشن و خاموش کردن فن ها(سه فن وجود دارند برای پذیرایی و دو اتاق)
   - روشن و خاموش کردن کامل لامپ ها(سه لامپ وجود دارند برای پذیرایی و دو اتاق)
   - روشن و خاموش کردن پارتی مود در خانه (رقص نور لامپ ها)
   - روشن و خاموش کردن حالت چراغ خواب (نور مخصوص شب)
   - روشن و خاموش کردن حالت حفاظت از بچه
   - در صورتی که بچه گریه کرد با صدای زنگ به پدر مادر اطلاع داده میشود
2. حالت اتوماتیک :
   - کنترل فن ها بر اساس دما (اگر دما بیش از 24 بود فعال میشوند)
   - کنترل نور بر اساس روشنایی محیط (نور لامپ ها ثابت نیست و داینانیک بر اساس نور کنترل میشود)
   - کنترل حالت خواب(چراغ خواب) بر اساس ساعت (پس از 11 شب)
3. قابلیت های مستقل از حالت دستی و اتوماتیک :
   - حالت خطر :در صورتی که گاز CO از حد مجاز بیشتر شد فن ها بصورت مکنده عمل میکنند و سیستم لاگ خطر ثبت میکند.(شبیه سازی با پتانسیومتر)
   - باز و بستن درب ورودی منزل
4. قابلیت گزارش دهی : تمامی اکشن های روزانه گزارش گیری میشوند و در پنل وب و اپلیکیشن به همراه زبان و سطح اهمیت (سه سطح WARNING,INFO,CRITICAL) قابل دیدن هستند.اهمیت : فرض کنید خانواده به سفر رفته و فرزند دلبندشان در خانه تنها است. به راحتی میتوانند متوجه زمان باز و بسته شدن درب ها، حالت پارتی مود .... بشوند و از موارد خطرناک و دیگر موارد باخبر شوند.
5. قابلیت قفل بچه :اگر فرزند دلبند خانواده سنی کم داشت و قوای دماغی کافی جهت استفاده از ریموت کنترلر را نداشت با فعال کردن این قابلیت دیگر ریموت کنترلر کار نمیکند و ناخواسته بچه باعث عوض شدن حالات سیستم نمیشود.

6. **وب اپلیکیشن و اپلیکیشن : پنل اپلیکیشن و وب اپلیکشن مانند ریموت کنترل دارای قابلیت کنترل تمام موارد گفته شده، تنظیم ساعت و زمان، ریبوت سیستم، مشاهده لاگ ها و... میباشد.**
7. **لوگوی اختصاصی**
8. **امنیت : تمام سرور با یوزرنیم و پسورد امن محافظت میشود و دسترسی غیر مجاز ممکن نخواهد بود.**



شکل 1 : شبیه سازی مدار در WOKWI

سخت افزار :

پردازنده:

- ESP32

سنسورها:

- سنسور دما و رطوبت DHT22
- سنسور نور LDR مقاومت نوری
- پتانسیومتر شبیه سازی سنسور گاز CO
- گیرنده مادون قرمز (IR Receiver)
- دکمه شبیه سازی تشخیص گریه بچه

عملگرها و ماژول‌ها:

- نمایشگر OLED SSD1306
- سروو موتور
- ماژول ساعت RTC DS1307
- LED RGB
- بازر (Buzzer)

توابع مهم:

توابع اصلی راه‌اندازی و حلقه(Setup & Loop)

- **setup : این تابع قلب تپنده اولیه سیستم است. در این بخش، تمام تنظیمات اولیه انجام می‌شود:**
  - **راه‌اندازی ارتباط سریال.(Serial.begin)**
  - **اتصال به شبکه وای‌فای.(WiFi.begin)**
  - **تنظیم پین‌های ورودی و خروجی.**

- راه‌اندازی سنسورها، نمایشگر(display.begin) ، ساعت (rtc.begin) و گیرنده مادون قرمز.(IrReceiver.begin)
- مهم‌تر از همه :ایجاد و مدیریت تمام تسک‌های سیستم عامل FreeRTOS که هر کدام وظیفه مشخصی را به صورت موازی اجرا می‌کنند.
- تعریف مسیرهای وب سرور (server.on) برای پاسخ به درخواست‌های.HTTP

- Loop : این تابع در پروژه‌های مبتنی بر FreeRTOS معمولاً خالی است. پس از اجرای setup(), کنترل برنامه به تسک‌های ایجاد شده واگذار می‌شود.این تابع تسک خودش را حذف میکند.

تسک‌های پس‌زمینه(FreeRTOS Tasks)

این توابع به صورت موازی اجرا می‌شوند تا وظایف مختلف سیستم را مدیریت کنند.

- displayUpdateTask تنها تسکی که وظیفه به‌روزرسانی نمایشگر OLED را بر عهده دارد . این تسک وضعیت‌های مختلف سیستم مانند حالت خودکار، دستی، پارتی مود، خطر و پیام‌های موقت را نمایش می‌دهد.
- mainControlTask مغز کنترلی سیستم در حالت دستی .این تسک دستورات دریافت شده از ریموت کنترل را از یک صف می‌خواند و بر اساس آن، وضعیت سیستم (روشن/خاموش کردن چراغ‌ها، تغییر حالت‌ها، قفل کودک و...) را تغییر می‌دهد .
- DHTTask به طور مداوم دما و رطوبت را از سنسور DHT22 می‌خواند .در حالت خودکار، اگر دما از حد معینی (24 درجه) بالاتر رود، فن‌ها را فعال می‌کند .
- ldrTask میزان نور محیط را با سنسور LDR اندازه‌گیری می‌کند .در حالت خودکار، روشنایی لامپ‌ها را بر اساس شدت نور محیط تنظیم می‌کند (در تاریکی روشن‌تر و در روشنایی کمتر) .
- partyTask مسئول اجرای حالت "پارتی مود ."وقتی فعال است، به صورت مداوم رنگ‌های تصادفی برای تمام لامپ‌های RGB تولید کرده و یک افکت رقص نور ایجاد می‌کند .

- webServerTask وظیفه این تسک، رسیدگی به درخواست‌های ورودی به وب سرور است تا سیستم پاسخگو باقی بماند .
- Fan1Task(), Fan2Task(), Fan3Task هر کدام از این تسک‌ها یک سروو موتور (فن) را کنترل می‌کنند .
- coTask سطح گاز CO را از سنسور مربوطه می‌خواند .اگر مقدار آن از یک آستانه مشخص بیشتر شود، متغیر سراسری danger را true می‌کند تا سیستم به حالت خطر برود .در این حالت فن ها به حالت مکنده در می آیند تا هوا را سریعا تهویه کنند.
-
- buzzerTask اگر گریه کودک تشخیص داده شود، این تسک یک ملودی هشدار را با استفاده از بازر پخش می‌کند .
- logProcessingTask لاگ‌های ثبت شده در سیستم را از یک صف دریافت کرده و در یک بافر چرخشی ذخیره می‌کند تا بعداً از طریق وب سرویس قابل دسترسی باشند .
- IRTask به سیگنال‌های گیرنده مادون قرمز گوش می‌دهد و پس از دریافت و کدگشایی، کد دستور را برای پردازش به تسک mainControlTaskارسال می‌کند .
- clockTask زمان را از ماژول RTC می‌خواند و تشخیص می‌دهد که آیا شب است یا نه .در حالت خودکار، با شروع شب، کنترل نور را از سنسور LDR گرفته و یک نور ملایم برای خواب تنظیم می‌کند .
- setColor رنگ یک لامپ RGB را با استفاده از سه مقدار قرمز، سبز و آبی تنظیم می‌کند .
- onOrOff وضعیت کلی سیستم (on) را تغییر می‌دهد و تسک‌های مربوطه را بر اساس روشن یا خاموش بودن، فعال یا غیرفعال می‌کند .
- openDoor / closeDoor سروو موتور مربوط به در را برای باز یا بسته کردن آن حرکت می‌دهند .
- showMessage یک پیام موقت را برای نمایش روی صفحه OLED تنظیم می‌کند .

- addLog یک رویداد یا خطا را با سطح اهمیت مشخص مانند INFO, ERROR به صف لاگ‌ها اضافه می‌کند تا ثبت شود .
- onChildCryInterrupt این یک تابع وقفه‌ای (Interrupt) است .به محض تشخیص سیگنال از پین سنسور صدای گریه، اجرا شده و اگر حالت محافظت از کودک فعال بود به والدین اطلاع داده میشود.
- childProtect : حالت حفاظت از گریه بچه فعال میشود
- moodToString وضعیت یک لامپ MOOD_ON, MOOD_OFF, MOOD_SLEEP را به یک رشته متنی برای نمایش یا ارسال در JSON تبدیل می‌کند .

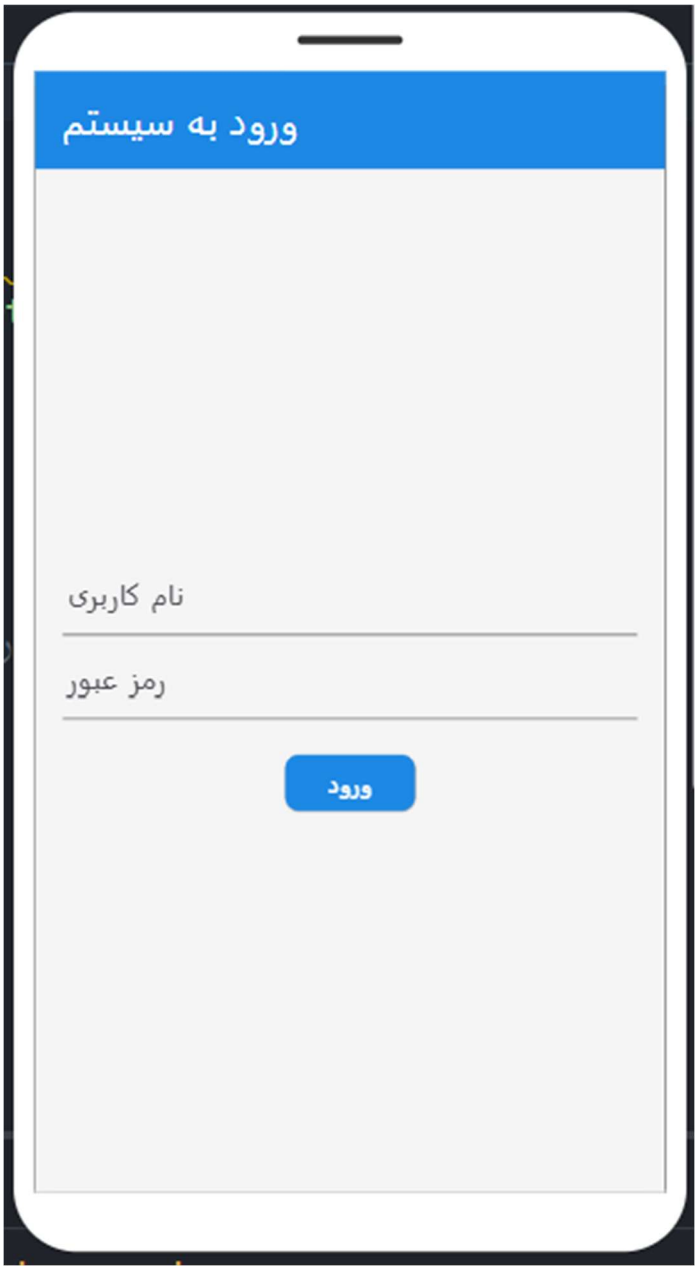
نحوه استفاده :

Figure 2 : ریموت کنترل

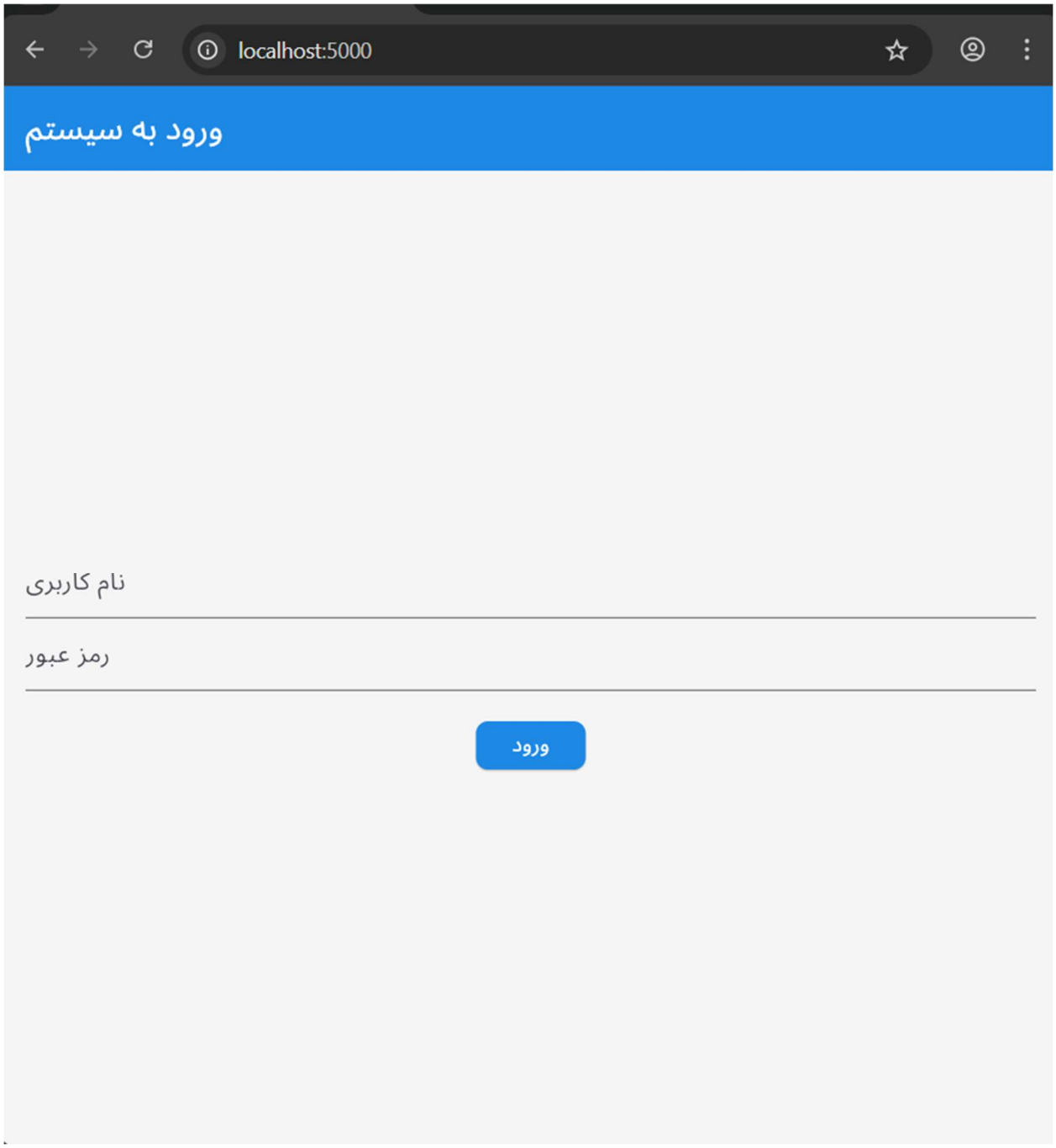
با زدن دکمه پاور لوگو اولیه مشاهده خواهد شد و ESP روشن می‌شود. عملکرد سایر دکمه ها به شرح زیر است:

- دکمه Menu : فعال یا غیر فعال‌سازی حالت خودکار
- دکمه Previuos : باز و بستن درب
- دکمه Play : فعال‌سازی یا غیر فعال‌سازی حالت حفاظت از بچه(در حالت خودکار فعال است)
- دکمه 1: روشن یا خاموش کردن حالت پارتی

- دکمه صفر: خاموشی تمام LED ها
- دکمه 2 : روشن و یا خاموش نمودن لامپ پذیرایی
- دکمه 3: روشن و یا خاموش نمودن حالت شب در پذیرایی
- دکمه 4: روشن و یا خاموش نمودن لامپ اتاق 1
- دکمه 5: روشن و یا خاموش نمودن حالت شب در اتاق 1
- دکمه 6: روشن یا خاموش نمودن لامپ اتاق 2
- دکمه شماره 7 : فعال سازی قفل بچه
- دکمه 8: روشن یا خاموش نمودن حالت شب در اتاق 2
- دکمه شماره 9: روشن یا خاموش نمودن کولر اتاق 1
- دکمه Plus: روشن یا خاموش نمودن کولر پذیرایی
- دکمه minus : روشن و یا خاموش نمودن کولر اتاق 2



شکل3 : نرم افزار اندروید

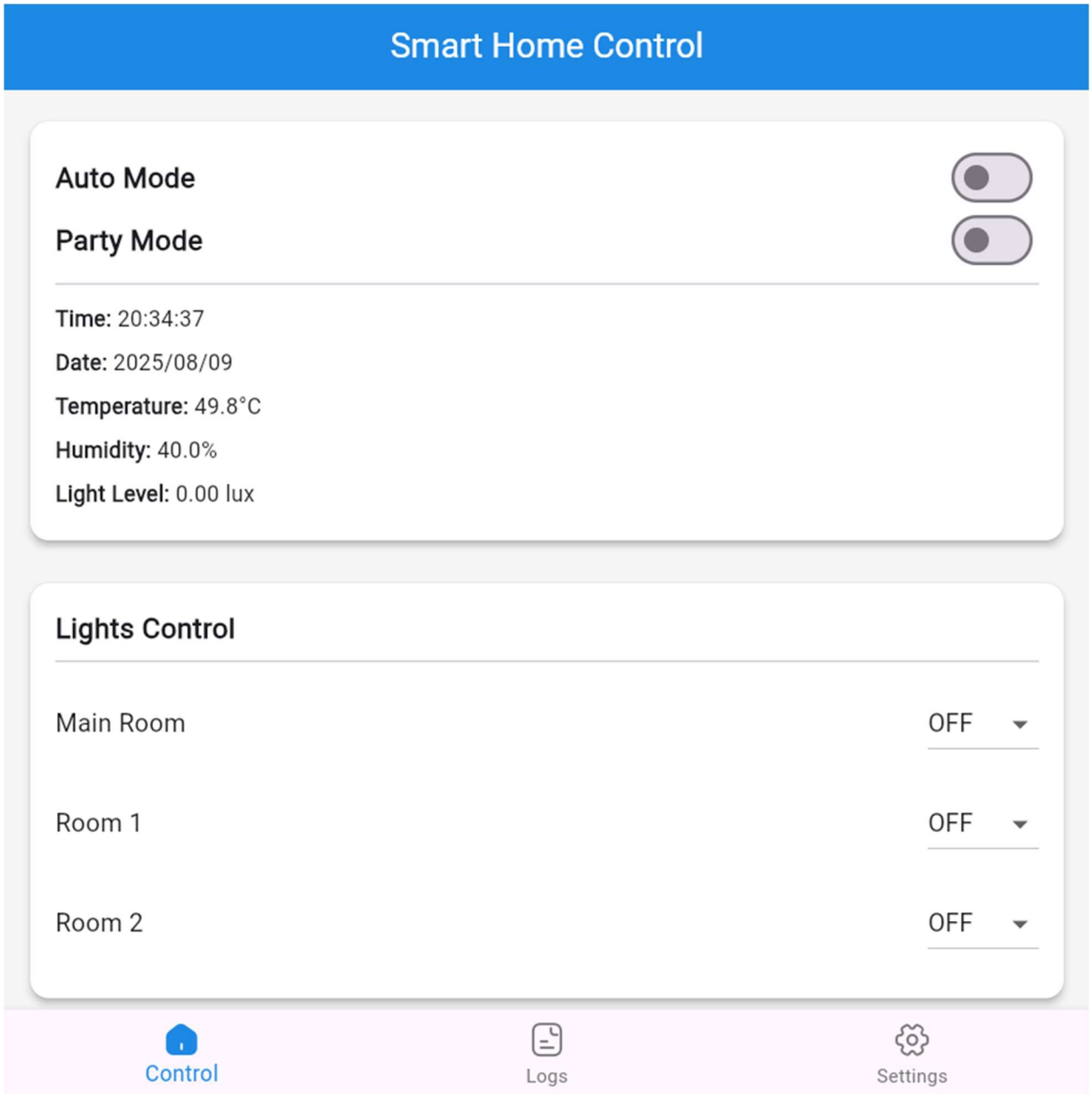


شکل چهارم : وب اپلیکیشن

نام کاربری و رمز عبور پیشفرض :

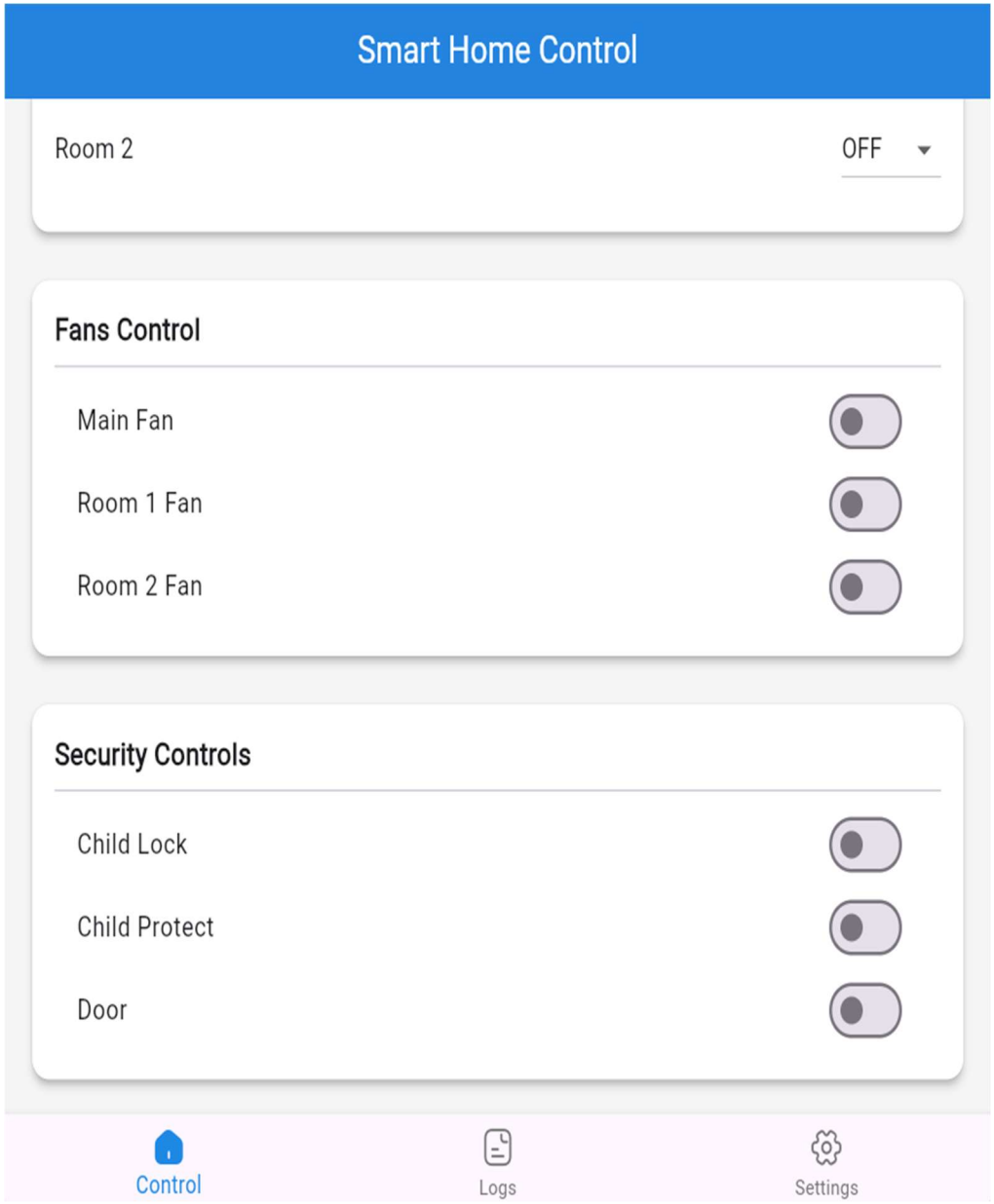
admin

1234

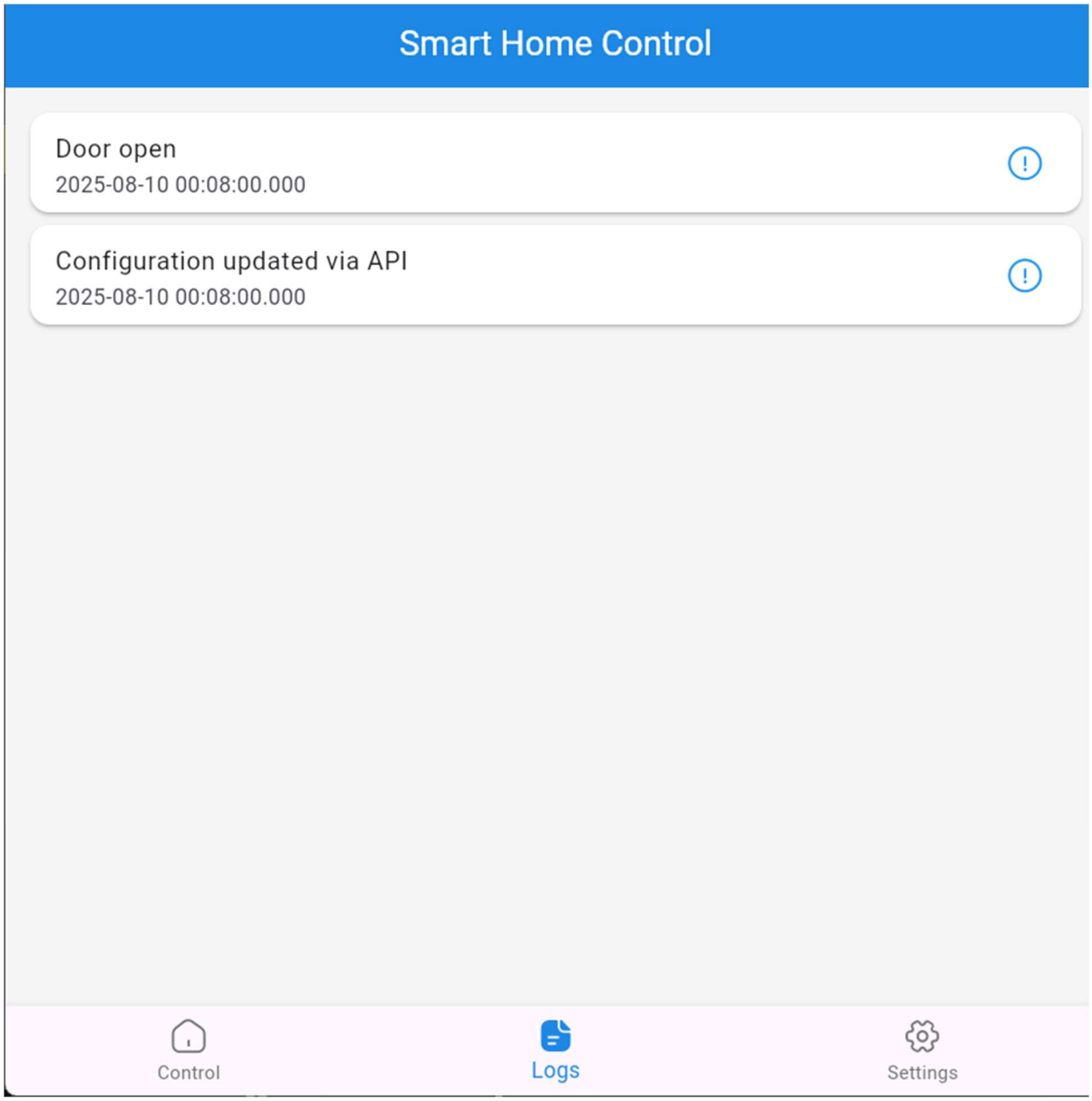


شکل پنجم: منو اولیه

در بخش Control میتوانید گزینه هارا مشاهده و تغییر دهید.

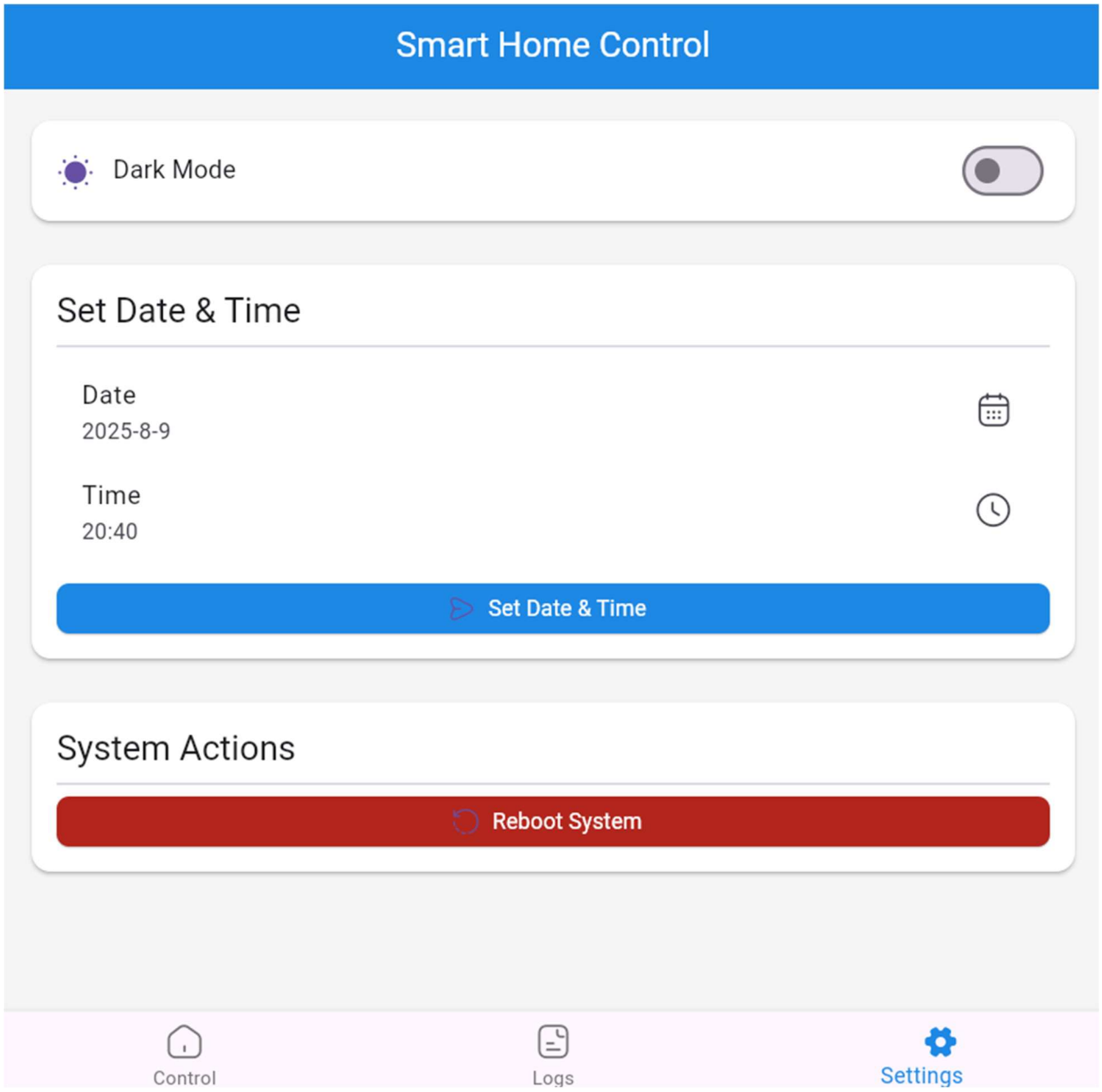


شکل ششم منو اولیه



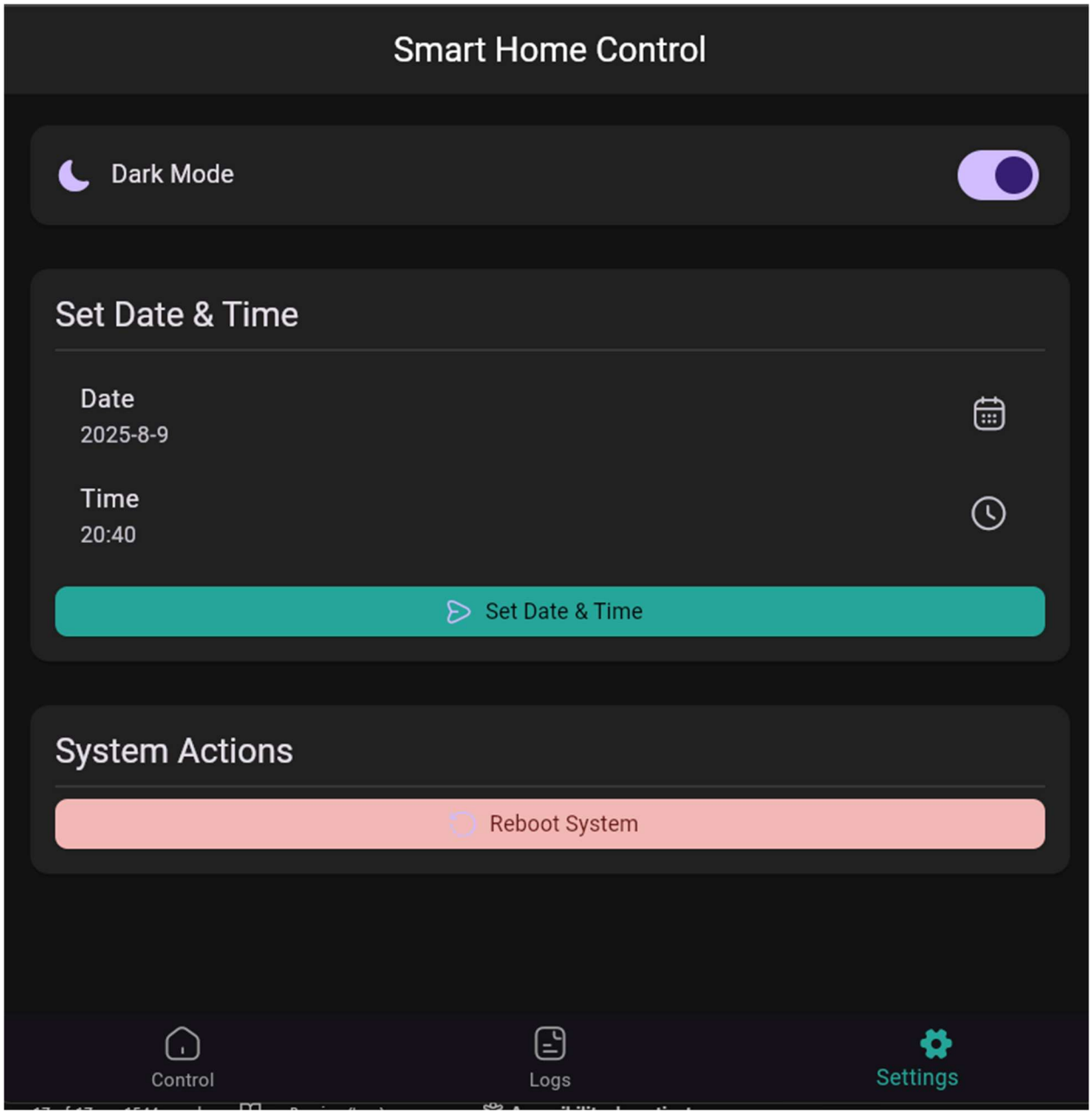
مشاهده گزارشات با زمان در بخش گزارشات

در بخش گزارشات به همراه زمان مربوطه میتوانید گزارش را دیده و بررسی کنید.

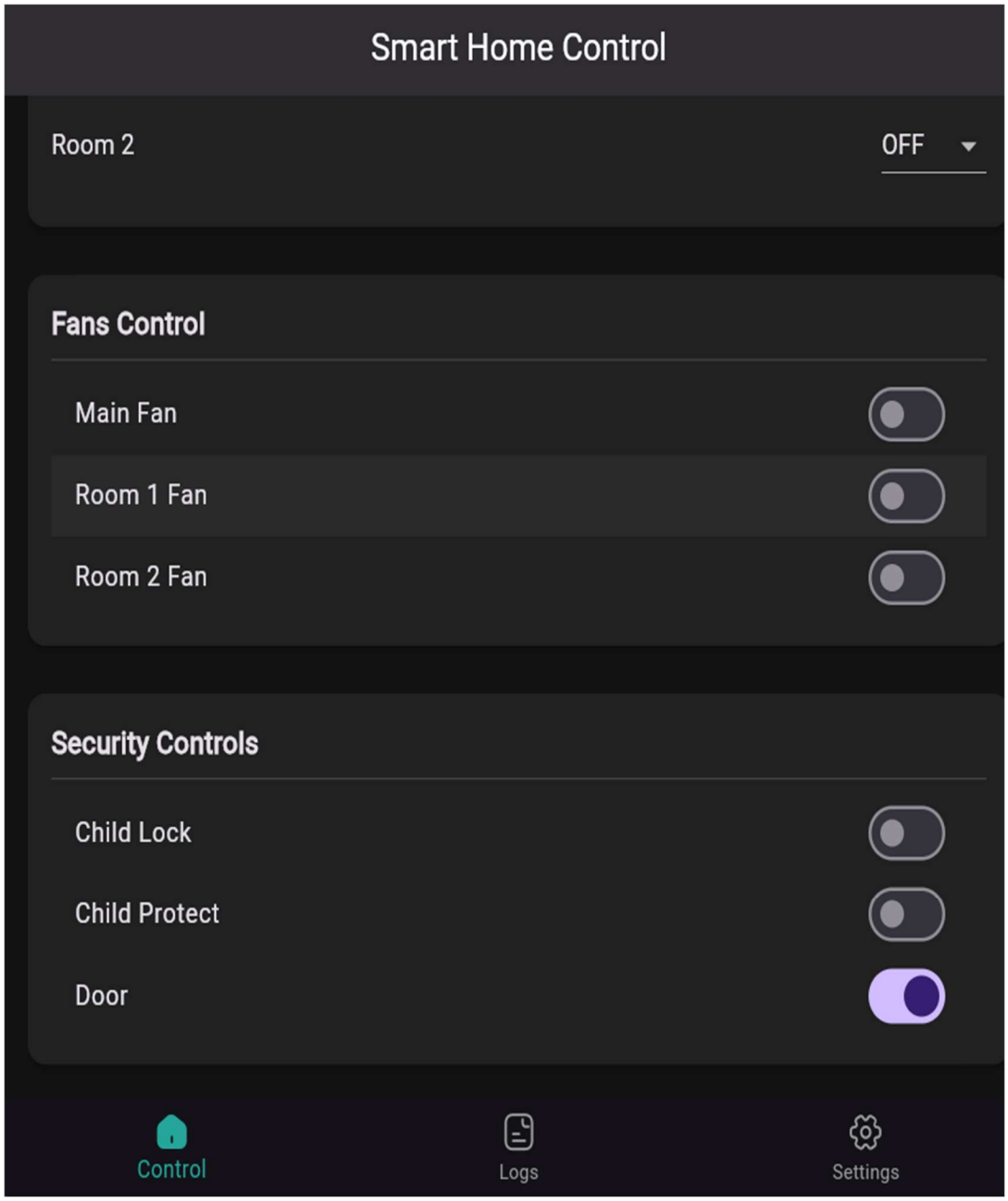


شکل هشتم بخش تنظیمات

در بخش تنظیمات میتوانید سیستم را از راه دور راه اندازی مجدد کنید و یا زمان را تنظیم کنید . همچنین میتوانید دارک مود را فعال کنید .



شکل نهم : نمایی از دارک مود



شکل دهم نمایی از حالت دارک مود

کد نهایی:

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <IRremote.h>
#include <DHT.h>
#include <ESP32Servo.h>
#include <ArduinoJson.h>
#include "RTClib.h"
#include <WebServer.h>
#include <WiFi.h>

RTC_DS1307 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday","Friday"};
#define room_1_led_blue_pin 32
#define room_2_led_blue_pin 33
#define room_1_led_green_pin 25
#define room_2_led_green_pin 12
#define room_1_led_red_pin 14
#define room_2_led_red_pin 15

#define co2_pin 35
#define door_pin 16
#define child_cry_pin 23
#define buzzer_pin 5
// --- تعریف پین‌ها ---
#define LDR_PIN 36
#define IR_RECEIVE_PIN 2
#define LED_RED_PIN 19
#define LED_GREEN_PIN 17
#define LED_BLUE_PIN 18
#define DHT22_PIN 4
#define SERVO_PIN 13

// --- تنظیمات نمایشگر ---
#define SCREEN_I2C_ADDR 0x3C
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RST_PIN -1

// --- تنظیمات انیمیشن ---
```

```
#define FRAME_DELAY_MS 42
#define FRAME_WIDTH 64
#define FRAME_HEIGHT 64
volatile bool childCry=false;
volatile bool night=false;

#define ROOM_1_FAN 26
#define ROOM_2_FAN 27
// --- تنظیمات وای‌فای و وب سرور ---
const char* ssid = "Wokwi-GUEST"; // نام وای‌فای برای شبیه‌ساز Wokwi
const char* password = "";        // رمز عبور برای شبیه‌ساز Wokwi

// ================== DISPLAY MANAGEMENT VARIABLES ==================
char temporaryMessageLine1[32] = "";
char temporaryMessageLine2[32] = "";
unsigned long messageDisplayUntil = 0;
// --- جدید: تعریف اطلاعات احراز هویت ---
const char* www_username = "admin";
const char* www_password = "1234";
WebServer server(80);             // ساخت یک نمونه وب سرور روی پورت 80
volatile bool partyMode=false;
// --- متغیرهای سراسری ---
volatile bool on = false;
volatile bool autoMode = false;
enum MOOD {
  MOOD_ON,
  MOOD_OFF,
  MOOD_SLEEP
};
// 1. تعریف سطح اهمیت برای لاگ‌ها
enum LogLevel {
  INFO,
  WARNING,
  ERROR
};

// 2. ساختار هر رکورد لاگ
struct LogEntry {
  uint32_t timestamp; // زمان وقوع بر حسب میلی‌ثانیه از بوت
  LogLevel level;
  char message[64]; // پیام لاگ
};

// 3. تعریف بافر چرخشی برای ذخیره لاگ‌ها
```

```
#define LOG_BUFFER_SIZE 100
LogEntry logBuffer[LOG_BUFFER_SIZE];
int logBufferIndex = 0;
volatile int logCount = 0; // تعداد لاگ‌های ثبت شده

// 4. تعریف یک صف برای ارتباط بین تسک‌ها
QueueHandle_t logQueue;
volatile MOOD ROOM1MOOD = MOOD_OFF;
volatile MOOD ROOM2MOOD = MOOD_OFF;
volatile MOOD ROOMMOOD  = MOOD_OFF;

volatile bool danger=false;
volatile bool fan1Active = false;
volatile bool fan2Active = false;
volatile bool fan3Active = false;
volatile bool room1LightOn = false;
volatile bool room2LightOn = false;
volatile bool mainRoomLightOn = false;
volatile bool doorOpen=false;
volatile bool childLock=false;
QueueHandle_t irQueue;

TaskHandle_t co2Task=NULL;
TaskHandle_t cooler1Task = NULL;
TaskHandle_t cooler2Task = NULL;
TaskHandle_t cooler3Task = NULL;

TaskHandle_t dhtHandle=NULL;
TaskHandle_t partyTaskHandle = NULL;
TaskHandle_t clockTaskHandle=NULL;
TaskHandle_t ldrTaskHandle = NULL;
TaskHandle_t webServerTaskHandle = NULL;
volatile bool childProtect=false;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST_PIN);

// --- تعریف سنسور و متغیرهای دما ---
DHT dht(DHT22_PIN, DHT22);
volatile float current_temperature = -99.0;
volatile float current_humidity = -99.0;
volatile float current_lux = 0.0;
  char uptimeString[9];
   char updateString[11];
```

```
// --- تعریف سروو و متغیرهای کنترلی فن ---
Servo fan,door,fan2,fan3;


// --- تعریف Mux برای بخش بحرانی ---
portMUX_TYPE dhtMux = portMUX_INITIALIZER_UNLOCKED;

// --- داده‌های انیمیشن (کوتاه شده) ---
const byte PROGMEM frames2[][512] = {
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,128,0,0,0,0,0,0,127,254,0,0,0,0,0,3,255,
255,192,0,0,0,0,31,241,255,248,0,0,0,0,63,1,240,252,0,0,0,0,248,1,225,255,0,0,0,3
,224,1,195,199,192,0,0,7,128,1,199,135,224,0,0,15,0,1,143,15,240,0,0,30,0,1,158,3
0,120,0,0,60,0,1,188,60,60,0,0,120,0,1,248,120,126,0,0,240,0,1,240,240,255,0,0,22
4,0,1,225,225,231,0,1,192,0,1,195,195,199,128,1,128,0,1,135,135,135,128,3,128,0,1
,143,15,1,192,7,0,0,1,158,30,96,224,7,0,0,3,252,60,96,224,6,0,0,31,248,120,64,96,
14,0,0,127,254,240,0,112,14,0,0,248,31,230,6,112,12,0,1,224,7,198,6,48,28,0,3,192
,3,192,4,56,28,0,3,128,1,192,0,56,28,0,7,0,0,224,96,120,24,0,7,0,0,224,96,248,24,
0,6,0,0,96,0,24,24,0,6,0,0,96,0,24,24,0,6,0,0,102,6,24,24,0,6,0,0,102,6,24,24,0,6
,0,0,96,0,24,24,0,6,0,0,96,0,24,24,0,7,0,0,224,96,248,28,0,7,0,0,224,96,120,28,0,
3,128,1,192,0,56,28,0,3,192,3,192,4,56,12,0,1,224,7,134,6,48,14,0,0,248,31,198,6,
112,14,0,0,127,254,224,0,112,6,0,0,31,248,112,64,96,7,0,0,0,0,56,96,224,7,0,0,0,0
,28,96,224,3,128,0,0,0,14,1,192,1,128,0,0,0,7,7,128,1,192,0,0,0,3,135,128,0,224,0
,0,0,1,199,0,0,240,0,0,0,0,239,0,0,120,0,0,0,0,126,0,0,60,0,0,0,0,60,0,0,30,0,0,0
,0,120,0,0,15,0,0,0,0,240,0,0,7,128,0,0,1,224,0,0,3,224,0,0,7,192,0,0,0,248,0,0,3
1,0,0,0,0,63,0,0,252,0,0,0,0,31,240,15,248,0,0,0,0,3,255,255,192,0,0,0,0,0,127,25
4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
};
// ==========================================================
// تابع جدید: تبدیل وضعیت Mood به رشته
// ==========================================================
const char* moodToString(MOOD mood) {
  switch (mood) {
    case MOOD_ON: return "ON";
    case MOOD_OFF: return "OFF";
    case MOOD_SLEEP: return "SLEEP";
    default: return "UNKNOWN";
  }
}
void addLog(LogLevel level, const char* message) {
  LogEntry newLog;
    DateTime now = rtc.now();

  newLog.timestamp = now.unixtime();
  newLog.level = level;
```

```
  strncpy(newLog.message, message, sizeof(newLog.message) - 1);
  newLog.message[sizeof(newLog.message) - 1] = '\0'; // Ensure null termination

  // ارسال لاگ به صف برای پردازش امن
  // از xQueueSend استفاده می‌کنیم تا اگر صف پر بود، تسک ارسال‌کننده قفل نشود
  xQueueSend(logQueue, &newLog, 0);
}

// This is the ONLY task that should ever call display.display()
void displayUpdateTask(void *pvParameters) {
    while (1) {
      if(!on){
                vTaskDelay(pdMS_TO_TICKS(1000)); // Update the display ~4 times
per second
continue;
      }
        display.clearDisplay();
        display.setTextColor(SSD1306_WHITE);

        // Check if a temporary message should be shown
        if (millis() < messageDisplayUntil) {
            display.setTextSize(2);
            display.setCursor(0, 10);
            display.print(temporaryMessageLine1);
            display.setCursor(0, 35);
            display.print(temporaryMessageLine2);
        } else {
            // If no temporary message, show the main status screen
         if (danger) {
                display.setTextSize(2);
                display.setCursor(5, 10);
                display.print("DANGER!");
                display.setTextSize(1);
                display.setCursor(5, 35);
                display.print("High CO Levels!");
            }
            else if (partyMode) {
                display.setTextSize(2);
                display.setCursor(0, 28);
                display.print("PARTY MODE");
            } else if (autoMode) {
                display.setTextSize(2);
                display.setCursor(5, 0);
                display.print("AUTO MODE");
```

```
                display.setTextSize(1);
                display.setCursor(0, 24);
                display.print("Temp: " + String(current_temperature, 1) + "C");
                display.setCursor(70, 24);
                display.print("Lux: " + String(current_lux, 0));
                display.setCursor(0, 40);
                DateTime now = rtc.now();
                char timeString[20];
                sprintf(timeString, "%02d:%02d:%02d %s", now.hour(),
now.minute(), now.second(), night ? "(N)":"");
                display.print(timeString);
            } else { // Manual Mode
                display.setTextSize(2);
                display.setCursor(0, 0);
                display.print("MANUAL");
                display.setTextSize(1);
                display.setCursor(0, 20);
                display.print("Main:  " + String(moodToString(ROOMMOOD)));
                display.setCursor(0, 32);
                display.print("Room1: " + String(moodToString(ROOM1MOOD)));
                display.setCursor(0, 44);
                display.print("Room2: " + String(moodToString(ROOM2MOOD)));
            }
        }

        display.display();
        vTaskDelay(pdMS_TO_TICKS(250)); // Update the display ~4 times per second
    }
}
// ==========================================================

void logProcessingTask(void *pvParameters) {
  LogEntry receivedLog;
  while (1) {
    // منتظر دریافت لاگ از صف بمان
    if (xQueueReceive(logQueue, &receivedLog, portMAX_DELAY) == pdPASS) {
      // لاگ جدید را به بافر چرخشی اضافه کن
      logBuffer[logBufferIndex] = receivedLog;
      logBufferIndex = (logBufferIndex + 1) % LOG_BUFFER_SIZE; // این باعث چرخشی شدن
بافر می‌شود

      if (logCount < LOG_BUFFER_SIZE) {
        logCount++;
      }
```

```
    }
  }
}
// --- توابع کمکی ---
void setColor(int redPin,int greenPin,int bluePin,int r, int g, int b) {

    analogWrite(redPin, r);
    analogWrite(greenPin, g);
    analogWrite(bluePin, b);
}

// ============================================================
// تابع بازگردانده شده
// ============================================================
void testServo(Servo fan,int pin) {

    fan.attach(pin); // اتصال سروو به پین مشخص شده
    for(int pos = 0; pos <= 180; pos += 1) {
        fan.write(pos);
    }
    for(int pos = 180; pos >= 0; pos -= 1) {
        fan.write(pos);
    }
}
void testServoReverse(Servo fan,int pin) {

    fan.attach(pin); // اتصال سروو به پین مشخص شده
      for(int pos = 180; pos >= 0; pos -= 1) {
        fan.write(pos);
    }
    for(int pos = 0; pos <= 180; pos += 1) {
        fan.write(pos);
    }

}
void handleLogsRequest() {
      // ---- بخش جدید برای احراز هویت ----
  if (!server.authenticate(www_username, www_password)) {
    Serial.println("LOGg");
    // اگر اطلاعات اشتباه بود یا وجود نداشت، درخواست احراز هویت ارسال کن
    return server.requestAuthentication();
  }
  Serial.println("Log");
  JsonDocument doc; // برای سادگی از داکیومنت داینامیک استفاده می‌کنیم
```

```
  JsonArray logArray = doc.to<JsonArray>();

  // این حلقه بافر چرخشی را به ترتیب صحیح (از قدیمی‌ترین به جدیدترین) می‌خواند
  int start_index = 0;
  if (logCount == LOG_BUFFER_SIZE) {
      start_index = logBufferIndex;
  }

  for (int i = 0; i < logCount; i++) {
    int current_index = (start_index + i) % LOG_BUFFER_SIZE;

    JsonObject logEntry = logArray.add<JsonObject>();
    logEntry["timestamp"] = logBuffer[current_index].timestamp;

    // تبدیل سطح لاگ به رشته
    switch (logBuffer[current_index].level) {
      case INFO:    logEntry["level"] = "INFO"; break;
      case WARNING: logEntry["level"] = "WARNING"; break;
      case ERROR:   logEntry["level"] = "ERROR"; break;
    }

    logEntry["message"] = logBuffer[current_index].message;
  }

  // ارسال پاسخ JSON به کلاینت
  server.setContentLength(measureJson(doc));
 WiFiClient client = server.client();

  String jsonOutput;
  serializeJson(doc, jsonOutput);
  server.send(200, "application/json", jsonOutput);
}
void onOrOff(){
     on = !on;
                if (on) {
                  vTaskResume(co2Task);
                  danger=false;
                    vTaskResume(webServerTaskHandle);
                                    partyMode=false;

                    vTaskSuspend(partyTaskHandle);
                    TaskHandle_t animHandle;
                    xTaskCreate(powerOnAnimationTask, "Anim", 4096, NULL, 2,
&animHandle);
```

```
                } else {
      vTaskSuspend(co2Task);
                  danger=false;
                                        partyMode=false;
                    vTaskSuspend(webServerTaskHandle);

                    vTaskSuspend(partyTaskHandle);
                    if (autoMode) {
                        autoMode = false;
                        vTaskSuspend(ldrTaskHandle);
                       vTaskSuspend(clockTaskHandle);

                    }
                    childProtect=false;
                    childLock=false;
                    fan1Active = false;
                                        fan2Active = false;
                    fan3Active = false;
vTaskSuspend(cooler1Task);
vTaskSuspend(cooler2Task);
vTaskSuspend(cooler3Task);

                          setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,0, 0,
0);

                }
}
void handleRebootRequest() {
   // ---- احراز هویت ----
  if (!server.authenticate(www_username, www_password)) {
    return server.requestAuthentication();
  }
 server.send(200,  "application/json", "{\"status\":\"Rebooting\"}");
  ESP.restart(); // ریستارت کردن ESP32
}


            void handleOffRequest() {
               // ---- احراز هویت ----
  if (!server.authenticate(www_username, www_password)) {
    return server.requestAuthentication();
  }
                           onOrOff();
```

```
            }
void handleSetTimeRequest() {
  // ---- احراز هویت ----
  if (!server.authenticate(www_username, www_password)) {
    return server.requestAuthentication();
  }

  if (server.method() != HTTP_POST) {

    server.send(405, "text/plain", "Method Not Allowed");
    return;
  }

  String postBody = server.arg("plain");
  JsonDocument doc;
  DeserializationError error = deserializeJson(doc, postBody);

  if (error) {
    server.send(400, "text/plain", "Invalid JSON");
    return;
  }

  if (!doc.containsKey("year") || !doc.containsKey("month")
|| !doc.containsKey("day") ||
      !doc.containsKey("hour") || !doc.containsKey("minute")
|| !doc.containsKey("second")) {
    server.send(400, "text/plain", "Missing required fields");
    return;
  }

  int year = doc["year"];
  int month = doc["month"];
  int day = doc["day"];
  int hour = doc["hour"];
  int minute = doc["minute"];
  int second = doc["second"];

  if (year < 2023 || year > 2100 ||
      month < 1 || month > 12 ||
      day < 1 || day > 31 ||
      hour < 0 || hour > 23 ||
      minute < 0 || minute > 59 ||
      second < 0 || second > 59) {
    server.send(400, "text/plain", "Invalid date/time values");
```

```
    return;
  }

  DateTime newTime(year, month, day, hour, minute, second);
  rtc.adjust(newTime);

  DateTime now = rtc.now();
  sprintf(uptimeString, "%02d:%02d:%02d", now.hour(), now.minute(),
now.second());
  sprintf(updateString, "%d/%02d/%02d", now.year(), now.month(), now.day());

  // ساخت پاسخ JSON به صورت صحیح
  JsonDocument responseDoc;
  responseDoc["status"] = "success";
  responseDoc["message"] = "RTC time updated successfully";

  // ایجاد آبجکت تودرتو برای new_time
  JsonObject newTimeObj = responseDoc.createNestedObject("new_time");
  newTimeObj["year"] = now.year();
  newTimeObj["month"] = now.month();
  newTimeObj["day"] = now.day();
  newTimeObj["hour"] = now.hour();
  newTimeObj["minute"] = now.minute();
  newTimeObj["second"] = now.second();
  newTimeObj["day_of_week"] = daysOfTheWeek[now.dayOfTheWeek()];

  String response;
  serializeJson(responseDoc, response);
  server.send(200, "application/json", response);

  char logMessage[64];
  snprintf(logMessage, sizeof(logMessage), "RTC time set
to %04d-%02d-%02d %02d:%02d:%02d",
           year, month, day, hour, minute, second);
  addLog(INFO, logMessage);
}

// اضافه کنید: server.on ، بعد از سایر setup در تابع

// تابع جدید برای پردازش درخواست POST /config
void handleConfigRequest() {

  // ---- احراز هویت ----
  if (!server.authenticate(www_username, www_password)) {
```

```
  return server.requestAuthentication();
}


// پردازش بدنه JSON
String postBody = server.arg("plain");
JsonDocument doc;
DeserializationError error = deserializeJson(doc, postBody);

if (error) {
  server.send(400, "text/plain", "Invalid JSON");
  return;
}

// اعمال تغییرات
if (doc.containsKey("system_power")) {
  on = doc["system_power"].as<bool>();
  if (!on) {
    autoMode = false;
    partyMode = false;
    vTaskSuspend(partyTaskHandle);
    vTaskSuspend(ldrTaskHandle);
    vTaskSuspend(clockTaskHandle);
    fan1Active = false;
    fan2Active = false;
    fan3Active = false;
    setColor(LED_RED_PIN, LED_GREEN_PIN, LED_BLUE_PIN, 0, 0, 0);
  }
}

if (doc.containsKey("auto_mode")) {
  autoMode = doc["auto_mode"].as<bool>();
  if (autoMode) {
    partyMode = false;
    vTaskResume(clockTaskHandle);
    vTaskResume(ldrTaskHandle);
    vTaskSuspend(partyTaskHandle);
  } else {
    vTaskSuspend(ldrTaskHandle);
    vTaskSuspend(clockTaskHandle);
  }
}

if (doc.containsKey("child_lock")) {
```

```
    childLock = doc["child_lock"].as<bool>();
  }

  if (doc.containsKey("child_protect")) {
    childProtect = doc["child_protect"].as<bool>();
  }

  if (doc.containsKey("party_mode")) {
    partyMode = doc["party_mode"].as<bool>();
    if (partyMode) {
      vTaskResume(partyTaskHandle);
    } else {
                                                                    setColor(LED_RED_PIN
,LED_GREEN_PIN,LED_BLUE_PIN,0,0,0);
                                      setColor(room_1_led_red_pin,room_1_led_blue_pi
n,room_1_led_green_pin,0,0,0);

                                      setColor(room_2_led_red_pin,room_2_led_blue_pi
n,room_2_led_green_pin,0,0,0);

      vTaskSuspend(partyTaskHandle);
    }
  }

  // کنترل نورها
  if (doc.containsKey("lights") && !partyMode && !autoMode) {
    JsonObject lights = doc["lights"];

    if (lights.containsKey("main_room")) {
      String mood = lights["main_room"].as<String>();
      if (mood == "ON") {
        ROOMMOOD = MOOD_ON;
        setColor(LED_RED_PIN, LED_GREEN_PIN, LED_BLUE_PIN, 255, 255, 200);
      } else if (mood == "SLEEP") {
        ROOMMOOD = MOOD_SLEEP;
        setColor(LED_RED_PIN, LED_GREEN_PIN, LED_BLUE_PIN, 50, 10, 0);
      } else {
        ROOMMOOD = MOOD_OFF;
        setColor(LED_RED_PIN, LED_GREEN_PIN, LED_BLUE_PIN, 0, 0, 0);
      }
    }

    if (lights.containsKey("room_1") && !partyMode  && !autoMode) {
      String mood = lights["room_1"].as<String>();
```

```
      if (mood == "ON") {
        ROOM1MOOD = MOOD_ON;
        setColor(room_1_led_red_pin, room_1_led_green_pin, room_1_led_blue_pin,
255, 255, 255);
      } else if (mood == "SLEEP") {
        ROOM1MOOD = MOOD_SLEEP;
        setColor(room_1_led_red_pin, room_1_led_green_pin, room_1_led_blue_pin,
50, 0, 0);
      } else {
        ROOM1MOOD = MOOD_OFF;
        setColor(room_1_led_red_pin, room_1_led_green_pin, room_1_led_blue_pin,
0, 0, 0);
      }
    }

    if (lights.containsKey("room_2") && !partyMode  && !autoMode) {
      String mood = lights["room_2"].as<String>();
      if (mood == "ON") {
        ROOM2MOOD = MOOD_ON;
        setColor(room_2_led_red_pin, room_2_led_green_pin, room_2_led_blue_pin,
255, 255, 255);
      } else if (mood == "SLEEP") {
        ROOM2MOOD = MOOD_SLEEP;
        setColor(room_2_led_red_pin, room_2_led_green_pin, room_2_led_blue_pin,
50, 0, 0);
      } else {
        ROOM2MOOD = MOOD_OFF;
        setColor(room_2_led_red_pin, room_2_led_green_pin, room_2_led_blue_pin,
0, 0, 0);
      }
    }
  }

  // کنترل فن‌ها
  if (doc.containsKey("fans")  && !autoMode) {
    JsonObject fans = doc["fans"];

    if (fans.containsKey("main_fan")) {
      fan1Active = fans["main_fan"].as<bool>();
      if (fan1Active) {
        vTaskResume(cooler1Task);
      } else {
        vTaskSuspend(cooler1Task);
      }
```

```
  }

  if (fans.containsKey("room_1_fan")  && !autoMode) {
    fan2Active = fans["room_1_fan"].as<bool>();
    if (fan2Active) {
      vTaskResume(cooler2Task);
    } else {
      vTaskSuspend(cooler2Task);
    }
  }

  if (fans.containsKey("room_2_fan")  && !autoMode) {
    fan3Active = fans["room_2_fan"].as<bool>();
    if (fan3Active) {
      vTaskResume(cooler3Task);
    } else {
      vTaskSuspend(cooler3Task);
    }
  }
}

// کنترل درب
if (doc.containsKey("door_is_open")) {
  bool doorState = doc["door_is_open"].as<bool>();
  if (doorState) {
    openDoor();
  } else {
    closeDoor();
  }
  doorOpen = doorState;
}

// پاسخ موفقیت‌آمیز
JsonDocument responseDoc;
responseDoc["status"] = "success";
responseDoc["message"] = "Configuration updated successfully";

String response;
serializeJson(responseDoc, response);

server.send(200, "application/json", response);

// ثبت لاگ
addLog(INFO, "Configuration updated via API");
```

```
}

// اضافه کنید server.on ، بعد از setup در تابع:
void handleDataRequest() {
    Serial.println("Handling data request...");
      // ---- بخش جدید برای احراز هویت ----
  if (!server.authenticate(www_username, www_password)) {
    // اگر اطلاعات اشتباه بود یا وجود نداشت، درخواست احراز هویت ارسال کن
    return server.requestAuthentication();
  }
  // این وارنینگ StaticJsonDocument فقط یک اطلاع‌رسانی است و مشکلی ایجاد نمی‌کند.
  StaticJsonDocument<1024> doc;

  DateTime now = rtc.now();
  char timeString[9];
  sprintf(timeString, "%02d:%02d:%02d", now.hour(), now.minute(), now.second());
  char dateString[11];
  sprintf(dateString, "%d/%02d/%02d", now.year(), now.month(), now.day());
 doc["uptime"] = uptimeString;
  doc["update"] = updateString;
  doc["time"] = timeString;
  doc["date"] = dateString;
  doc["day_of_week"] = daysOfTheWeek[now.dayOfTheWeek()];

  // ===== بخش تغییر یافته با سینتکس جدید =====
  JsonObject sensors = doc["sensors"].to<JsonObject>();
  sensors["temperature"] = String(current_temperature, 1);
  sensors["humidity"] = String(current_humidity, 1);
  sensors["light_lux"] = String(current_lux, 2);
  sensors["co2_raw_value"] = analogRead(co2_pin);
  sensors["child_cry_detected"] = childCry;
  JsonObject status = doc["status"].to<JsonObject>();
  status["system_power"] = on;
  status["auto_mode"] = autoMode;
  status["child_lock"] = childLock;
  status["child_protect"] = childProtect;
  status["door_is_open"] = doorOpen;
  status["party_mode"] = partyMode;
  JsonObject lights = status["lights"].to<JsonObject>();
  lights["main_room"] = moodToString(ROOMMOOD);
  lights["room_1"] = moodToString(ROOM1MOOD);
  lights["room_2"] = moodToString(ROOM2MOOD);
  JsonObject fans = status["fans"].to<JsonObject>();
  fans["main_fan"] = fan1Active;
```

```
  fans["room_1_fan"] = fan2Active;
  fans["room_2_fan"] = fan3Active;
  // ==========================================
  // ارسال جریانی که قبلا اصلاح کردیم
  server.setContentLength(measureJson(doc));

      /*server.sendHeader(F("Access-Control-Max-Age"), F("600"));

  server.sendHeader("Access-Control-Allow-Origin", "*");
server.sendHeader("Access-Control-Allow-Methods", "GET,POST,OPTIONS");
server.sendHeader("Access-Control-Allow-Headers", "*");*/
  String jsonOutput;
  serializeJson(doc, jsonOutput);
  server.send(200, "application/json", jsonOutput);
}// ================== NEW & REFACTORED DISPLAY FUNCTIONS ==================

// This function NO LONGER draws to the screen. It just sets a temporary message
// that the dedicated display task will show.
void showMessage(String line1, String line2 = "", int duration_ms = 2500) {
    strncpy(temporaryMessageLine1, line1.c_str(), sizeof(temporaryMessageLine1) -
1);
    temporaryMessageLine1[sizeof(temporaryMessageLine1) - 1] = '\0';

    strncpy(temporaryMessageLine2, line2.c_str(), sizeof(temporaryMessageLine2) -
1);
    temporaryMessageLine2[sizeof(temporaryMessageLine2) - 1] = '\0';

    messageDisplayUntil = millis() + duration_ms;
}
// --- تسک‌ها ---
void powerOnAnimationTask(void *pvParameters) {
    int frame_count = sizeof(frames2) / sizeof(frames2[0]);
    for (int i = 0; i < frame_count; i++) {
        display.clearDisplay();
        display.drawBitmap(32, 0, frames2[i], FRAME_WIDTH, FRAME_HEIGHT,
SSD1306_WHITE);
        display.display();
        delay(FRAME_DELAY_MS);
        //vTaskDelay(pdMS_TO_TICKS(FRAME_DELAY_MS));
    }
    vTaskDelete(NULL);
}

void partyTask(void *pvParameters) {
```

```
    while (1) {
        if(!partyMode || !on) {
            vTaskSuspend(NULL); // Suspend the task if party mode is off or
system is off
        }
                    setColor(room_2_led_red_pin,room_2_led_green_pin,room_2_led_b
lue_pin,random(256), random(256), random(256));

              setColor(room_1_led_red_pin,room_1_led_green_pin,room_1_led_blue_pi
n,random(256), random(256), random(256));

        setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,random(256), random(256),
random(256));
        vTaskDelay(pdMS_TO_TICKS(150));
    }
}

void IRTask(void *pvParameters) {
    IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);
    while (true) {
        if (IrReceiver.decode()) {
            uint8_t code = IrReceiver.decodedIRData.command;
            xQueueSend(irQueue, &code, portMAX_DELAY);
            IrReceiver.resume();
        }
        vTaskDelay(pdMS_TO_TICKS(100));
    }
}


// ==========================================================
// تسک فن به حالت اولیه بازگردانده شد
// ==========================================================
void Fan1Task(void *pvParameters) {
    while (true) {
        if(on ){
                    if(danger){
                                                        testServoReverse(fan,SERV
O_PIN);
                    }

         else if(fan1Active){
```

```
                    Serial.println("Fan1 Task Running");

            testServo(fan,SERVO_PIN);

            vTaskDelay(pdMS_TO_TICKS(100));
        }
       } else {
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
    }
}

// ============================================================
// تسک فن به حالت اولیه بازگردانده شد
// ============================================================
void Fan2Task(void *pvParameters) {
    while (true) {
        if(on ){
                  if(danger){
            testServoReverse(fan2,26);
          }else if(fan2Active){

                       testServo(fan2,26);
          }

            vTaskDelay(pdMS_TO_TICKS(100));
        } else {
            vTaskDelay(pdMS_TO_TICKS(1000));
        }

}
}

void Fan3Task(void *pvParameters) {
    while (true) {
        if(on ){
               if(danger){
              testServoReverse(fan3,27);
            }else if(fan3Active){

            testServo(fan3,27);

          }
            vTaskDelay(pdMS_TO_TICKS(100));
```

```
        } else {
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
    }
}
void DHTTask(void *pvParameters) {
    dht.begin();
    vTaskDelay(pdMS_TO_TICKS(2000));
    while (true) {
        if(!on) {
           // vTaskSuspend(NULL); // Suspend the task if system is off or auto
mode is off
         continue; // Skip the rest of the loop if not in auto mode
        }
        float h = NAN, t = NAN;
        int maxRetries = 5;
        for (int i = 0; i < maxRetries; i++) {
            taskENTER_CRITICAL(&dhtMux);
            h = dht.readHumidity();
            t = dht.readTemperature();
            taskEXIT_CRITICAL(&dhtMux);
            if (!isnan(h) && !isnan(t)) {
                current_temperature = t;
                current_humidity = h;
                break;
            } else {
                vTaskDelay(pdMS_TO_TICKS(500));
            }
        }
        if (isnan(h) || isnan(t)) {
            Serial.println("FATAL: Failed to read from DHT sensor after all
retries!");
        } else {
            if (current_temperature > 24.0) {
                   Serial.println("Temperature is high, activating fans...");
                if (!fan1Active && autoMode) {
                                        Serial.println("Fan1 Activated");

                    fan1Active = true;
                }
                if(!fan2Active && autoMode) {
                    Serial.println("Fan2 Activated");
                    fan2Active = true;
                }
```

```
                if(!fan3Active && autoMode) {
                    Serial.println("Fan3 Activated");
                    fan3Active = true;
                }
                       vTaskResume(cooler1Task);
                       vTaskResume(cooler2Task);
                       vTaskResume(cooler3Task);




            } else {
                 if ((fan1Active || fan2Active || fan3Active) && autoMode) {
                                        vTaskSuspend(cooler1Task);
                                          vTaskSuspend(cooler2Task);
                                            vTaskSuspend(cooler3Task);

                  Serial.println("Fan's Deactivated");
                  fan1Active = false;
                                        fan2Active = false;
                                                            fan3Active = false;


                }
            }
            Serial.println("Temperature: " + String(current_temperature) + " C,
Humidity: " + String(current_humidity) + " %");
        }
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
const float GAMMA = 0.7;
const float RL10 = 50;
void ldrTask(void *pvParameters){
    // پارامترهای تنظیم روشنایی بر اساس لوکس
    const float MIN_LUX = 1.0;     // حداقل نور محیط (تاریک)
    const float MAX_LUX = 100.0;   // حداکثر نور محیط (روشن)
    const int MIN_BRIGHTNESS = 10; // حداقل روشنایی LED
    const int MAX_BRIGHTNESS = 255; // حداکثر روشنایی LED

    // ضرایب رنگ برای RGB (می‌توانید تنظیم کنید)
    const float RED_RATIO = 1.0f;
    const float GREEN_RATIO = 0.85f;
    const float BLUE_RATIO = 0.60f;
```

```
    while (true) {
        eTaskState partyState = eTaskGetState(partyTaskHandle);
        if (on && autoMode && (partyState == eSuspended || partyState !=
eRunning)) {
            // خواندن مقدار LDR
            int ldrValue = analogRead(LDR_PIN);

            // محاسبه مقاومت و لوکس
            float voltage = ldrValue * (5.0 / 4095.0);
            float resistance = 2000 * voltage / (5.0 - voltage);
            float lux = pow(RL10 * 1000 * pow(10, GAMMA) / resistance, (1.0 /
GAMMA));
                        current_lux = lux; // <<<<<<<<   مقدار لوکس در متغیر سراسری ذخیره
می‌شود

            Serial.print("Lux: ");
            Serial.println(lux);

            // محاسبه روشنایی بر اساس لوکس (با نگاشت لگاریتمی)
            int brightness;
            if (lux <= MIN_LUX) {
                brightness = MAX_BRIGHTNESS; // حداکثر روشنایی در تاریکی
            } else if (lux >= MAX_LUX) {
                brightness = MIN_BRIGHTNESS; // حداقل روشنایی در نور زیاد
            } else {
                // نگاشت لگاریتمی برای پاسخ طبیعی‌تر به تغییرات نور
                float logLux = log10(lux);
                float logMin = log10(MIN_LUX);
                float logMax = log10(MAX_LUX);
                brightness = map(logLux * 100, logMin * 100, logMax * 100,
MAX_BRIGHTNESS, MIN_BRIGHTNESS);
            }

            brightness = constrain(brightness, MIN_BRIGHTNESS, MAX_BRIGHTNESS);

                          setColor(room_2_led_red_pin,
                          room_2_led_green_pin,room_2_led_blue_pin,
                                  (int)(brightness * RED_RATIO),
                (int)(brightness * GREEN_RATIO),
                (int)(brightness * BLUE_RATIO));

              setColor(room_1_led_red_pin,room_1_led_green_pin,
              room_1_led_blue_pin,        (int)(brightness * RED_RATIO),
```

```
                (int)(brightness * GREEN_RATIO),
                (int)(brightness * BLUE_RATIO));

            // تنظیم رنگ با در نظر گرفتن ضرایب
                   setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,
                (int)(brightness * RED_RATIO),
                (int)(brightness * GREEN_RATIO),
                (int)(brightness * BLUE_RATIO)
            );

            vTaskDelay(pdMS_TO_TICKS(500));
        } else {
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
    }
}

void openDoor(){
    addLog(INFO, "Door open");
showMessage("Door Open");
  Serial.println("OPEN");
    door.attach(door_pin);
 door.write(180);



}
void closeDoor(){
  addLog(INFO, "Door closed");
showMessage("Door Close");
      door.attach(door_pin); // این خط بهتر است در setup باشد
 door.write(0);


}
void mainControlTask(void *pvParameters) {
    uint8_t code;

            setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,0, 0, 0);
    while (true) {
        if (xQueueReceive(irQueue, &code, portMAX_DELAY) == pdTRUE) {
          //Power button
            if (code == 162) {
             onOrOff();
```

```
                    }
            else if (on && !childLock) {
                                char ch[30];

                switch(code) {
                  case 66:
                                    childLock=!childLock;

                    snprintf(ch, sizeof(ch), "Child lock set to %s", childLock ?
"ON" : "OFF");
Serial.println(ch);
                  addLog(INFO, ch);
                  showMessage("Child Lock", childLock ? "ON" : "OFF");

                  break;
                  //playButton
                case 168 :
                                childProtect=!childProtect;

                    snprintf(ch, sizeof(ch), "Child protection set to %s",
childProtect ? "ON" : "OFF");

                  addLog(INFO, ch);
                  showMessage("Child protection", childProtect ? "ON" : "OFF");
                break;
                //Previous button
                  case 224:
                      if(doorOpen){

                        closeDoor();
                      }
                      else {
                        openDoor();
                      }
                      doorOpen=!doorOpen;
                    break;
                    //Menu button
                    case 226:
                        autoMode = !autoMode;
                        Serial.println("Auto Mode: " + String(autoMode ? "ON" :
"OFF"));

                    snprintf(ch, sizeof(ch), "Auto mode set to %s", autoMode ?
"ON" : "OFF");
```

```
                addLog(INFO, ch);
                showMessage("Auto Mode", autoMode ? "ON" : "OFF");

                     if (autoMode) {
                                                  childProtect=true;

                                  vTaskResume(clockTaskHandle);

                  Serial.println("System ON");
                          vTaskResume(ldrTaskHandle);

                                                       partyMode=false;

                          vTaskSuspend(partyTaskHandle);
                                 setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PI
N,0,0,0);
                     } else {
                       fan1Active=false;
                       fan2Active=false;
                       fan3Active=false;
                                                                 setColor(LED_RED_PIN
,LED_GREEN_PIN,LED_BLUE_PIN,0,0,0);
                                 setColor(room_1_led_red_pin,room_1_led_blue_pi
n,room_1_led_green_pin,0,0,0);

                                 setColor(room_2_led_red_pin,room_2_led_blue_pi
n,room_2_led_green_pin,0,0,0);

                                                     childProtect=false;

                          vTaskSuspend(ldrTaskHandle);
                      vTaskSuspend(clockTaskHandle);
                                  setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_P
IN,255, 255, 200);
                     }

                     break;
                // کد را در mainControlTask جایگزین کنید

                //1 button
case 48: { // دکمه پارتی مود (روشن/خاموش کردن)
    eTaskState partyState = eTaskGetState(partyTaskHandle);
```

```
if (partyState != eSuspended) {        // ---- خاموش کردن پارتی مود ----
        vTaskSuspend(partyTaskHandle);
        Serial.println("Party Mode Deactivated.");


                addLog(INFO, "Party Mode Deactivated");
                showMessage("Party mode deactivated");

        // حالا باید به حالت قبل برگردیم
        if (autoMode) {
                setColor(room_2_led_red_pin,
                        room_2_led_green_pin,room_2_led_blue_pin,
                0,0,0);

            setColor(room_1_led_red_pin,room_1_led_green_pin,room_1_led_blue_pi
n,
            0,0,0);

           // اگر حالت اتوماتیک فعال است، به کنترل سنسور برگرد
                setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,0, 0, 0); //
کنترل را به دست بگیرد  LDR  نور را موقتا خاموش کن تا تسک
           // نمایشگر را برای حالت اتوماتیک آپدیت کن

        } else {
                setColor(room_2_led_red_pin,
                        room_2_led_green_pin,room_2_led_blue_pin,
                        255,255, 255);

            setColor(room_1_led_red_pin,room_1_led_green_pin,
            room_1_led_blue_pin,255, 255, 255);

           // اگر حالت دستی فعال است، به نور معمولی برگرد
               setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,255, 255, 200);
// بازگشت به نور معمولی

        }
    } else {
        partyMode=true;
        // ---- روشن کردن پارتی مود ----
        vTaskResume(partyTaskHandle);
        Serial.println("Party Mode Activated!");

                addLog(INFO, "Party Mode Activated");
                showMessage("Party Mode", partyMode ? "ON" : "OFF");
```

```
    }

    break;
}
//4 button
case 16:
                        if (!autoMode) {
                            partyMode=false;
                            vTaskSuspend(partyTaskHandle);

       if(ROOM1MOOD==MOOD_ON){

              setColor(room_1_led_red_pin,room_1_led_green_pin,
              room_1_led_blue_pin,0,0,0);
                            ROOM1MOOD=MOOD_OFF;

       }else{

              setColor(room_1_led_red_pin,room_1_led_green_pin,
              room_1_led_blue_pin,255,255,255);
                            ROOM1MOOD=MOOD_ON;

       }


                        }
break;

//5 button
case 56:
                        if (!autoMode) {
                            partyMode=false;
                            vTaskSuspend(partyTaskHandle);

       if(ROOM1MOOD==MOOD_SLEEP){

              setColor(room_1_led_red_pin,room_1_led_green_pin,
              room_1_led_blue_pin,0,0,0);
                            ROOM1MOOD=MOOD_OFF;

       }else{
```

```
                                setColor(room_1_led_red_pin,room_1_led_green_pin,
              room_1_led_blue_pin,50,0,0);
                              ROOM1MOOD=MOOD_SLEEP;

       }
                        }
break;
// 6 button
case 90:

          if (!autoMode) {

           partyMode=false;
                              vTaskSuspend(partyTaskHandle);

       if(ROOM2MOOD==MOOD_ON){
   setColor(room_2_led_red_pin,room_2_led_green_pin,
              room_2_led_blue_pin,0,0,0);
                              ROOM2MOOD=MOOD_OFF;

       }else{
              setColor(room_2_led_red_pin,room_2_led_green_pin,
              room_2_led_blue_pin,255,255,255);
                              ROOM2MOOD=MOOD_ON;

       }

                        }
break;
// 8 button
case 74:
 if (!autoMode) {
    partyMode=false;
                              vTaskSuspend(partyTaskHandle);

       if(ROOM2MOOD==MOOD_SLEEP){
              setColor(room_2_led_red_pin,room_2_led_green_pin,
              room_2_led_blue_pin,0,0,0);
                              ROOM2MOOD=MOOD_OFF;

       }else{
                     setColor(room_2_led_red_pin,room_2_led_green_pin,
              room_2_led_blue_pin,50,0,0);
```

```
                            ROOM2MOOD=MOOD_SLEEP;

        }


                        }
break;
//0 button
case 104:
partyMode=false;
   vTaskSuspend(partyTaskHandle);
   ROOMMOOD=MOOD_OFF;
      ROOM1MOOD=MOOD_OFF;
   ROOM2MOOD=MOOD_OFF;

        Serial.println("Party Mode Deactivated.");
               setColor(room_2_led_red_pin,
                          room_2_led_green_pin,room_2_led_blue_pin,
                        0,0,0);

              setColor(room_1_led_red_pin,room_1_led_green_pin,
              room_1_led_blue_pin,0,0,0);

               setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,0, 0, 0); //  نور
قرمز برای حالت هشدار
break;

                    case 24: // 2 button
                        if (!autoMode) {
                            partyMode=false;
                            vTaskSuspend(partyTaskHandle);
                            if(ROOMMOOD==MOOD_ON){
                            setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,0, 0,
0);
                           ROOMMOOD=MOOD_OFF;

                             }else{

                                  setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN
,255, 255, 200);
                           ROOMMOOD=MOOD_ON;

                             }
```

```
                        }
                        break;
                        // minus button
                        case 152:
                           if(!autoMode ){
                            fan2Active=!fan2Active;

                        }
                        break;
                        // plus  button
                        case 2:

                        if(!autoMode ){
                            fan1Active=!fan1Active;

                        }

                        break;
                        // 9 button
                        case 82:
                               if(!autoMode ){
                            fan3Active=!fan3Active;

                        }
                        break;
                    case 122:
                    // 3 button
                        if (!autoMode) {
                            partyMode=false;
                            vTaskSuspend(partyTaskHandle);

                                        if(ROOMMOOD==MOOD_SLEEP){
                         setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,0, 0,
0);

                           ROOMMOOD=MOOD_OFF;
                             }else{
                               setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,50
, 10, 0);

                           ROOMMOOD=MOOD_SLEEP;

                             }
```

```
                        }
                        break;
                }
            }
        }
    }
}
void clockTask(void * pvParameters) {
    while (true) {
        if (on) {
            DateTime now = rtc.now();
       if(now.hour()>23 && now.hour()<6 && !night){
                night=true;
                if(autoMode){
                    vTaskSuspend(ldrTaskHandle);
                       setColor(LED_RED_PIN,LED_GREEN_PIN,LED_BLUE_PIN,50, 10,
0);

            }else{
                night=false;
            }
        }
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
}
void buzzerTask(void *pvParameters) {
  const int melody[] = {262, 294, 330, 349, 392, 440, 494, 523}; // نت‌های ساده
  const int noteDuration = 200; // میلی‌ثانیه

  while (true) {
    if (childCry) {
              char ch[30];

                addLog(ERROR, "Baby crying Please check!");
                showMessage("Baby crying", "Please check!");

      Serial.println("Cry");
      for (int i = 0; i < 8; i++) {
        tone(buzzer_pin, melody[i]);
        vTaskDelay(pdMS_TO_TICKS(noteDuration));
        noTone(buzzer_pin);
        vTaskDelay(pdMS_TO_TICKS(50));
      }
```

```
      childCry = false; // ریست فلگ
    }
    vTaskDelay(pdMS_TO_TICKS(10)); // بررسی سریع
  }
}
void onChildCryInterrupt() {
  Serial.println("Cryy");
  if(childProtect){
  childCry = true;
  }
}
void webServerTask(void *pvParameters) {
    while(true) {
        server.handleClient();
        vTaskDelay(pdMS_TO_TICKS(2));
    }
}
void cors(){
  server.send(200, "application/text", "CORS preflight response");

}
void coTask(void *pvParameters) {
    while(true) {

      int r=analogRead(co2_pin);
      if(r>=3000){
        addLog(ERROR,"The CO amount is higher than standard and it's dangerous");
danger=true;

      }
      else {

        danger=false;
      }
      vTaskDelay(pdMS_TO_TICKS(100));
    }
}
void setup() {
    Serial.begin(115200);
      Serial.print("Connecting to WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
```

```
    }
    Serial.println("\nWiFi connected!");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
    server.enableCORS();
server.on("/settime",HTTP_POST, handleSetTimeRequest);
server.on("/settime",HTTP_OPTIONS, cors);

            server.on("/off", HTTP_POST,handleOffRequest);
                        server.on("/off", HTTP_OPTIONS,cors);

      server.on("/reboot", HTTP_POST,handleRebootRequest);
                        server.on("/reboot", HTTP_OPTIONS,cors);

          server.enableCORS(true);
    server.on("/statistics",HTTP_GET, handleDataRequest);
    server.on("/statistics",HTTP_OPTIONS, cors);

      server.on("/logs",HTTP_GET, handleLogsRequest); // <-- این خط جدید
                  server.on("/logs", HTTP_OPTIONS,cors);

      server.enableCORS(true);

      server.on("/config", HTTP_POST,handleConfigRequest);
            server.on("/config", HTTP_OPTIONS,cors);

server.onNotFound([]() {
  Serial.print("Method: ");
Serial.println((int)server.method());
Serial.println(server.uri());

  /*if (server.method() == HTTP_OPTIONS) {
        server.sendHeader(F("Access-Control-Max-Age"), F("600"));

    server.sendHeader("Access-Control-Allow-Origin", "*");
    server.sendHeader("Access-Control-Allow-Methods", "GET,POST,OPTIONS");
    server.sendHeader("Access-Control-Allow-Headers", "*");
    server.send(204); // No Content
  }*/
    server.send(404, "text/plain", "Not found");

});
```

```
    server.begin(); // راه‌اندازی وب سرور
    pinMode(co2_pin, INPUT);

    Serial.println("HTTP server started. Open /data endpoint to see JSON data.");
    // ==========================================================
    xTaskCreate(webServerTask, "WebServer", 8192, NULL,1, &webServerTaskHandle);
vTaskSuspend(webServerTaskHandle); // وب سرور را موقتا غیرفعال می‌کنیم
 logQueue = xQueueCreate(10, sizeof(LogEntry)); // صف با ظرفیت ۱۰ لاگ
  xTaskCreate(logProcessingTask, "LogProcessor", 2048, NULL, 1, NULL);
  xTaskCreate(coTask,"CO task",2048,NULL,1,&co2Task);
  vTaskSuspend(co2Task);
    Wire.begin(21, 22);
    pinMode(LDR_PIN, INPUT);
      if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

  if (!rtc.isrunning()) {
    Serial.println("RTC lost power, lets set the time!");

    // Comment out below lines once you set the date & time.
    // Following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

    // Following line sets the RTC with an explicit date & time
    // for example to set January 27 2017 at 12:56 you would call:
    // rtc.adjust(DateTime(2017, 1, 27, 12, 56, 0));
  }

    DateTime now = rtc.now();

  sprintf(uptimeString, "%02d:%02d:%02d", now.hour(), now.minute(),
now.second());
  sprintf(updateString, "%d/%02d/%02d", now.year(), now.month(), now.day());

    door.attach(door_pin, 500, 2400); // فراخوانی می‌شود setup فقط یک بار در attach تابع
pinMode(buzzer_pin, OUTPUT);
xTaskCreate(buzzerTask, "Buzzer Task", 2048, NULL, 2, NULL);
pinMode(child_cry_pin, INPUT_PULLUP); // دکمه باید به زمین وصل باشد
attachInterrupt(digitalPinToInterrupt(child_cry_pin), onChildCryInterrupt,
FALLING);

pinMode(room_1_led_blue_pin,OUTPUT);
```

```
pinMode(room_2_led_blue_pin,OUTPUT);
pinMode(room_1_led_green_pin,OUTPUT);
pinMode(room_2_led_green_pin,OUTPUT);
pinMode(room_1_led_red_pin,OUTPUT);
pinMode(room_2_led_red_pin,OUTPUT);
ESP32PWM::allocateTimer(0);  // اختیاری ولی مفیده
ESP32PWM::allocateTimer(1);  // اختیاری ولی مفیده
ESP32PWM::allocateTimer(2);  // اختیاری ولی مفیده
ESP32PWM::allocateTimer(3);  // اختیاری ولی مفیده

fan.setPeriodHertz(50);      // برای سرووها ضروریه
fan.setPeriodHertz(50);      // برای سرووها ضروریه
    fan2.setPeriodHertz(50);      // برای سرووها ضروریه
    fan3.setPeriodHertz(50);      // برای سرووها ضروریه
    fan2.attach(ROOM_1_FAN, 500, 2400); // فراخوانی می‌شود setup فقط یک بار در attach تابع
   fan.attach(SERVO_PIN, 500, 2400); // فراخوانی می‌شود setup فقط یک بار در attach تابع
   fan3.attach(ROOM_2_FAN, 500, 2400); // فراخوانی می‌شود setup فقط یک بار در attach تابع

    if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_I2C_ADDR)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;);
    }

    pinMode(LED_RED_PIN, OUTPUT);
    pinMode(LED_GREEN_PIN, OUTPUT);
    pinMode(LED_BLUE_PIN, OUTPUT);

    irQueue = xQueueCreate(5, sizeof(uint8_t));

    xTaskCreate(partyTask, "Party Task", 2048, NULL, 1, &partyTaskHandle);
    if (partyTaskHandle != NULL) {
                                    partyMode=false;

        vTaskSuspend(partyTaskHandle);
    }

    xTaskCreate(ldrTask, "LDR Task", 2048, NULL, 1, &ldrTaskHandle);
    if (ldrTaskHandle != NULL) {
        vTaskSuspend(ldrTaskHandle);
    }
 xTaskCreate(clockTask, "Clock Task", 2048, NULL, 1, &clockTaskHandle);
    if (ldrTaskHandle != NULL) {
        vTaskSuspend(clockTaskHandle);
    }
```

```
   xTaskCreate(IRTask, "IR Receiver", 2048, NULL, 2, NULL);
    xTaskCreate(mainControlTask, "Main Control", 4096, NULL, 1, NULL);
   // xTaskCreate(FanTask, "Fan Task", 2048, NULL, 1, &coolerTask);
    xTaskCreatePinnedToCore(
  Fan1Task,     // function
  "Fan1 Task",  // name
  2048,       // stack
  NULL,        // param
  1,           // priority
  &cooler1Task,        // handle
  0            // <- core 0
);
    xTaskCreatePinnedToCore(
  Fan2Task,     // function
  "Fan2 Task",  // name
  2048,       // stack
  NULL,        // param
  1,           // priority
  &cooler2Task,        // handle
  0            // <- core 0
);
    xTaskCreatePinnedToCore(
  Fan3Task,     // function
  "Fan3 Task",  // name
  2048,       // stack
  NULL,        // param
  1,           // priority
  &cooler3Task,        // handle
  0            // <- core 0
);
    xTaskCreatePinnedToCore(
        DHTTask, "DHT Task", 4096, NULL, 3, &dhtHandle, 1);
   // ==========================================================
    xTaskCreate(displayUpdateTask, "Display Task", 4096, NULL, 2, NULL);

    delay(500);
}

void loop() {
    vTaskDelete(NULL);
}
```