

خانه هوشمند

Smart Home

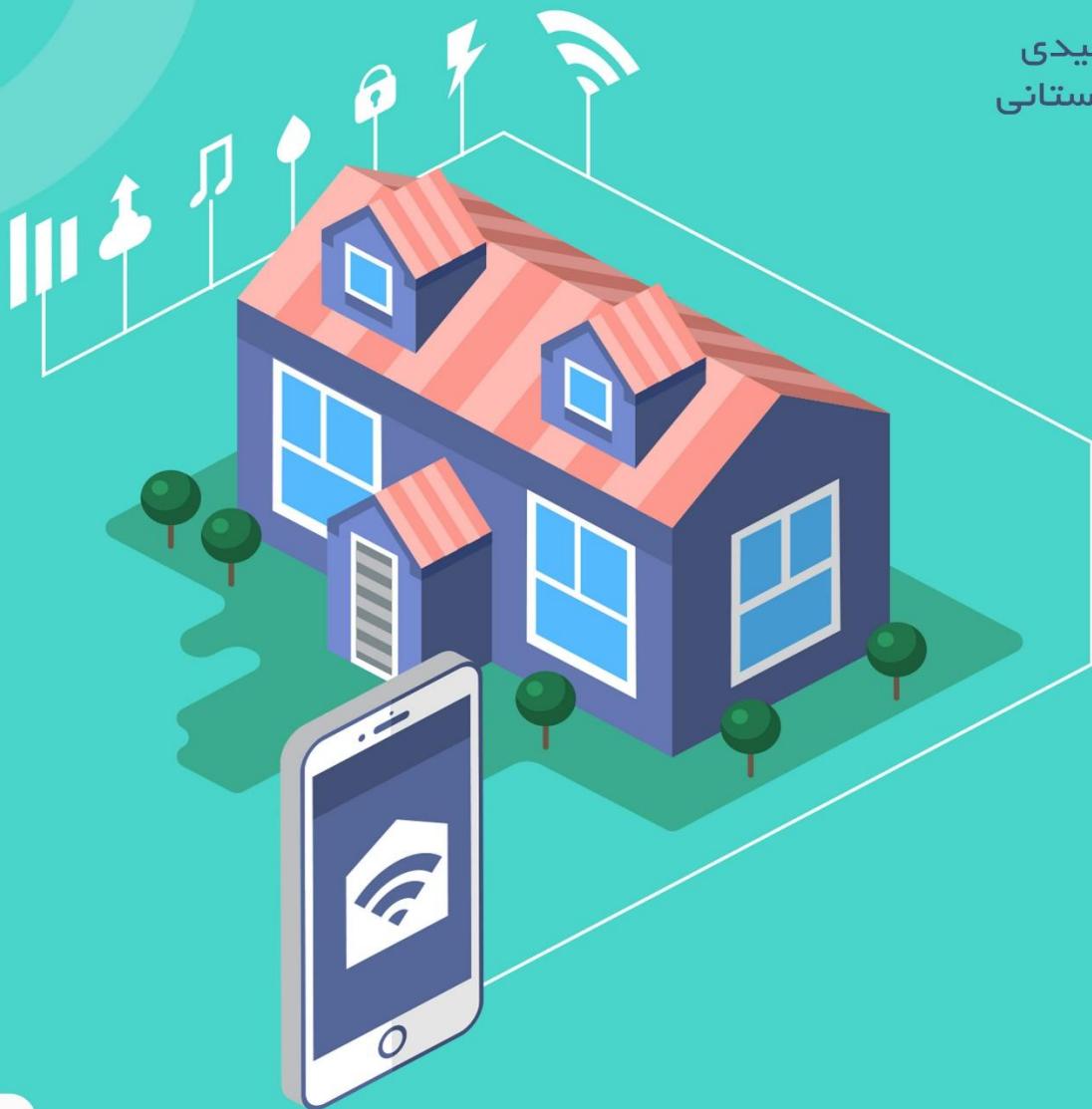
پروژه درس ریز پردازندہ

مبتنی بر Arduino

نیم سال اول ۱۴۰۳

دکتر محسن عباسی

وحید سیدی
زکیه داستانی



GitHub

فهرست مطالب

۳ مقدمه
۴ ۱- اهداف پروژه
۵ ۲- ابزار ها و تجهیزات استفاده شده
۶ ۳- معماری سیستم
۷ ۴- معماری نرم افزار
۱۰ ۵- چالش ها
۱۲ ۶- نقشه
۱۳ ۷- توضیحات کد ESP8266 MCU
۲۰ ۸- توضیحات کد WEB PAGE
۳۶ ۹- توضیحات کد Arduino UNO
۴۳ ۱۰- تقدیر و تشکر

مقدمه و تعریف پروژه

در دنیای امروز، مفهوم خانه هوشمند به یکی از مهم‌ترین جنبه‌های فناوری‌های نوین تبدیل شده است. سیستم‌های خانه هوشمند با هدف بهینه‌سازی مصرف انرژی، افزایش رفاه و امنیت، و تسهیل مدیریت خانه طراحی می‌شوند. این فناوری‌ها به کاربران اجازه می‌دهند تا دستگاه‌ها و تجهیزات خانگی را از راه دور کنترل کنند، اطلاعات محیطی را جمع‌آوری کرده و تحلیل کنند، و شرایط ایده‌آلی برای زندگی روزمره فراهم آورند.

پروژه حاضر با استفاده از برد آردوینو به طراحی و پیاده‌سازی یک سیستم خانه هوشمند ساده می‌پردازد. در این سیستم از سنسورها و ماژول‌های مختلف برای نظارت بر شرایط محیطی مانند دما و رطوبت، کنترل روشنایی، و تشخیص حرکت استفاده شده است. همچنین، قابلیت کنترل تجهیزات خانه از طریق نرمافزار یا اپلیکیشن فراهم شده است تا کاربر بتواند از راه دور به مدیریت خانه بپردازد.

این پروژه نه تنها جنبه‌های تئوری و عملی مرتبط با طراحی سیستم‌های تعییه‌شده را بررسی می‌کند، بلکه به عنوان نمونه‌ای از کاربردهای واقعی اینترنت اشیا (IoT) در زندگی روزمره ارائه می‌شود. هدف اصلی این سیستم، ارائه یک راه حل ساده، کم‌هزینه، و کاربردی برای بهبود کیفیت زندگی و افزایش امنیت در خانه است.

اهداف پروژه

۱. بهینه‌سازی مصرف انرژی

طراحی سیستمی که با مدیریت هوشمند منابع مانند روشنایی و وسایل برقی، مصرف انرژی را کاهش دهد و از هدررفت آن جلوگیری کند.

۲. افزایش امنیت خانه

استفاده از سنسورهای تشخیص حرکت برای شناسایی فعالیتهای مشکوک و ارسال هشدار در موقع ضروری.

۳. تسهیل کنترل تجهیزات خانگی

فراهم کردن امکان کنترل وسایل برقی و سیستم روشنایی از راه دور از طریق اپلیکیشن موبایل یا رابط کاربری تحت وب.

۴. نظارت بر شرایط محیطی

مانیتورینگ پارامترهای محیطی مانند دما و رطوبت به منظور ایجاد شرایط ایدهآل در فضای خانه.

۵. ایجاد یک نمونه مقرر به صرفه از سیستم خانه هوشمند

طراحی سیستمی ساده و کم‌هزینه با استفاده از برد آردوینو و قطعات متداول برای کاربران با دانش فنی اولیه.

۶. آموزش و یادگیری تکنولوژی‌های مرتبط با اینترنت اشیا(IoT)

استفاده از این پروژه به عنوان بستری برای درک بهتر مفاهیم مرتبط با IoT، سنسورها، و ارتباطات بی‌سیم.

۷. افزایش رفاه و آسایش کاربران

کاهش نیاز به انجام کارهای دستی و ایجاد امکاناتی برای راحت‌تر شدن مدیریت خانه.

ابزارها و تجهیزات استفاده شده



Buzzer



Push Button



LDR



LED



DHT22



مقاومة و سیم



PIR



MQ9



Relay 2ch



ESP8266

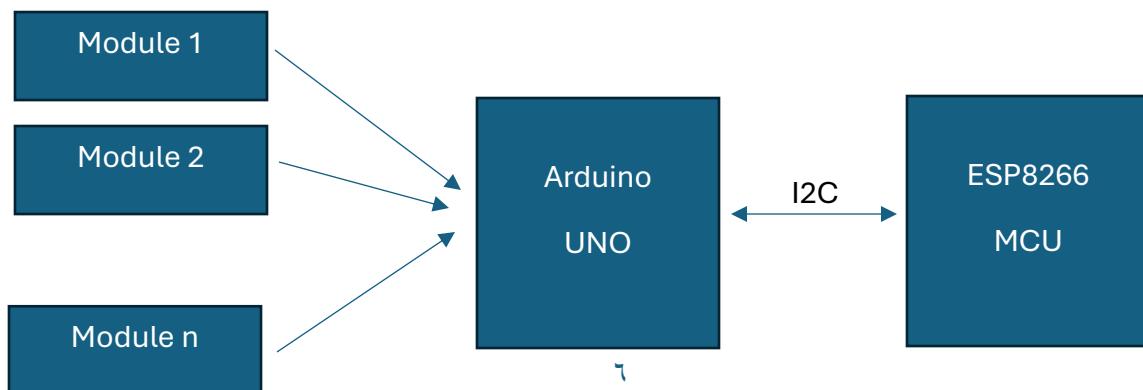


Arduino UNO

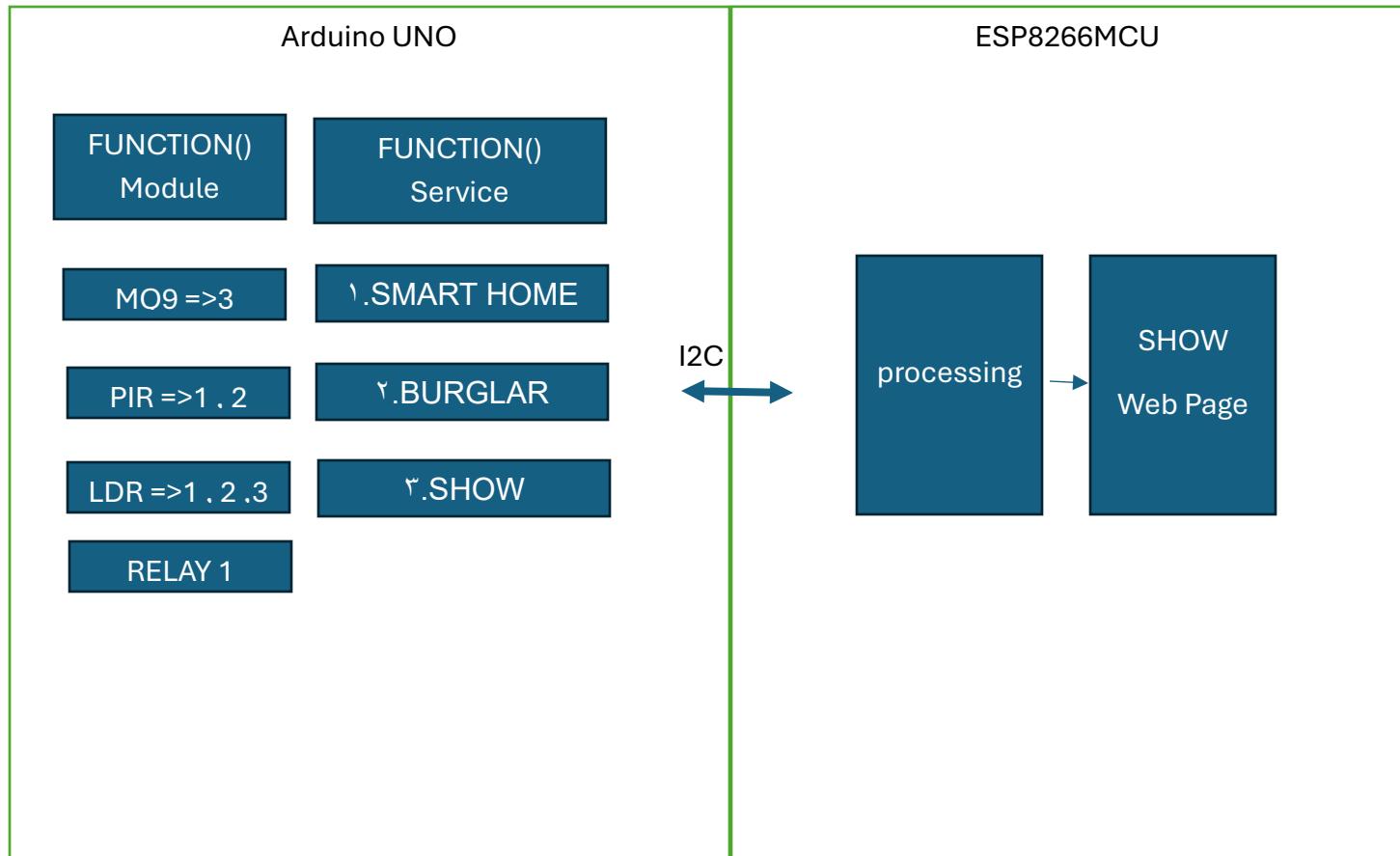
معماری سیستم

سیستم خانه هوشمند طراحی شده مبتنی بر برد آردوینو، از بخش‌های زیر تشکیل شده است:

- سنسورها:
 - DHT22 برای اندازه‌گیری دما و رطوبت محیط.
 - PIR برای تشخیص حرکت در محیط.
 - LDR (Light Dependent Resistor) برای تشخیص سطح نور محیط (اختیاری).
- ماژول ارتباطی:
 - ESP8266MCU: برای ارتباط وای‌فای و ارسال اطلاعات به اپلیکیشن یا سرور.
- عملگرها:
 - ماژول رله: برای کنترل وسایل برقی مانند چراغ‌ها، پنکه، یا بخاری.
 - LED‌ها: برای نشان دادن وضعیت سیستم.
- برد آردوینو:
 - مغز اصلی سیستم که داده‌ها را از سنسورها دریافت کرده، پردازش می‌کند و دستورات لازم را به عملگرها ارسال می‌کند.
- اپلیکیشن یا رابط کاربری تحت وب:
 - برای کنترل از راه دور تجهیزات خانه و مشاهده اطلاعات محیطی.



معماری نرم افزار



۱. دریافت داده از حسگرهای (Sensors Layer)

در این بخش، حسگرهای متصل به برد آردوینو (مانند DHT22، PIR، LDR) داده‌های محیطی را جمع‌آوری می‌کنند. برای هر حسگر، یک تابع اختصاصی در کد آردوینو تعریف شده است که وظیفه خواندن داده و آماده‌سازی آن برای مراحل بعدی را بر عهده دارد.

این توابع به صورت ماثولار طراحی شده‌اند تا بتوان به راحتی حسگرهای جدید را به سیستم اضافه کرد.

۲. مدیریت داده‌ها در آردوینو (Arduino Data Handling)

داده‌های جمع‌آوری شده توسط توابع حسگرها در آردوینو پردازش می‌شوند.

پردازش اولیه شامل:

تعیین وضعیت حرکت یا عدم حرکت) برای PIR).

مقایسه مقادیر دما و رطوبت با حد مجاز.

تشخیص شدت نور محیط.

داده‌های پردازش شده به صورت بسته‌های اطلاعاتی از طریق پروتکل I2C به ماژول ESP8266 ارسال می‌شوند.

۳. انتقال داده به ESP8266 (Communication Layer)

برد آردوینو از پروتکل I2C برای انتقال داده‌ها به ESP8266 MCU استفاده می‌کند.

I2C به عنوان یک پروتکل سریع و ساده برای ارتباط داخلی بین ماژول‌ها استفاده شده است.

۴. پردازش داده در ESP8266 (Processing Layer)

ESP8266 به عنوان پردازشگر مرکزی وظیفه دریافت داده از آردوینو و آماده‌سازی آن برای نمایش در صفحه وب را بر عهده دارد.

وظایف اصلی ESP8266 شامل:

تجزیه و تحلیل داده‌های دریافتی از آردوینو.

تصمیم‌گیری درباره دستورات ارسال شده از طریق صفحه وب.

ارسال داده‌های پردازش شده به رابط کاربری.

۵. رابط کاربری تحت وب (Presentation Layer)

داده‌های پردازش شده در ESP8266 به صورت پویا در یک رابط وب به کاربر نمایش داده می‌شوند.

رابط وب شامل:

نمایش مقادیر دما، رطوبت، وضعیت حرکت، و شدت نور.

کنترل تجهیزات برقی متصل (مانند روشن/خاموش کردن چراغ‌ها).

ارسال دستورات به ESP8266 از طریق درخواست‌های HTTP یا WebSocket.

این رابط کاربری با استفاده از یک وب‌서ور داخلی که روی ESP8266 اجرا می‌شود، پیاده‌سازی شده است.

چالش ها

چالش در مورد تامین انرژی مورد نیاز برد esp8266 mcu است .

در حالت آزمایشی ما یک کابل USB ولتاژ مورد نیاز ESP را تامین میکردیم ولی در حالت اجرای عملیاتی اون ، نیاز داشتیم که توسط آردیونو تامین ولتاژ صورت بگیره و کاری که انجام دادیم :

یک پنج ولت و یک GND به پایه V in و GND در ESP متصل کردیم .

در حالت اولیه :

در صورتی که برد آردیونو اتصال داشت برد ESP هم متصل میشد و مشکلی پیش نمی اومد اما پس از قطع برد آردیونو از منبع ولتاژ و اتصال دوباره آن ، برد آردیونو راه اندازی میشد ولی برد ESP فعال نمیشد.

پس از جست و جوهای متفاوت و پیدا نکردن جواب های قانع کننده و بدرد بخور نکته ای رو فهمیدیم که در همون حالت در صورت روی استارت کردن ESP (با Button روی برد) برد ESP فعال و به کار خود ادامه میده (احتمالا دلیل این اتفاق بحث حالت بهینه سازی انرژی فعال شده و با عث قطعی می شده)

و پس از فهمیدن این موضوع که با دکمه روی استارت این مشکل حل میشه و با توجه به اینکه برد ESP دارای پایه rest است و به روی استارت متصل است و در حالت عادی در حالت HIGH قرار دارد ، برای روی استارت کردن برد نیاز داره که ولتاژ به LOW برسه و مجدد به HIGH برسه، عملا برد روی استارت میشه .

برای این کار میتونستیم ما از برد آردیونو به صورت مستقیم به پایه Rst وصل کنیم

اما مشکلی که وجود داشت این بود که ولتاژ کاری آردیونو ۵V بود و ولتاژ کاری ESP8266 ۳.۳V بود و اگر به صورت مستقیم وصل میشد احتمالا باعث خرابی برد ESP میشد.

بعد از اون تصمیم گرفتیم از متدهای تقسیم ولتاژ استفاده کنیم

با قرار دادن یک مدار ستاره که از دو مقاومت تشکیل شده بود ، مقاومت های $1\text{ k}\Omega$ و $2.2\text{ k}\Omega$ بتونیم برد رو روی استارت کنیم.

و در قسمت برد آردیونو در محل setup() یک pinMode گذاشتیم و digitalWrite تعریف کردیم و در نیم ثانیه LOW میشه و بعد از اون در حالت HIGH قرار میگیره و کار میکنه

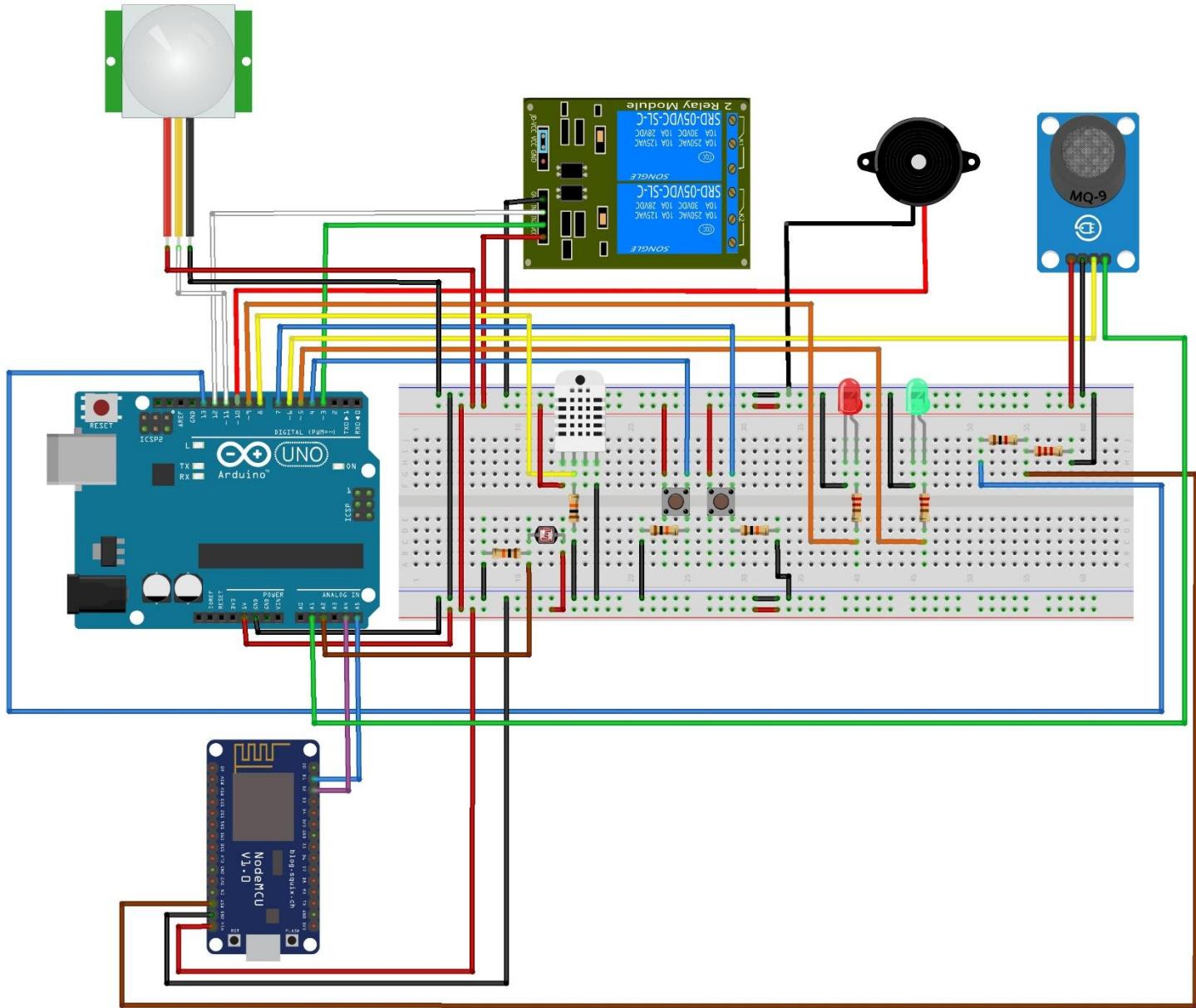
موضوع مهم ترین چالش ما در این پروژه بود

و البته چالش های دیگری نیز بود ، به طور مثال با توجه به اینکه آردیونو یه صورت خط به خط اجرا میکنه و نمیتونه به صورت موازی یک فعالیت رو بررسی کنه ، مشکلاتی از این قبیل داشتیم که باید در طول تابع هامون از flag ها و پرچم هایی استفاده کنیم تا بتوانیم این مشکلات رو برطرف کنیم

به همین دلیل پروژه دارای flag های بسیار زیادی است...

همچنین flag ها در بعضی از بخش ها برای اتصال تابع ها به همدیگه و بعضی موقع نیز برای قطع کردن اون تابع استفاده شده.

نقشه



توضیحات کد

ESP8266 MCU

- `#include <Wire.h>`

این دستور دایرہ المعارف **Wire** را وارد کد می کند که برای ارتباط **I2C** استفاده می شود.

- `#include <ESP8266WiFi.h>`

این دستور کتابخانه **ESP8266WiFi** را وارد می کند که امکان استفاده از **Wi-Fi** را در **ESP8266** فراهم می کند.

- `#include <ESP8266WebServer.h>`

این دستور کتابخانه **ESP8266WebServer** را وارد می کند که برای ایجاد یک وب سرور در **ESP8266** استفاده می شود.

- `#include <ArduinoJson.h>`

در کد ارائه شده، ابتدا کتابخانه های مورد استفاده برای اتصال **ESP8266** به وای فای، ایجاد وب سرور و کار با **JSON** از طریق **ArduinoJson** وارد شده اند. سپس نام شبکه و رمز عبور وای فای تعریف شده و یک وب سرور با شماره پورت ۸۰ ایجاد شده است. همچنین یک ثابت با نام **ARDUINO_ADDRESS** تعریف شده است.

```
● ● ●  
#include <Wire.h>  
#include <ESP8266WiFi.h>  
#include <ESP8266WebServer.h>  
#include <ArduinoJson.h>  
const char *ssid = "Smart Home";  
const char *password = "12345678";  
  
ESP8266WebServer server(80);  
#define ARDUINO_ADDRESS 0x08
```

تعریف متغیرها

```
const char *ssid = "Smart Home";      // نام شبکه وایفای (SSID)  
const char *password = "12345678";    // رمز عبور وایفای  
  
ESP8266WebServer server(80);        // سرور وب روی پورت 80  
#define ARDUINO_ADDRESS 0x08          // آردوینو I2C آدرس
```

رابط کاربری

```
const char index_html[] PROGMEM = R"rawliteral(  
....  
)rawliteral";
```

اینجا کد HTML برای رابط کاربری ذخیره می‌شود، (بعدا توضیح داده میشده)

تابع setup

```
void setup() {  
  Serial.begin(115200);           // اغاز ارتباط سریال برای اشکالزدایی  
  Wire.begin(4, 5);               // I2C پین‌های (SCL و SDA) برای پین‌های (GPIO4 و GPIO5)  
  WiFi.softAP(ssid, password);   // ایجاد شبکه وایفای در حالت Access Point
```

تعریف مسیرهای سرور:

```
server.on("/", []() { // درخواست به آدرس اصلی "/"  
    server.send(200, "text/html", index_html); // ارسال صفحه HTML  
});  
server.on("/get-sensor-data", handleSensorData); // دریافت داده‌های سنسور  
server.on("/toggle-device", HTTP_POST, handleDeviceToggle); // کنترل دستگاهها  
server.on("/toggle-alarm", HTTP_POST, handleAlarmToggle); // کنترل آلام  
server.on("/toggle-smart", HTTP_POST, handleSmartToggle); // تغییر وضعیت هوشمند
```

شروع سرور:

```
server.begin();  
Serial.println("HTTP server started");  
}
```

تابع loop

```
void loop() {  
    server.handleClient(); // پردازش درخواست‌های وب
```

درباره داده‌های سنسور

```
void handleSensorData() {
    Wire.requestFrom(ARDUINO_ADDRESS, 32); // درخواست 32 بایت داده از آردوینو
    String data = ""; // متغیر ذخیره‌سازی داده‌ها
    while (Wire.available()) { // دریافت داده‌ها از I2C
        char c = Wire.read(); // چاپ داده دریافتی
        data += c;
    }
    Serial.println("Received: " + data); // چاپ داده دریافتی
```

پردازش داده‌ها:

```
StaticJsonDocument<200> json; // JSON تعریف سند
int tempIndex = data.indexOf("T:"); // پیدا کردن دما
int humIndex = data.indexOf(",H:"); // پیدا کردن رطوبت
int lightIndex = data.indexOf(",L:"); // پیدا کردن نور
int gasIndex = data.indexOf(",G:"); // پیدا کردن گاز
int alarmIndex = data.indexOf(",A:"); // پیدا کردن وضعیت آلرم
```

استخراج مقادیر:

```
// JSON بررسی وجود تمامی مقادیر و افزودن به
if (tempIndex != -1 && humIndex != -1 && lightIndex != -1 && gasIndex != -1) {
    json["temperature"] = data.substring(tempIndex + 2, humIndex).toFloat(); // دما
    json["humidity"] = data.substring(humIndex + 3, lightIndex).toFloat(); // رطوبت
    json["light"] = data.substring(lightIndex + 3, gasIndex).toInt(); // نور
    json["gas"] = data.substring(gasIndex + 3, alarmIndex).toInt(); // گاز
    json["alarm"] = data.substring(alarmIndex + 3).toInt(); // آلارم
}
```

ارسال پاسخ:JSON

```
String response;
serializeJson(json, response); // تبدیل به رشته JSON
server.send(200, "application/json", response); // ارسال پاسخ
}
```

کنترل دستگاهها

```
void handleDeviceToggle() {
    StaticJsonDocument<200> json; // تعریف سند JSON
    String body = server.arg("plain"); // دریافت داده POST
    deserializeJson(json, body); // پردازش JSON
    String device = json["device"]; // نام دستگاه
    bool status = json["status"]; // وضعیت دستگاه
```

ارسال دستور به آردوینو:

```
Wire.beginTransmission(ARDUINO_ADDRESS);
if (device == "lamp") {
    Wire.write(status ? 'L' : 'l'); // کنترل لامپ
} else if (device == "cooling") {
    Wire.write(status ? 'C' : 'c'); // کنترل خنک کننده
}
Wire.endTransmission();
server.send(200, "text/plain", "OK"); // پاسخ موفقیت
}
```

کنترل آلام

```
// تابع برای کنترل آلام
void handleAlarmToggle() {
    StaticJsonDocument<200> json; // JSON تعریف سند
    String body = server.arg("plain"); // POST دریافت داده
    deserializeJson(json, body); // JSON تجزیه
    bool status = json["status"]; // وضعیت آلام (فعال یا غیرفعال)

    // ارسال دستور به آردوینو
    Wire.beginTransmission(ARDUINO_ADDRESS);
    Wire.write(status ? 'A' : 'a'); // فعال یا غیرفعال کردن آلام
    Wire.endTransmission(); // پایان انتقال داده

    server.send(200, "text/plain", "OK"); // ارسال پاسخ موفقیت
}
```

smart home وضعیت تغییر

```
تابع برای تنظیمات هوشمند //  
void handleSmartToggle() {  
    StaticJsonDocument<200> json;          // JSON تعریف سند  
    String body = server.arg("plain");     // دریافت داده POST  
    deserializeJson(json, body);           // تجزیه JSON  
  
    int status = json["status"];           // دریافت مقدار وضعیت هوشمند //  
  
    ارسال دستور مناسب به آردوینو //  
    Wire.beginTransmission(ARDUINO_ADDRESS);  
    if (status == 0) {  
        Wire.write('s');                  // غیرفعال کردن هوشمند //  
    } else if (status == 1) {  
        Wire.write('1');                  // حالت شدت روشنایی (LDR)  
    } else if (status == 2) {  
        Wire.write('2');                  // حالت حرکتی (PIR)  
    }  
    Wire.endTransmission();               // پایان انتقال داده //  
  
    server.send(200, "text/plain", "OK"); // ارسال پاسخ موفقیت //  
}
```

توضیحات کد

WEB PAGE

این بخش از کد **HTML** و **CSS** تعریف می‌کند که یک صفحه وب ساده است که شامل تنظیمات کدگذاری، تنظیمات نمایش برای دستگاه‌های مختلف، و تعریف یک فونت به نام "Dirooz" است که از فایل‌های مختلف فونت استفاده می‌کند.

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
@font-face {
    font-family: Dirooz;
    src: url('../fonts/Dirooz.eot');
    src: url('../fonts/Dirooz.eot?#iefix') format('embedded-
opentypeurl('../fonts/Dirooz.woff') format('woff'),
        url('../fonts/Dirooz.ttf') format('truetype'));
    font-weight: normal;
}

```

در این قسمت، یک فونت به نام "Dirooz" تعریف شده است که از فایل‌های مختلف فونت استفاده می‌کند (**TTF**, **WOFF**, **EOT**)

تمامی محتویات صفحه از نوع فونت "Dirooz" استفاده می‌کند و حاشیه و فاصله‌ها را حذف می‌کند.

یک هدر بالای صفحه با پس زمینه سفید و سایه، و امکان چینش افقی ایجاد می‌کند.

تنظیمات مربوط به عنوان صفحه که در وسط نمایش داده می‌شود و با وزن **bold** تعریف شده است.

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
@font-face { body {
    font-family: Dirooz;
    margin: 0;
    padding: 0;
}

.header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    /* Space between items */
    padding: 10px 20px;
    background-color: #fff;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    position: relative;
}

.title {
    flex-grow: 1;
    /* Allows title to take up available space */
    text-align: center;
    /* Center the title text */
    font-weight: bold;
    /* Optional: make the title bold */
}
    font-family: Dirooz;
    src: url('../fonts/Dirooz.eot');
    src: url('../fonts/Dirooz.eot?#iefix') format('embedded-opentype'),
        url('../fonts/Dirooz.woff') format('woff'),
        url('../fonts/Dirooz.ttf') format('truetype');
    font-weight: normal;
}
)rawliteral";
```

یک آواتار که یک دایره با پس زمینه خاکستری است و یک وضعیت (به صورت یک دایره سبز) را نشان می‌دهد.

محتوای دکمه‌ها را در یک محیط نمایش می‌دهد و از فاصله بین دکمه‌ها استفاده می‌کند.

- تنظیمات مربوط به دکمه‌ها که شامل رنگ پس زمینه، گوشه‌های گرد، حالت های `focus` و `hover`، و اینیمیشن‌های CSS مختلف است. و `btn` `svg` تنظیمات مربوط به آیکون‌های SVG که از رنگ، ابعاد و تنظیمات دیگر استفاده می‌کنند.

این کد HTML و CSS، یک صفحه وب ساده و زیبا را تعریف می‌کند که می‌تواند برای نمایش اطلاعات یا ایجاد واسط کاربری در یک پروژه وب مفید باشد.



```
.avatar {
    width: 50px;
    height: 50px;
    border-radius: 50%;
    background-color: #ccc;
    position: relative;
}

.status {
    position: absolute;
    bottom: 0;
    right: 0;
    width: 15px;
    height: 15px;
    border-radius: 50%;
    background-color: green;
    border: 2px solid #fff;
}
```

```
.button-container {  
    display: flex;  
    gap: 10px;  
    /* Space between buttons */  
}  
  
.btn {  
    background-color: #ffffff;  
    /* Light background */  
    border: none;  
    /* Remove default border */  
    border-radius: 5px;  
    /* Rounded corners */  
    padding: 8px;  
    /* Padding for better click area */  
    cursor: pointer;  
    /* Pointer on hover */  
    transition: background-color 0.3s, transform 0.2s;  
    /* Animation effects */  
}
```

```
.btn:hover {  
    background-color: #e0e0e0;  
    /* Darker on hover */  
    transform: scale(1.05);  
    /* Slightly enlarge on hover */  
}  
  
.btn:focus {  
    outline: none;  
    /* Remove outline on focus */  
}  
  
.btn svg {  
    fill: #333;  
    /* Color for SVG icons */  
    width: 24px;  
    /* Adjust icon size */  
    height: 24px;  
    /* Adjust icon size */  
}
```

:status_box.

- مشخصات: متن رنگ سفید با پدینگ ۵ پیکسل، پس زمینه سبز با حاشیه ۲ پیکسل سفید، گوشه های گرد با شعاع ۷ پیکسل، و زیر بنای مطلق.

- موقعیت: سمت راست ۶۰ پیکسل، بالا به اندازه ۵۰٪ از نقطه مرکزی صفحه، و ترانسفورم برای تنظیم عمودی.

:container. .۲

- نمایش: نمایش از نوع **flex** با قابلیت **wrap** و فاصله ۲۰ پیکسل بین اجزا.

- ظاهر: پس زمینه **#f9f9f9** و پدینگ ۲۰ پیکسل.

:box. .۳

- انعطاف پذیری: **flex: 1 1 200px** به این معنا که انعطاف پذیری در پهنا، ارتفاع، و پهناي حداکثری ۲۰۰ پیکسل دارد.

- سایر مشخصات: انتقال مرکزی، جهت عمودی، حاشیه گرد با شعاع ۸ پیکسل، سایه با شعاع ۲ پیکسل، و اورفلو برای محتويات بيش از اندازه.

:box2..

- مشخصات مشابه **box** با انعطاف پذیری در ارتفاع.

:box23. و box22. .۵

- انعطاف پذیری با فاكتور ۲ و **box23** با فاكتور ۱.

:box1..۶

- انعطاف پذیری: ارتفاع خودکار با انعطاف پذیری در پهنا.

- سایر مشخصات: مشابه **box** با پدینگ ۲۰ پیکسل برای فضای بهتر.

```
● ● ●
```

```
.status_box {
    color: #fff;
    padding: 5px;
    background-color: green;
    border: 2px solid #fff;
    border-radius: 7px;
    position: absolute;
    right: 60px;
    top: 50%;
    transform: translateY(-50%);
}

.container {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    padding: 20px;
    background-color: #f9f9f9;
}

.box {
    flex: 1 1 200px;
    height: 150px;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
    position: relative;
    overflow: hidden;
}
```

```
● ● ●
```

```
.box2 {
    flex: 1 1 200px;
    height: auto;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
    position: relative;
    overflow: hidden;
}

.box22 {
    flex: 2;
}

.box23 {
    flex: 1;
}

.box1 {
    height: auto;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
    position: relative;
    overflow: hidden;
    padding: 20px;
    /* Added padding for better spacing */
}

.highlight {
    width: 100%;
    height: 30px;
    background-color: #2e7d32;
    position: absolute;
    bottom: 0;
    left: 0;
    border-bottom-left-radius: 8px;
    border-bottom-right-radius: 8px;
}
```

- **.weather**: تنظیمات: پس زمینه با رنگ **#1076d0** و تصویر زمینه از فایل **weather.svg** با ابعاد ۱۵۰x۱۵۰ پیکسل، و موقعیت مرکزی - .سایر تنظیمات: حاشیه ۱ پیکسلی با رنگ **#ccc**، گوشه های گرد با شعاع ۸ پیکسل، و اورفلو برای محتوا.



- **.titleInfo, .data**: تنظیمات: تعیین رنگ متن، اندازه قلم، و موقعیت مطلق برای متن.

- **.containerTitle**: تنظیمات: تعیین عرض، ارتفاع، و تنظیمات مربوط به زبانه های مرکزی.

- **.bcolor**: تنظیمات: پس زمینه با افقی خط کرنریلیتی خطی از رنگ های **#6439FF** و **#5c7ae6** به حداقل ارتفاع ۳۰۰ پیکس.

- :تنظیمات: تعیین رنگ متن برای موارد امنیتی.

```
● ● ●

.titleInfo {
    color: white;
    font-size: 25px;
    position: absolute;
    top: -20px;
    right: 20px;
}

.data {
    color: white;
    font-size: 35px;
}

.temperature {
    background-color: #ebbb0d;
    background-image: url('../src/temperature.svg');
    background-size: cover;
    background-position: center;
    background-size: 150px 150px;
    background-position: -45px -25px;
    background-repeat: no-repeat;
    position: relative;
    border: 1px solid #ccc;
    border-radius: 8px;
    overflow: hidden;
}
```

کد های HTML

```
<body>
  <header>
    <div class="header">
      <div class="button-container">
        <!-- نکمه خروج با ایکون -->
        <button class="btn">
          <object data="src/logout.svg" type="image/svg+xml" width="30" height="30"></object>
        </button>
        <!-- نکمه تنظیمات با ایکون -->
        <button class="btn">
          <object data="src/settings.svg" type="image/svg+xml" width="30" height="30"></object>
        </button>
        <!-- نکمه کنترل از دست رفته با ایکون -->
        <button class="btn">
          <object data="src/missing_controller.svg" type="image/svg+xml" width="30" height="30"></object>
        </button>
      </div>
      <!-- عنوان پنل مدیریت -->
      <div class="title">پنل مدیریت</div>
      <!-- ناحیه اواتار با وضعیت -->
      <div class="avatar">
        <div class="status"></div>
      </div>
    </div>
  </header>
```

```
<!-- اطلاعات حسگرها -->
<div class="infoBox container">
  <!-- دما و رطوبت -->
  <div class="temperature box">
    <p class="titleInfo">دما و رطوبت</p>
    <p class="data" id="temp-hum">0 0</p>
    <div class="highlight hTemperature"></div>
  </div>
  <!-- میزان گاز -->
  <div class="gas box">
    <p class="titleInfo">میزان گاز</p>
    <p class="data" id="gas">0%</p>
    <div class="highlight hGas"></div>
  </div>
  <!-- نور -->
  <div class="light box">
    <p class="titleInfo">نور</p>
    <p class="data" id="light">0</p>
    <div class="highlight hLight"></div>
  </div>
</div>
```

```
<!-- کنترل مستحکمها -->
<div class="container">
    <!-- ذذگیر -->
    <div class="box">
        <div class="containerTitle">
            <p>ذذگیر</p>
            <object data="src/verified_user.svg" type="image/svg+xml" width="30" height="30"></object>
        </div>
        <div id="toggleSwitchAlarms" class="toggle" onclick="toggleSwitch(this) ; toggleAlarm()">
            <div class="circle"></div>
        </div>
    </div>
    <!-- کنترل لامپ -->
    <div class="box">
        <div class="containerTitle">
            <p>کنترل لامپ</p>
            <object data="src/light_icon.svg" type="image/svg+xml" width="30" height="30"></object>
        </div>
        <div class="toggle" onclick="toggleSwitch(this) ; toggleDevice('lamp')">
            <div class="circle"></div>
        </div>
    </div>

```

```
<!-- سرمایشی -->
<div class="box">
    <div class="containerTitle">
        <p>سرمایشی</p>
        <object data="src/cold.svg" type="image/svg+xml" width="30" height="30"></object>
    </div>
    <div class="toggle" onclick="toggleSwitch(this) ; toggleDevice('cooling')">
        <div class="circle"></div>
    </div>
</div>

<!-- خانه هوشمند -->
<div class="container">
    <div class="bcolor box">
        <div class="security dis">
            <p>smart home</p>
        </div>
    </div>

```

```
<!-- دکمه فعال/غیرفعال کردن --!>
<button class="toggle-button off" id="toggleButton"
onclick="check( )">OFF</button>
<!-- فرم انتخاب نوع -->
<div id="smartHomeForm">
<form>
    <label class="label-container">
        <input type="radio" name="type-smart" value="2" required>
        <span class="custom-radio"></span>
        <span class="label-text">P I R</span>
    </label>
    <label class="label-container">
        <input type="radio" name="type-smart" value="1">
        <span class="custom-radio"></span>
        <span class="label-text">LDR</span>
    </label>
</form>
</div>
<div class="highlight hsmart">
</div>
</div>
</div>
```

```
<!-- اعلان هشدار گاز -->
<div class="overlay" id="overlay">
<div class="notification">
    مقدار گاز از حد مجاز عبور کرده است !!!
    <br>
    <button class="close-btn" onclick="closeNotification( )">بستن</button>
</div>
</div>

<!-- اعلان هشدار دزدگیر -->
<div class="overlay" id="overlay-alarm">
<div class="notification">
    دزدگیر یه چیزایی حس میکنه !!!! !
    <br>
    <button class="close-btn" onclick="closeNotificationAlarm( )">بستن</button>
</div>
</div>
```

```
<script>
<! -- کد های جاوا اسکریپت -->
</script>
```

```
// تغییر وضعیت دکمه به فعال/غیرفعال
function toggleSwitch(element) {
    element.classList.toggle('on');
}

// نمایش اعلان هشدار گاز
function showNotification() {
    const overlay = document.getElementById('overlay');
    overlay.style.display = 'flex';
}

// نمایش اعلان هشدار دزدگیر
function showNotificationAlarm() {
    const overlay = document.getElementById('overlay-alarm');
    overlay.style.display = 'flex';
}

// بستن اعلان هشدار گاز
function closeNotification() {
    const overlay = document.getElementById('overlay');
    overlay.style.display = 'none';
}
```

```

    // مدیریت هشدار دزدگیر
    if (data.alarm == 1 && deviceStates["alarm"] == true) {
        if (al == 1) {
            showNotificationAlarm();
            al = 0;
        }
    }
    // مدیریت هشدار گاز
    if (Math.round(data.gas * 100 / 1023) > 20) {
        showNotification();
    }
});

}

// بهروزرسانی دکمه‌ها با وضعیت جدید
function updateButton(id, state, labels) {
    const button = document.getElementById(id);
    button.textContent = `${labels.prefix}: ${state ? labels.on : labels.off}`;
    button.className = state ? 'btn btn-on' : 'btn btn-off';
}

```

```

const deviceStates = {
    lamp: false, // وضعیت لامپ
    cooling: false, // وضعیت سیستم سرمایشی
    alarm: false // وضعیت دزدگیر
};

// تغییر وضعیت دستگاه‌ها
function toggleDevice(device) {
    deviceStates[device] = !deviceStates[device];
    fetch('/toggle-device', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ device: device, status: deviceStates[device] })
    })
        .then(() => {
            updateButton(`${device}-btn`, deviceStates[device], {
                prefix: device === 'lamp' ? 'سیستم سرمایشی' : 'لامپ',
                on: 'روشن', off: 'خاموش'
            });
        })
        .catch(err => console.error(`Error toggling ${device}:`, err));
}

```

```
// تغییر وضعیت در دیگر
function toggleAlarm() {
    if (al == 0) {
        al = 1;
    }
    deviceStates.alarm = !deviceStates.alarm;
    fetch('/toggle-alarm', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ status: deviceStates.alarm })
    })
    .then(() => {
        updateButton('alarm-btn', deviceStates.alarm, {
            prefix: 'دزدگیر',
            on: 'فعال', off: 'غیرفعال'
        });
    })
    .catch(err => console.error("Error toggling alarm:", err));
}
```

```
let checkBox = 0; // وضعیت فرم هوشمند

// پیغامی فرم هوشمند
function check() {
    const selectedOption = document.querySelector('input[name="type-smart"]:checked');
    const button = document.getElementById('toggleButton');

    if (selectedOption) {
        checkBox = selectedOption.value;
        if (button.innerText === 'OFF') {
            button.innerText = 'ON';
            button.classList.add('on');
            button.classList.remove('off');
        } else {
            button.innerText = 'OFF';
            button.classList.remove('on');
            button.classList.add('off');
            checkBox = 0;
        }
        toggleSmartHome();
    } else {
        alert("لطفاً یک گزینه رو انتخاب کن");
    }
}
```

```
// فعال/غیرفعال کردن خانه هوشمند
function toggleSmartHome() {
    const formContainer = document.getElementById('smartHomeForm');
    const currentDisplay = window.getComputedStyle(formContainer).display;

    // نمایش یا پنهان کردن فرم
    formContainer.style.display = currentDisplay === 'none' ? 'block' : 'none';

    // ارسال درخواست fetch
    fetch('/toggle-smart', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ status: checkBox }),
    })
        .then((response) => response.json())
        .then((data) => {
            console.log("Server Response:", data);
        })
        .catch((error) => {
            console.error("Error:", error);
        });
}
```

```
// غیرفعال کردن مزدگیر از طریق ID
function deactivateToggleById() {
    const toggle = document.getElementById("toggleSwitchAlarms");
    if (toggle && toggle.classList.contains("active")) {
        toggle.classList.remove("active");
        toggleAlarm();
    }
}

// بهروزرسانی اطلاعات هر 2 ثانیه
setInterval(fetchData, 2000);
fetchData();
```

توضیحات کد

ARDUINO UNO

```
#include <Wire.h> // کتابخانه برای ارتباط I2C
#include <DHT.h> // کتابخانه برای سنسور DHT
#include <DHT_U.h> // کتابخانه برای نسخه یونیورسال DHT

#define DHTPIN 8 // پین اتصال DHT
#define DHTTYPE DHT22 // نوع سنسور DHT
DHT dht(DHTPIN, DHTTYPE); // تعریف شیء برای کار با DHT

int DO = 6; // پین سنسور دود
int ledRedPin = 9; // قرمز LED پین
int ledGreenPin = 5; // سبز LED پین
int buzzerPin = 10; // پین بوزر
int button1Pin = 7; // پین کلید 1
int button2Pin = 4; // پین کلید 2
int pirPin = 11; // پین سنسور PIR
int rel1Pin = 12; // پین رله 1
int rel2Pin = 3; // پین رله 2
int rstESP = 13; // پین ریست ESP
```

```
// متغیرهای فلگ برای مدیریت حالتها
int pirState = LOW; // حالت پیشفرض سنسور PIR
bool relay1 = 0; // وضعیت رله 1
bool relay2 = 0; // وضعیت رله 2
bool flagPir = 0; // فلگ برای دزدگیر
int flagS = 0; // فلگ برای حالتهای دیگر
bool flagSmartHome = 0; // فلگ برای خانه هوشمند
bool alarm = 0; // فلگ هشدار
bool Continue = 0; // ادامه عملیات

// دادههای سنسورها
int but1 = 0; // وضعیت کلید 1
int but2 = 0; // وضعیت کلید 2
int pot_t2 = 0; // مقدار پتانسیومتر
int ldr_t2 = 0; // مقدار سنسور نور
int temp_t2 = 0; // مقدار دما
int hum_t2 = 0; // مقدار رطوبت
int mq9_t2 = 0; // مقدار سنسور گاز
```

```
void setup() {  
    Wire.begin(8); // با آدرس 8 I2C مقداردهی اولیه  
    Wire.onReceive(receiveEvent); // تنظیم رویداد دریافت داده  
    Wire.onRequest(requestEvent); // تنظیم رویداد درخواست داده  
    Serial.begin(9600); // مقداردهی اولیه ارتباط سریال  
  
    // تنظیم پین‌ها  
    pinMode(buzzerPin, OUTPUT);  
    pinMode(pirPin, INPUT);  
    pinMode(ledRedPin, OUTPUT);  
    pinMode(ledGreenPin, OUTPUT);  
    pinMode(button1Pin, INPUT);  
    pinMode(rel1Pin, OUTPUT);  
    pinMode(rel2Pin, OUTPUT);  
    pinMode(DO, INPUT);  
    pinMode(rstESP, OUTPUT);  
    dht.begin(); // DHT مقداردهی اولیه سنسور
```

```
// تنظیم پیشفرض برای خروجی‌ها  
digitalWrite(rel1Pin, HIGH);  
digitalWrite(rel2Pin, HIGH);  
digitalWrite(rstESP, LOW);  
delay(500);  
digitalWrite(rstESP, HIGH);  
digitalWrite(ledGreenPin, HIGH); // سیز LED روشن کردن  
}
```

```

void loop() {
    // خواندن داده‌ها از سنسورها
    ldr_t2 = LDR();
    temp_t2 = temp();
    hum_t2 = hum();
    mq9_t2 = Mq9();

    int but1 = digitalRead(button1Pin); // 1
    int but2 = digitalRead(button2Pin); // 2

    // بررسی وضعیت سنسور گاز
    if (mq9_t2 > 204) {
        digitalWrite(buzzerPin, HIGH);
        digitalWrite(ledRedPin, HIGH);
        delay(1000);
        digitalWrite(buzzerPin, LOW);
        digitalWrite(ledRedPin, LOW);
        delay(1000);
    }
}

```

```

// 1 مدیریت کلید
if (but1 == 1) {
    flagPir = flagPir + 1;
    delay(200);
}

// 2 مدیریت کلید
if (but2 == 1) {
    flagS = flagS + 1;
    delay(200);
}

// اجرای سرویس دزدگیر
if (flagPir == 1) {
    ServiceBurglarAlarm(5, 1000);
}

// مدیریت خانه هوشمند
if (flagS == 1) {
    smartHomeService();
} else if (flagS > 1) {
    if (relay1 == 1) {
        digitalWrite(rel1Pin, LOW);
    } else {
        digitalWrite(rel1Pin, HIGH);
    }
}

```

```
● ● ●

if (relay2 == 1) {
    digitalWrite(rel2Pin, LOW);
} else {
    digitalWrite(rel2Pin, HIGH);
}
flagS = 0; // بازنگشانی فلگ
}
delay(100);
}

// توابع خواندن از سنسورها
int temp() {
    return dht.readTemperature();
}

int hum() {
    return dht.readHumidity();
}

int LDR() {
    int data = analogRead(A2);
    return data;
}
```



```
// مدیریت PIR
void pir() {
    int val = digitalRead(pirPin);
    if (val == HIGH && pirState == LOW) {
        pirState = HIGH;
    } else if (val == LOW && pirState == HIGH) {
        pirState = LOW;
    }
}

// خواندن مقدار سنسور MQ-9
float Mq9() {
    float sensorValue = analogRead(A1);
    return sensorValue;
}

// سرویس دزدگیر
void ServiceBurglarAlarm(int n, int del) {
    pir();

    if (pirState == HIGH || Continue == 1) {
        if (pirState == HIGH) {
            Continue = 1;
        }
        if (flagPir == 1) {
            for (int i = 0; i < n; i++) {
                alarm = 1;
                digitalWrite(buzzerPin, HIGH);
                digitalWrite(ledRedPin, HIGH);
                delay(del);
                digitalWrite(buzzerPin, LOW);
                digitalWrite(ledRedPin, LOW);
                delay(del);
            }
        } else {
            alarm = 0;
            digitalWrite(ledGreenPin, HIGH);
            delay(1000);
            digitalWrite(ledGreenPin, LOW);
            delay(1000);
        }
    }
}
```



```
// خانه هوشمند
void smartHomeService() {
    int lightLevel = ldr_t2;
    int currentTemp = temp_t2;
    int currentHum = hum_t2;

    if (flagSmartHome == 0) {
        if (lightLevel < 70) {
            digitalWrite(rel1Pin, LOW);
        } else {
            digitalWrite(rel1Pin, HIGH);
        }
    } else if (flagSmartHome == 1) {
        pir();
        if (pirState == HIGH) {
            digitalWrite(rel1Pin, LOW);
            delay(1500);
        } else {
            digitalWrite(rel1Pin, HIGH);
        }
    }

    if (currentTemp > 30) {
        digitalWrite(rel2Pin, LOW);
    } else if (currentTemp < 18) {
        digitalWrite(rel2Pin, HIGH);
    }
}
```



```
// مدیریت رویداد درخواست داده
void requestEvent() {
    String data = "T:" + String(temp_t2) + ",H:" + String(hum_t2) + ",L:" +
    String(ldr_t2) + ",G:" + String(mq9_t2) + ",A:" + String(alarm);
    Wire.write(data.c_str(), min(data.length(), 32));
    Serial.println(data);
}
```



```
مديريت رويداد دريافت داده //  
void receiveEvent(int howMany) {  
    while (Wire.available()) {  
        char command = Wire.read();  
        دستورات مختلف //  
        if (command == 'L') {  
            relay1 = 1;  
            digitalWrite(rel1Pin, LOW);  
        } else if (command == 'l') {  
            relay1 = 0;  
            digitalWrite(rel1Pin, HIGH);  
        } else if (command == 'C') {  
            relay2 = 1;  
            digitalWrite(rel2Pin, LOW);  
            relay2 = 0;  
        } else if (command == 'c') {  
            digitalWrite(rel2Pin, HIGH);  
        } else if (command == 'A') {  
            alarm = 0;  
            flagPir = 1;  
            Serial.println("Active Sevice");  
        } else if (command == 'a') {  
            alarm = 0;  
            flagPir = 0;  
            Continue = 0;  
            Serial.println("Deactive Service");  
        } else if (command == '1') {  
            Serial.println("Smart Home: LDR Mode");  
            flagSmartHome = 0;  
            flagS = 1;  
        } else if (command == '2') {  
            Serial.println("Smart Home: PIR Mode");  
            flagSmartHome = 1;  
            flagS = 1;  
        } else if (command == 's') {  
            Serial.println("Deactivating Smart Home");  
            flagS = 2;  
        }  
    }  
}
```

فایل ها و کد برنامه



با تشکر از دکتر محسن عباسی بابت تمام زحمات ایشان

<https://github.com/vahidseyyedi/Arduino-Smart-Home>