HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO CUỐI KÌ

HỌC PHẦN IOT VÀ ỨNG DỤNG

Đề tài: Thiết kế hệ thống IOT giám sát ánh sáng và điều khiển đèn tự động trong phòng học.

Giảng viên hướng dẫn: Kim Ngọc Bách

Lớp: D22CNPM02

Nhóm: 01

Thành viên:

- Đàm Anh Đức – B22DCCN219
- Lê Phương Nam – B22DCCN555
- Nguyễn Văn Nhất – B22DCCN579
- Nguyễn Việt Quang – B22DCCN651

Hà Nội – 2025

Mục Lục

I.Giới thiệu chung

1. Lý do chọn đề tài

Trong thực tế, hệ thống chiếu sáng trong phòng học thường được bật/tắt thủ công bởi giảng viên hoặc cán bộ kỹ thuật. Điều này dẫn đến nhiều bất cập:

- Đèn vẫn bật khi phòng không có người, gây lãng phí điện năng.
- Mức độ chiếu sáng không ổn định khi thời tiết thay đổi (trời nắng, trời âm u, buổi tối…), ảnh hưởng đến sự thoải mái và thị lực của người học.
- Việc kiểm soát chiếu sáng cho nhiều phòng học đôi khi đòi hỏi nhân lực và thời gian.

Sự phát triển của công nghệ IoT mở ra khả năng tự động hóa hệ thống chiếu sáng, biến các thiết bị truyền thống thành các "thiết bị thông minh", có khả năng cảm nhận môi trường, ra quyết định và kết nối Internet. Chính vì vậy, nhóm lựa chọn đề tài *"Thiết kế hệ thống IoT giám sát ánh sáng và điều khiển đèn tự động trong phòng học"* nhằm giải quyết các vấn đề trên và đồng thời ứng dụng kiến thức IoT vào một bài toán gần gũi với môi trường đại học.

2. Tổng quan về dự án

Dự án hướng tới việc xây dựng một hệ thống gồm:

- Cảm biến ánh sáng BH1750 đo cường độ chiếu sáng trong phòng học
- ESP32 đọc giá trị từ cảm biến, xử lý theo thuật toán điều khiển, điều khiển relay bật/tắt đèn.
- Kết nối Wi-Fi truyền dữ liệu thời gian thực lên nền tảng IoT (Web server).
- Giao diện người dùng trên điện thoại hoặc trình duyệt web cho phép theo dõi độ rọi, trạng thái đèn và điều khiển chiếu sáng từ xa.

Hệ thống hoạt động theo hai chế độ:

- Tự động: Bật/tắt đèn dựa trên độ sáng và tiêu chuẩn chiếu sáng.
- Thủ công: Người dùng chủ động điều khiển trên giao diện.

3. Mục đích dự án

Mục tiêu chính của hệ thống gồm:

- Xây dựng mô hình IoT có khả năng giám sát cường độ ánh sáng phòng học theo thời gian thực.
- Tự động điều khiển đèn sao cho độ sáng đáp ứng tiêu chuẩn chiếu sáng trong phòng học và tiết kiệm điện năng.
- Cung cấp giao diện trực quan giúp người dùng:
    - Theo dõi độ sáng, trạng thái đèn.
    - Chuyển chế độ điều khiển (auto/manual).
    - Điều khiển đèn từ xa.
- Đánh giá hiệu quả tiết kiệm năng lượng so với việc bật/tắt đèn thủ công.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu

- Các giải pháp IoT trong điều khiển chiếu sáng.
- Các linh kiện: cảm biến ánh sáng, ESP32, module relay, LED/đèn 220V.
- Giao thức truyền thông (Wi-Fi, HTTP/MQTT) và nền tảng IoT (Server web).

Phạm vi áp dụng

- Mô hình triển khai thử nghiệm trong phòng học hoặc mô hình phòng học thu nhỏ.
- Số lượng đèn điều khiển: 6 đèn (mô phỏng bằng đèn LED).
- Hệ thống ưu tiên chức năng giám sát và bật/tắt đèn, chưa đi sâu vào điều chỉnh độ sáng liên tục hay tích hợp nhiều cảm biến khác.
- Đối tượng sử dụng:
    - Giảng viên.
    - Nhân viên kĩ thuật.

5. Phương pháp nghiên cứu

- **Nghiên cứu lý thuyết**: Tìm hiểu về IoT, ESP32, cảm biến ánh sáng, relay, giao thức mạng, tiêu chuẩn chiếu sáng TCVN 7114-1:2008.
- **Khảo sát yêu cầu**: Phân tích nhu cầu thực tế trong phòng học, xác định các yêu cầu chức năng và phi chức năng.
- **Thiết kế hệ thống**: Thiết kế kiến trúc 3 lớp IoT, sơ đồ khối, sơ đồ kết nối phần cứng và kiến trúc phần mềm.
- **Thực nghiệm**: Lắp ráp mô hình, lập trình ESP32, cấu hình nền tảng IoT, đo và ghi nhận kết quả.

- **Đánh giá**: So sánh trước – sau khi áp dụng hệ thống dựa trên tiêu chí độ trễ, độ ổn định, mức tiết kiệm điện và sự hài lòng của người dùng.

6. Công nghệ và thiết bị cần thiết cho dự án

Phần cứng

- Vi điều khiển **ESP32** tích hợp Wi-Fi.
- Cảm biến ánh sáng **BH1750**.
- Module **relay 8 kênh** điều khiển tải 220V AC.
- Đèn LED 220V hoặc LED 5V mô phỏng đèn học.
- Nguồn DC 5V–12V cho mạch điều khiển, nguồn AC 220V cho đèn (nếu dùng đèn thật).
- Breadboard, dây nối, điện trở, mạch cách ly nếu cần.
- Bộ phát Wi-Fi (router) hoặc hotspot từ điện thoại.

Phần mềm

- **Arduino IDE**: Lập trình và nạp code cho ESP32.
- Thư viện cho ESP32, BH1750, HTTP/MQTT.
- **Blynk/Firebase/web server**: Nền tảng IoT dùng để trực quan hóa dữ liệu và điều khiển từ xa.

II. Nền tảng lý thuyết và công nghệ sử dụng

1. Module ESP32

1.1 Giới thiệu chung

ESP32 là một vi điều khiển (microcontroller) tích hợp Wi-Fi và Bluetooth do hãng **Espressif Systems** phát triển, thuộc họ chip IoT rất phổ biến hiện nay. ESP32 được thiết kế như một hệ thống trên chip (SoC – System on Chip), tích hợp CPU, bộ nhớ, các khối ngoại vi (peripheral) và module không dây ngay trên cùng một chip, giúp giảm kích thước, chi phí và độ phức tạp của mạch phần cứng.

Một số đặc điểm chính của ESP32:

- CPU: lõi kép 32-bit (Tensilica Xtensa LX6), xung nhịp tối đa 240 MHz.
- Bộ nhớ: SRAM tích hợp, Flash ngoài (thường 4 MB trên các module DevKit).
- Kết nối không dây:
    - Wi-Fi 2.4 GHz chuẩn 802.11 b/g/n.
    - Bluetooth/BLE (tùy phiên bản).
- Ngoại vi phong phú: nhiều chân GPIO, ADC, DAC, PWM, UART, SPI, I2C, I²S, Touch Sensor,…
- Hỗ trợ tốt cho các nền tảng phát triển: Arduino IDE, ESP-IDF, PlatformIO,…

Nhờ tích hợp sẵn Wi-Fi và tài nguyên đủ mạnh, ESP32 rất phù hợp cho các ứng dụng **IoT giám sát và điều khiển từ xa**, như: nhà thông minh, hệ thống chiếu sáng thông minh, giám sát môi trường, điều khiển thiết bị điện qua Internet,…

Trong đề tài của nhóm, ESP32 đóng vai trò **"bộ não" của hệ thống**:

- Đọc dữ liệu từ cảm biến ánh sáng.
- Xử lý theo thuật toán điều khiển (so sánh với ngưỡng, áp dụng hysteresis).
- Xuất tín hiệu điều khiển đến module relay để bật/tắt đèn.
- Kết nối Wi-Fi và gửi/nhận dữ liệu với nền tảng IoT hoặc server (Blynk/web,…).

1.2 Nguyên lý hoạt động

Về nguyên tắc, ESP32 hoạt động theo mô hình **nhúng (embedded)**:

1. **Khởi tạo (Initialization)**
   Khi cấp nguồn, ESP32 khởi động và chạy hàm setup() (nếu lập trình bằng Arduino IDE):
   - Cấu hình các chân GPIO (chân đọc cảm biến, chân điều khiển relay).
   - Khởi tạo giao tiếp (I2C).
   - Kết nối Wi-Fi tới mạng nội bộ/trường.
   - Khởi tạo kết nối với nền tảng IoT (Blynk/server).
   - Nạp hoặc đọc các tham số cấu hình ban đầu (ngưỡng sáng, chế độ Auto/Manual) từ bộ nhớ.
2. **Vòng lặp chính (Main loop)**
   Sau khi khởi tạo xong, ESP32 lặp đi lặp lại khối lệnh trong hàm loop():
   - **Đọc dữ liệu cảm biến ánh sáng** (LDR qua ADC hoặc BH1750 qua I2C).
   - **Xử lý dữ liệu**:
     - Chuyển giá trị đọc được về đơn vị lux (nếu cần).
     - So sánh với ngưỡng **threshold low** và **threshold high** để quyết định bật/tắt đèn (hysteresis).
   - **Điều khiển relay**:
     - Nếu cần bật đèn → xuất mức tín hiệu phù hợp lên chân điều khiển relay.
     - Nếu cần tắt đèn → đưa chân điều khiển relay về trạng thái ngược lại.
   - **Giao tiếp với nền tảng IoT**:
     - Gửi dữ liệu (giá trị ánh sáng, trạng thái đèn, chế độ) lên server/app.
     - Nhận lệnh điều khiển từ xa (bật/tắt đèn thủ công, đổi chế độ Auto/Manual, thay đổi ngưỡng).
   - **Quản lý kết nối**:
     - Kiểm tra trạng thái Wi-Fi, nếu mất kết nối thì tự động thử kết nối lại.
3. **Tương tác với môi trường và phản hồi**

- Khi ánh sáng môi trường thay đổi (trời tối đi, rèm đóng/mở,…), giá trị đo được từ cảm biến thay đổi → ESP32 phát hiện → điều chỉnh trạng thái đèn thông qua relay.
- Khi người dùng thao tác trên ứng dụng (bấm nút bật/tắt, đổi chế độ, chỉnh ngưỡng), server chuyển lệnh tới ESP32 → ESP32 cập nhật biến trạng thái và thực hiện điều khiển tương ứng.

Như vậy, trong hệ thống ESP32 vừa là **bộ xử lý trung tâm**, vừa là **thiết bị kết nối mạng**, đóng vai trò cầu nối giữa:

- **Thế giới vật lý**: cảm biến ánh sáng, đèn, relay.
- **Thế giới số**: ứng dụng IoT, server, người dùng điều khiển từ xa.

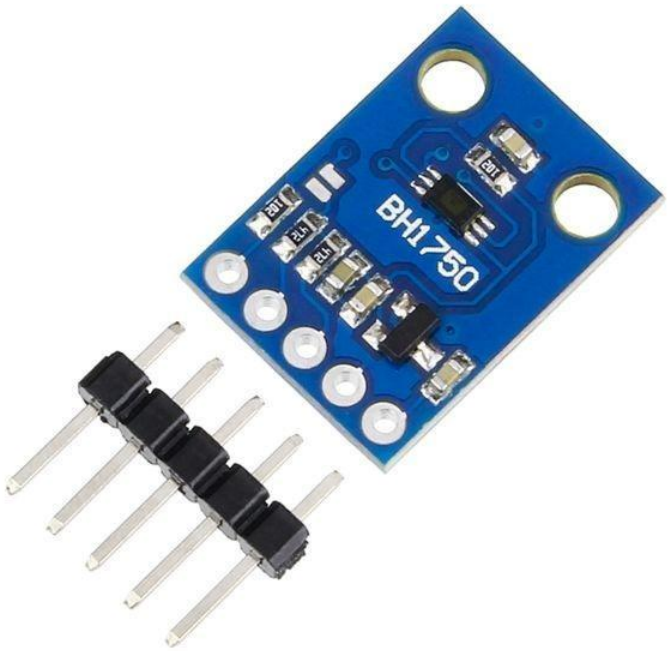
2. Cảm biến ánh sáng BH1750

2.1 Giới thiệu chung

BH1750 là một **cảm biến đo cường độ ánh sáng môi trường** (ambient light sensor) do hãng Rohm (Nhật Bản) sản xuất. Cảm biến này cho ra kết quả đo trực tiếp theo đơn vị **độ rọi – lux**, rất thuận tiện cho các ứng dụng cần so sánh với **tiêu chuẩn chiếu sáng** trong thực tế (phòng học, văn phòng, nhà xưởng,…).

Một số đặc điểm chính của BH1750:

- Dải đo rộng, thường từ khoảng **1 lux đến 65.535 lux**.
- Giao tiếp **kỹ thuật số qua bus I2C**, chỉ cần 2 dây dữ liệu (SDA, SCL).
- Tích hợp sẵn **ADC 16 bit**, cho kết quả đo ổn định, độ phân giải cao.
- Có nhiều **chế độ đo** (ví dụ: High Resolution, Low Resolution) với thời gian chuyển đổi khác nhau.
- Điện áp hoạt động phổ biến từ 3.0V–5.0V, phù hợp để kết nối trực tiếp với ESP32.



Trong đề tài, BH1750 được dùng làm cảm biến chính để:

- Đo cường độ ánh sáng trong phòng học theo **đơn vị lux**.
- Cung cấp dữ liệu chính xác cho ESP32 nhằm:
    - Tự động **bật/tắt đèn** theo ngưỡng ánh sáng cấu hình (liên quan đến chuẩn 300–500 lux của phòng học).
    - Gửi số liệu lux lên ứng dụng IoT/web để người dùng **giám sát và đánh giá chất lượng chiếu sáng**.

Việc sử dụng BH1750 giúp hệ thống:

- Dễ hiệu chỉnh ngưỡng theo tiêu chuẩn (300, 350, 400 lux…).
- Đo được thay đổi ánh sáng nhỏ, phản hồi tốt với biến thiên ánh sáng tự nhiên (trời nắng, trời âm u, bật/tắt thêm đèn ngoài,…).

2.2 Nguyên lý hoạt động

BH1750 hoạt động theo nguyên tắc:

1. **Cảm biến quang – chuyển đổi ánh sáng thành dòng điện**
   Bên trong BH1750 có phần tử cảm biến quang (photodiode). Khi ánh sáng chiếu vào, photodiode tạo ra dòng điện tỷ lệ với cường độ ánh sáng.

2. **Chuyển đổi tương tự – số (ADC) tích hợp**
   Dòng điện từ photodiode được đưa vào bộ ADC (bộ chuyển đổi tương tự – số) tích hợp bên trong BH1750. Kết quả là BH1750 tạo ra một **giá trị số tương ứng với độ rọi ánh sáng**.
3. **Giao tiếp với vi điều khiển qua I2C**
   ESP32 giao tiếp với BH1750 qua bus **I2C** gồm 2 đường:
   - **SDA (Serial Data)**: đường dữ liệu.
   - **SCL (Serial Clock)**: đường xung clock.
4. Quy trình làm việc cơ bản:
   - ESP32 gửi lệnh cấu hình tới BH1750 (chọn chế độ đo: high resolution, low resolution,…).
   - BH1750 tiến hành đo ánh sáng trong khoảng thời gian đo (ví dụ ~120ms). Sau khi đo xong, ESP32 đọc về **2 byte dữ liệu** từ BH1750 qua I2C.
   - Dữ liệu này được chuyển đổi theo hệ số trong datasheet để ra **giá trị lux**.
5. **Tích hợp vào thuật toán điều khiển đèn**
   Sau khi ESP32 đọc được giá trị lux, hệ thống tiến hành:
   - So sánh giá trị lux với các **ngưỡng điều khiển**:
     - Nếu lux < ngưỡng dưới (ví dụ *L_low* = 280 lux) → coi là **thiếu sáng → bật đèn**.
     - Nếu lux > ngưỡng trên (ví dụ *L_high* = 320 lux) → coi là **đủ sáng → tắt đèn**.
   - Khoảng giữa hai ngưỡng (*L_low*, *L_high*) tạo thành **vùng hysteresis**, giúp tránh việc bật/tắt liên tục khi độ sáng dao động nhẹ quanh ngưỡng.
6. Đồng thời, ESP32 cũng:
   - Gửi giá trị lux hiện tại và trạng thái đèn lên **nền tảng IoT** (Blynk/web server).
   - Nhận lệnh điều khiển từ xa (bật/tắt, đổi chế độ Auto/Manual, chỉnh lại ngưỡng).
7. **Vai trò của BH1750 trong toàn hệ thống**
   Tóm lại, BH1750 giữ vai trò:
   - Là "**mắt cảm nhận ánh sáng**" của hệ thống trong phòng học.
   - Cung cấp giá trị **lux chính xác**, làm cơ sở để so sánh với các chuẩn chiếu sáng.
   - Giúp hệ thống:
     - **Tự động điều chỉnh chiếu sáng** để đảm bảo điều kiện học tập.
     - **Tối ưu tiêu thụ điện** bằng cách tắt đèn khi ánh sáng tự nhiên đã đủ.

3. Relay điều khiển đèn

3.1 Giới thiệu chung

Relay là một **công tắc điện tử điều khiển bằng tín hiệu điện**. Nó cho phép dùng một tín hiệu điều khiển điện áp thấp, dòng nhỏ (từ vi điều khiển như ESP32) để **đóng/cắt một mạch điện áp cao, dòng lớn** (như đèn 220V AC trong phòng học).

Trên thực tế, trong các hệ thống IoT điều khiển thiết bị điện gia dụng, **module relay** là linh kiện rất phổ biến vì:

- Giúp **cách ly an toàn** giữa mạch điều khiển (3.3V/5V) và mạch tải 220V.
- Có thể đóng/cắt tải công suất tương đối lớn (đèn, quạt, ổ cắm,…).
- Dễ sử dụng, thường chỉ cần vài chân kết nối (VCC, GND, IN, COM, NO, NC).

Cấu trúc cơ bản của một module relay:

- **Phần relay cơ (mechanical relay):**
  - **Cuộn dây (coil):** khi có dòng điện chạy qua sẽ tạo từ trường, hút/thả tiếp điểm.
  - **Tiếp điểm:**
    - **COM (Common):** chân chung.
    - **NO (Normally Open):** tiếp điểm thường hở, chỉ đóng lại khi relay được kích.
    - **NC (Normally Closed):** tiếp điểm thường đóng, mở ra khi relay được kích.
- **Phần mạch điều khiển trên module:**
  - **Transistor/driver:** khuếch đại dòng, giúp tín hiệu từ vi điều khiển đủ sức kích cuộn dây.
  - **Optocoupler (tùy loại):** cách ly quang để tăng độ an toàn.
  - **Diode bảo vệ:** chống xung ngược khi cuộn dây ngắt dòng.
  - Đèn LED báo trạng thái, trở hạn dòng,…

Nhờ module relay, ESP32 chỉ cần xuất tín hiệu **3.3V mức HIGH/LOW** là đã có thể điều khiển bật/tắt một thiết bị chạy **điện lưới 220V AC** mà vẫn đảm bảo an toàn cho vi điều khiển.

3.2 Nguyên lý hoạt động

Trong hệ thống **giám sát ánh sáng và điều khiển đèn tự động trong phòng học**, relay là "cầu nối" giữa:

- **Khối điều khiển logic:** ESP32 (nguồn 5V).
- **Khối tải công suất:** đèn 220V AC (hoặc LED mô phỏng).

Có thể mô tả nguyên lý hoạt động qua 3 lớp:

a) Kết nối phần cứng

Một kênh relay điều khiển đèn thường được đấu như sau:

- Phía **điều khiển (low voltage):**
  - Chân **IN** của relay nối với một chân **GPIO** của ESP32 (ví dụ GPIO23).
  - Chân **VCC** của module relay nối với nguồn 5V.
  - Chân **GND** nối chung mass với ESP32.
- Phía **tải (high voltage – đèn):**
  - Dây "nóng" (L) của nguồn 220V được **cắt** và nối qua hai chân **COM** và **NO** của relay:
    - Một đầu dây L nối vào **COM**.
    - Đầu còn lại từ **NO** đi tới đèn.
  - Dây "nguội" (N) nối trực tiếp tới đèn.

Khi relay chưa kích (IN ở trạng thái không kích), tiếp điểm NO–COM hở → mạch đèn hở → **đèn tắt**.
Khi relay được kích, NO–COM đóng → mạch đèn kín → **đèn sáng**.

b) Hoạt động logic điều khiển từ ESP32

1. **ESP32 đọc độ sáng từ BH1750**
   - BH1750 đo độ rọi trong phòng học và gửi giá trị lux về cho ESP32 qua I2C.
   - Ví dụ: lux = 150, 250, 350,…
2. **ESP32 xử lý và quyết định bật/tắt đèn**
   - Hệ thống sử dụng hai ngưỡng để tạo hysteresis, ví dụ:
     - L_{low}=300 lux.
     - L_{high}=500 lux.
   - Thuật toán:
     - Nếu **lux** < L_{low} → môi trường thiếu sáng → quyết định **bật đèn**.
     - Nếu **lux** > L_{high} → môi trường đủ sáng → quyết định **tắt đèn**.
     - Nếu lux nằm giữa hai ngưỡng → giữ nguyên trạng thái hiện tại để tránh bật/tắt liên tục.
3. **ESP32 xuất tín hiệu điều khiển tới relay**

- Nếu cần **bật đèn**:
  - ESP32 đưa chân GPIO (kết nối với IN của relay) lên mức **HIGH** (hoặc LOW, tùy loại relay kích mức cao/thấp).
  - Dòng điều khiển chạy qua cuộn dây relay → relay hút → NO–COM đóng → đèn sáng.
- Nếu cần **tắt đèn**:
  - ESP32 đưa chân GPIO về mức ngược lại.
  - Cuộn dây relay ngừng được cấp, tiếp điểm trở lại trạng thái ban đầu → NO–COM hở → đèn tắt.

4. **Phản hồi lên hệ thống IoT**
   - Sau khi relay đổi trạng thái, ESP32 cập nhật **trạng thái đèn (ON/OFF)** lên giao diện điều khiển.
   - Người dùng có thể xem trên web đèn đang bật hay tắt, đồng thời gửi lệnh **điều khiển thủ công**:
     - Nếu ở **chế độ Manual (thủ công)**: lệnh từ web được ưu tiên, ESP32 điều khiển relay theo yêu cầu người dùng.
     - Nếu ở **chế độ Auto (tự động)**: relay chủ yếu được điều khiển bởi thuật toán dựa trên giá trị lux.

c) Vai trò của relay trong hệ thống chiếu sáng thông minh

Tóm lại, trong đề tài:

- Relay là phần tử **chấp hành chính** để thực hiện quyết định bật/tắt đèn của ESP32.
- Giúp **ngăn cách an toàn** giữa mạch điều khiển logic (ESP32 + cảm biến) và mạch tải 220V.
- Cho phép hệ thống:
  - Tự động điều chỉnh chiếu sáng theo độ sáng đo được.
  - Vẫn hỗ trợ người dùng **can thiệp thủ công** (bật/tắt đèn từ giao diện điều khiển) mà không cần chạm vào công tắc cơ.

4. Giao thức I2C

4.1 Giới thiệu chung

I2C (Inter-Integrated Circuit) là một **giao thức truyền thông nối tiếp đồng bộ** được sử dụng rất rộng rãi để kết nối các linh kiện trên mạch in, đặc biệt là giữa vi điều khiển và các cảm biến, IC ngoại vi.

Đặc điểm nổi bật của I2C:

- **Chỉ dùng 2 đường tín hiệu chính:**
  - **SCL (Serial Clock):** đường xung clock do master tạo ra.
  - **SDA (Serial Data):** đường dữ liệu hai chiều.

- Hỗ trợ kết nối **nhiều thiết bị** trên cùng một bus (multi-slave), mỗi thiết bị có một **địa chỉ (address)** riêng.
- Theo mô hình **master – slave**:
    - Thường **ESP32 là master**,
    - Các cảm biến như **BH1750 là slave**.
- Dữ liệu truyền theo dạng **khung (frame)**, gồm: bit START, địa chỉ slave, bit R/W, các byte dữ liệu, bit ACK/NACK, bit STOP.

Nhờ cấu trúc đơn giản (2 dây), hỗ trợ nhiều thiết bị, lại tích hợp sẵn trong hầu hết vi điều khiển, I2C rất phù hợp để:

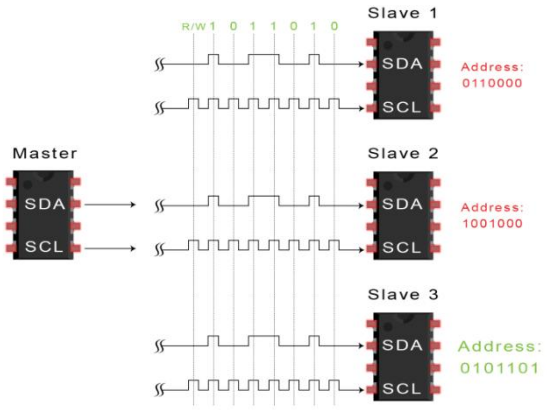
- Kết nối ESP32 với **cảm biến BH1750**.
  Mở rộng thêm các cảm biến khác (nhiệt độ, độ ẩm, OLED hiển thị…) trên cùng một bus.

Trong đề tài, **BH1750 sử dụng giao thức I2C** để gửi dữ liệu đo ánh sáng (lux) về cho ESP32.

4.2 Nguyên lý hoạt động

Trong hệ thống "Giám sát ánh sáng và điều khiển đèn tự động", I2C chính là **kênh giao tiếp** giữa:

- **Master:** ESP32.
- **Slave:** cảm biến ánh sáng BH1750.



Có thể mô tả nguyên lý hoạt động theo các bước sau:

a) Cấu trúc bus I2C

- ESP32 và BH1750 cùng nối chung vào **2 đường**:
    - SDA ↔ SDA.
    - SCL ↔ SCL.

- Hai đường này được **kéo lên VCC qua điện trở pull-up** (thường 4.7kΩ–10kΩ).
- Khi bus rỗi, SDA và SCL đều ở mức HIGH.

Mỗi slave trên bus (ở đây là BH1750) có **địa chỉ riêng** (ví dụ 0x23 hoặc 0x5C tùy cách nối chân ADDR), giúp ESP32 biết mình đang giao tiếp với thiết bị nào.

b) Quy trình truyền dữ liệu cơ bản (chuẩn I2C)

Một giao dịch I2C cơ bản gồm:

1. **Điều kiện START**
   - Master (ESP32) kéo SDA từ HIGH xuống LOW trong khi SCL vẫn ở HIGH → báo hiệu **bắt đầu một khung truyền**.
2. **Gửi địa chỉ slave + bit R/W**
   - ESP32 gửi **7 bit địa chỉ** của BH1750 + 1 bit R/W (0 = ghi, 1 = đọc).
   - Sau 8 bit này, BH1750 sẽ phản hồi **ACK (kéo SDA xuống LOW)** nếu nhận địa chỉ đúng.
3. **Ghi lệnh cấu hình (nếu cần)**
   - Khi ESP32 muốn yêu cầu BH1750 đo, nó gửi **một byte lệnh** (ví dụ chọn chế độ đo High Resolution).
     Sau mỗi byte, BH1750 đều gửi lại 1 bit ACK để báo nhận thành công.
4. **Đợi BH1750 đo ánh sáng**
   - BH1750 thực hiện phép đo trong một khoảng thời gian (ví dụ ~120ms).
   - Trong thời gian này, ESP32 có thể chờ hoặc làm việc khác.
5. **Đọc dữ liệu từ BH1750**
   - ESP32 phát lại START, gửi địa chỉ BH1750 + bit R/W = 1 (chế độ đọc).
   - BH1750 gửi lại **2 byte dữ liệu** đo được (MSB, LSB) lên SDA, xung SCL do ESP32 điều khiển.
   - Sau mỗi byte, ESP32 gửi ACK (trừ byte cuối có thể gửi NACK để kết thúc đọc).
6. **Điều kiện STOP**
   - Master kéo SDA từ LOW lên HIGH trong khi SCL ở HIGH → báo hiệu **kết thúc khung truyền**.

ESP32 sau đó dùng công thức (theo datasheet) để chuyển giá trị 2 byte nhận được thành **giá trị lux**.

c) Vai trò của I2C trong hệ thống ESP32 – BH1750 – Relay

Toàn bộ chuỗi hoạt động chính liên quan đến I2C trong đề tài có thể tóm tắt:

1. ESP32 cấu hình BH1750 qua I2C (chọn chế độ đo).
2. BH1750 đo ánh sáng và lưu kết quả dạng số bên trong.
3. ESP32 đọc 2 byte dữ liệu đo lux từ BH1750 qua bus I2C.

4. Từ giá trị lux:
    - So sánh với ngưỡng L_{low}, L_{high}.
    - Quyết định bật/tắt relay điều khiển đèn.
5. ESP32 đồng thời gửi giá trị lux và trạng thái đèn lên **nền tảng IoT** để hiển thị.

Như vậy, **I2C là mắt xích bắt buộc** để:

- BH1750 truyền thông tin ánh sáng về cho ESP32.
- ESP32 có dữ liệu đầu vào chính xác để điều khiển đèn theo **điều kiện chiếu sáng thực tế** trong phòng học.

5. Giao thức wifi

5.1. Giới thiệu chung

Wi-Fi (Wireless Fidelity) là một công nghệ mạng không dây cho phép các thiết bị kết nối Internet hoặc mạng nội bộ thông qua sóng radio trong phạm vi nhất định. Wi-Fi dựa trên chuẩn IEEE 802.11, cung cấp kết nối tốc độ cao, linh hoạt và tiện lợi, rất phổ biến trong các ứng dụng IoT và hệ thống nhúng.

Wi-Fi giúp các thiết bị có thể giao tiếp với nhau và với Internet mà không cần dây cáp, tạo điều kiện thuận lợi cho việc điều khiển từ xa và thu thập dữ liệu.

5.2. Nguyên lý hoạt động

Trong dự án ESP32 sử dụng Wi-Fi để kết nối với mạng nội bộ tại phòng học. Khi được cấu hình với tên mạng (SSID) và mật khẩu Wi-Fi, ESP32 sẽ đăng nhập vào mạng này và nhận địa chỉ IP nội bộ.

Thông qua kết nối Wi-Fi này, các thiết bị khác như máy tính hoặc điện thoại trong cùng mạng có thể truy cập vào ESP32 bằng địa chỉ IP được cấp. ESP32 vận hành một web server nhỏ, cho phép người dùng điều khiển các relay (đèn) và đọc dữ liệu cảm biến ánh sáng từ xa qua giao diện web hoặc ứng dụng.

Việc sử dụng Wi-Fi giúp dự án có khả năng vận hành không dây, không phụ thuộc vào cáp mạng, tăng tính linh hoạt và dễ dàng mở rộng.

6. Giao thức HTTP

6.1. Giới thiệu chung

- HTTP là một giao thức truyền tải dữ liệu ứng dụng dựa trên mô hình client-server. Nó là nền tảng cho các giao tiếp web, cho phép các trình duyệt web gửi yêu cầu (request) đến server và nhận dữ liệu phản hồi (response) như trang HTML, JSON, hình ảnh, v.v.
- HTTP hoạt động trên giao thức TCP/IP, sử dụng các phương thức như GET, POST, PUT, DELETE để thực hiện các thao tác khác nhau trên tài nguyên mạng.

6.2. Nguyên lý hoạt động

Trong dự án này, ESP32 đóng vai trò là một HTTP server. Nó lắng nghe các yêu cầu HTTP từ client (trình duyệt hoặc ứng dụng web) gửi đến địa chỉ IP của ESP32.

Các API HTTP được xây dựng gồm:

- `/status`: Client gửi yêu cầu GET để lấy trạng thái hiện tại của hệ thống, bao gồm mức ánh sáng (lux), trạng thái các relay (đèn), và chế độ hoạt động (tự động/manual). ESP32 trả về dữ liệu dạng JSON.
- `/toggle`: Client gửi yêu cầu GET kèm tham số `id` (mã đèn) và `state` (bật/tắt) để điều khiển trạng thái relay tương ứng. ESP32 nhận lệnh và cập nhật trạng thái đèn.
- `/setMode`: Client gửi yêu cầu GET để chuyển đổi chế độ hoạt động giữa tự động và thủ công.

Thông qua giao thức HTTP, người dùng có thể giám sát và điều khiển hệ thống một cách dễ dàng từ bất kỳ thiết bị nào trong mạng Wi-Fi, chỉ cần trình duyệt web hoặc ứng dụng có khả năng gửi HTTP request.

7. Phần mềm và nền tảng IoT

7.1. Phần mềm lập trình: Arduino IDE

- **Arduino IDE (Integrated Development Environment)** là phần mềm mã nguồn mở, đa nền tảng, giúp viết mã và nạp chương trình cho các vi điều khiển một cách dễ dàng và nhanh chóng.
- Ngôn ngữ lập trình dựa trên C/C++, được đơn giản hóa với framework Wiring cung cấp các hàm tiện ích như `digitalWrite()`, `analogRead()`.

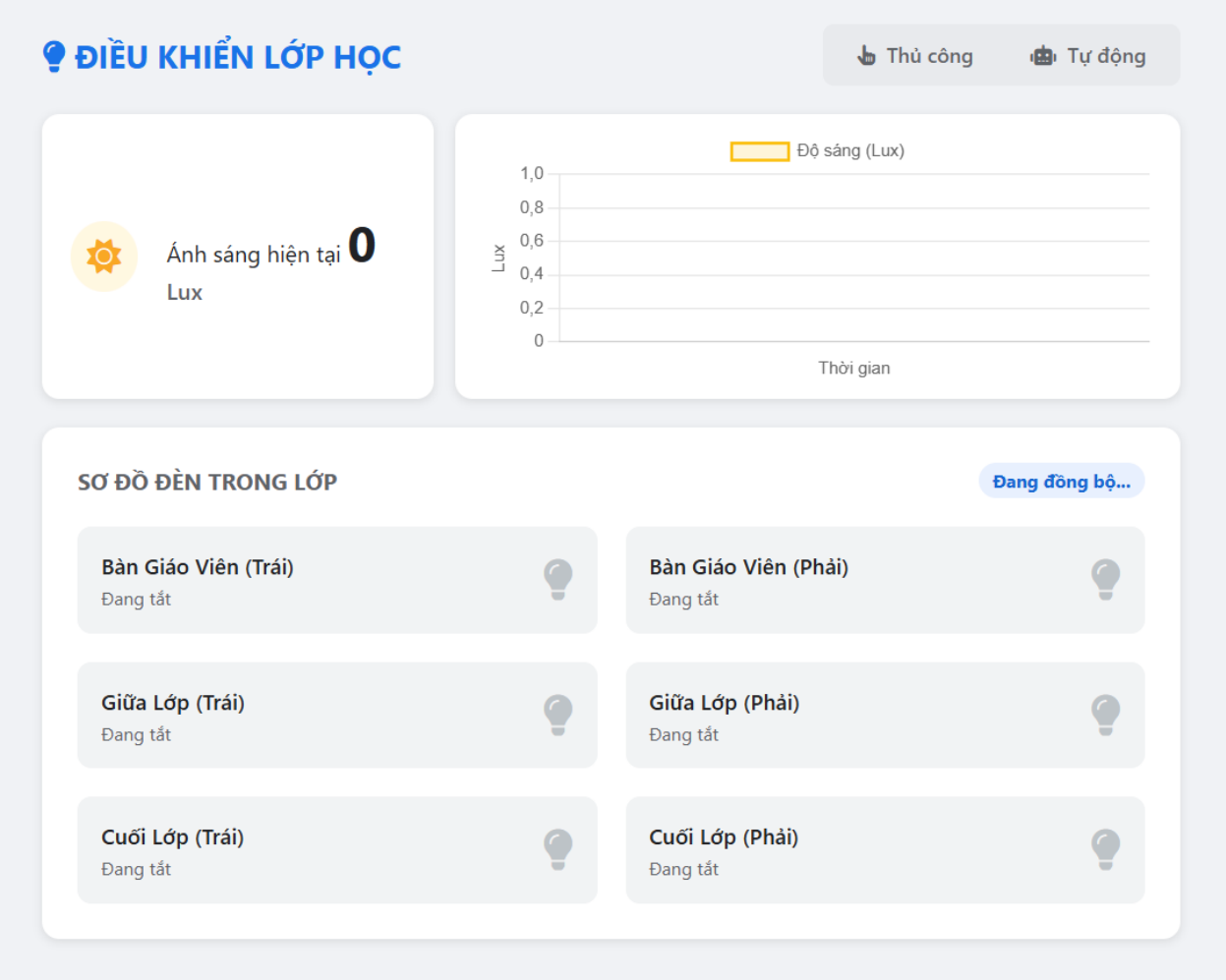
- Thông qua **Boards Manager**, Arduino IDE hỗ trợ lập trình cho ESP32, giúp lập trình viên dễ dàng triển khai ứng dụng trên nền tảng này bằng cú pháp quen thuộc.

7.2. Thư viện sử dụng

- Dự án sử dụng thư viện **BH1750** để đọc dữ liệu cảm biến ánh sáng qua giao tiếp I2C.
- Sử dụng thư viện **WiFi.h** và **WebServer.h** để thiết lập kết nối Wi-Fi và tạo web server trên ESP32, phục vụ việc truyền nhận dữ liệu và điều khiển từ xa qua mạng LAN.

7.3. Nền tảng IoT tự phát triển

- Hệ thống sử dụng **Web Server tích hợp trên ESP32** để cung cấp giao diện điều khiển và API cho việc điều khiển trạng thái relay và đọc giá trị cảm biến ánh sáng (lux).
- Dữ liệu cảm biến được gửi qua giao thức HTTP dưới dạng JSON để giao diện web có thể lấy và cập nhật trạng thái theo thời gian thực.
- Giao diện web trực quan được xây dựng để người dùng có thể theo dõi trạng thái đèn, điều chỉnh thủ công hoặc để hệ thống tự động điều khiển dựa trên giá trị cảm biến.



5.4. Luồng hoạt động

- ESP32 kết nối vào mạng Wi-Fi để tạo Web Server nội bộ, cho phép thiết bị khác trong mạng truy cập và tương tác.
- Sensor BH1750 đo ánh sáng môi trường và truyền dữ liệu về ESP32.
- Người dùng truy cập giao diện web qua địa chỉ IP ESP32 để xem thông số ánh sáng, biểu đồ thời gian thực và trạng thái các relay (đèn).
- Người dùng có thể điều khiển bật/tắt từng đèn trong chế độ thủ công hoặc chọn chế độ tự động để hệ thống tự bật/tắt đèn dựa trên ngưỡng ánh sáng đã định.

8. Tiêu chuẩn chiếu sáng trong phòng học

Theo tiêu chuẩn **TCVN 7114-1:2008 (hoặc EN 12464-1)**:

- Mức độ chiếu sáng yêu cầu cho phòng học: **300 – 500 lux**.
- Độ đồng đều ≥ 0.7, đảm bảo ánh sáng phân bố đều và không gây chói.
- Hệ thống sẽ điều chỉnh bật/tắt đèn để duy trì độ sáng trong khoảng tiêu chuẩn này.

9. Lý thuyết điều khiển tự động (Ngưỡng – Hysteresis)

Hệ thống hoạt động dựa trên **nguyên lý điều khiển phản hồi (feedback control):**

- Đầu vào: độ sáng môi trường (lux) từ cảm biến
- Bộ điều khiển: thuật toán trong ESP32.
- Đầu ra: trạng thái đèn(bật/tắt).
- Phản hồi: độ sáng đèn sau khi bật tắt.

III. Phân tích yêu cầu – Các chức năng triển khai

Phân tích yêu cầu

1. Yêu cầu chức năng

1.1. Thu thập dữ liệu

- ESP32 đọc liên tục giá trị ánh sáng từ **cảm biến BH1750**.
- Dữ liệu thu thập là **độ rọi (lux)**, phản ánh cường độ sáng môi trường.

1.2. Xử lý và ra quyết định

- So sánh giá trị đo được với **ngưỡng chiếu sáng tiêu chuẩn** (ví dụ 300 lux theo TCVN 7114-1:2008).
- Áp dụng **thuật toán điều khiển tự động (hysteresis)** để tránh bật/tắt đèn liên tục khi ánh sáng dao động quanh ngưỡng.

Ví dụ: nếu ánh sáng < 300 lux → bật đèn; nếu > 500 lux → tắt đèn.

1.3. Điều khiển thiết bị (Relay)

- Gửi tín hiệu điều khiển **bật/tắt relay** tương ứng với đèn 220V.
- Đảm bảo **an toàn cách ly điện áp cao – thấp**.

1.4. Truyền và lưu trữ dữ liệu

- Gửi dữ liệu cảm biến (lux, trạng thái đèn) lên **nền tảng trang web riêng** qua Wi-Fi.

1.5. Hiển thị và điều khiển từ xa

- Người dùng xem dữ liệu ánh sáng và trạng thái đèn trên **trang web riêng**
- Có thể **điều khiển bật/tắt đèn thủ công** qua trang web (chế độ override).
- Giao diện hiển thị gồm:
    - Widget hiển thị độ rọi (lux)
    - Nút điều khiển đèn (manual)

1.6. Tự động – Thủ công

- Hai chế độ hoạt động:
    - **Tự động:** điều khiển theo cảm biến.
    - **Thủ công:** người dùng điều khiển trực tiếp qua trang web riêng.

2. Yêu cầu phi chức năng

2.1. Hiệu năng

- **Độ trễ (Latency):** ≤ 1 giây giữa hành động người dùng trên app và phản hồi của đèn.
- **Tần suất đọc cảm biến:** mỗi 1–2 giây/lần.
- **Tốc độ phản hồi:** bật/tắt relay tức thời (<100 ms).
- **Chịu tải:** hệ thống có thể mở rộng để giám sát **nhiều cảm biến và đèn** trong cùng một mạng.

2.2. Bảo mật

- Hệ thống hoạt động trong **mạng LAN nội bộ**, chỉ các thiết bị cùng mạng mới có thể truy cập Web Server của ESP32.
- ESP32 chỉ xử lý các yêu cầu hợp lệ đúng theo API /status, /toggle, /setMode để ngăn truy cập trái phép.
- Có thể bổ sung **Basic Authentication** (tài khoản/mật khẩu) để hạn chế quyền truy cập và phân quyền giữa người dùng và quản trị viên.
- Không mở port ra Internet nên tránh được các nguy cơ tấn công từ bên ngoài.
- Trong các phiên bản nâng cấp, hệ thống có thể sử dụng HTTPS/TLS hoặc token xác thực nếu mở rộng ra môi trường cloud.

2.3. Độ tin cậy

- Hệ thống tự khởi động lại khi mất kết nối hoặc mất điện (auto reconnect).
- ESP32 có thể lưu **giá trị ngưỡng và trạng thái đèn cuối cùng** trong bộ nhớ EEPROM.
- Cảm biến và relay được **bảo vệ điện áp** chống nhiễu và quá tải.

2.4. Mở rộng

- Dễ dàng **thêm nhiều cảm biến ánh sáng hoặc relay** khác (theo phòng, khu vực).
- Có thể **tích hợp nền tảng khác** như Firebase hoặc MQTT broker.
- Cho phép **nâng cấp firmware OTA (Over-the-Air)** qua Wi-Fi.

2.5. Chi phí và năng lượng

- Tối ưu phần cứng, chỉ dùng ESP32, cảm biến và relay phổ thông.
- **Chi phí thấp (~200.000–300.000 VNĐ/bộ)**.
- **Tiêu thụ năng lượng thấp**, phù hợp hoạt động 24/7.
- Dung lượng băng thông nhỏ (dữ liệu vài trăm byte/s).

3. Ràng buộc kỹ thuật và chức năng

3.1. Ràng buộc kỹ thuật và môi trường

Hệ thống IoT giám sát ánh sáng và điều khiển đèn tự động trong phòng học được thiết kế để hoạt động ổn định trong môi trường thực tế, tuy nhiên vẫn chịu ảnh hưởng bởi một số yếu tố sau:

- **Nhiệt độ:**
  Các linh kiện như vi điều khiển ESP32, cảm biến ánh sáng BH1750 có dải hoạt động tốt nhất trong khoảng **0°C đến 50°C**. Khi nhiệt độ vượt ngưỡng, độ chính xác đo sáng có thể giảm, và relay hoạt động không ổn định.
- **Độ ẩm:**
  Môi trường có **độ ẩm cao (>80%)** có thể gây **oxy hóa chân linh kiện** hoặc **chập mạch**, đặc biệt nếu hệ thống đặt gần cửa sổ hoặc khu vực có hơi nước.
- **Nhiễu sóng:**
  Hệ thống sử dụng WiFi 2.4GHz nên có thể bị ảnh hưởng bởi các thiết bị phát sóng cùng tần số (modem WiFi, camera IP, thiết bị Bluetooth). Nhiễu sóng mạnh có thể làm **mất kết nối tạm thời** hoặc **truyền dữ liệu chậm**.
- **Nguồn cấp:**
  Vi điều khiển và cảm biến cần nguồn ổn định **5V DC (hoặc 3.3V)**. Nếu sử dụng adapter hoặc cổng USB kém chất lượng, hệ thống có thể **tự khởi động lại hoặc relay bật/tắt sai**.
  Do đó, cần dùng nguồn có **dòng ra tối thiểu 2A** để đảm bảo hoạt động ổn định.

3.2. Ràng buộc pháp lý

Khi triển khai hệ thống IoT, cần tuân thủ các quy định và chuẩn kỹ thuật về truyền thông không dây và bảo mật dữ liệu:

- **Quy định tần số vô tuyến:**
  Module ESP32 sử dụng sóng **WiFi 2.4GHz**, thuộc dải tần ISM được **Bộ Thông tin và Truyền thông cho phép sử dụng miễn phí**, nhưng phải đảm bảo **công suất phát sóng dưới giới hạn quy định** để không gây nhiễu cho các thiết bị khác.

3.3. Tài nguyên và thiết bị

Do đây là hệ thống mô hình IoT nhỏ, các linh kiện sử dụng đều có **tài nguyên phần cứng hạn chế**, dẫn đến một số ràng buộc như sau:

- **Bộ nhớ và CPU:**
  Vi điều khiển ESP32 có **RAM khoảng 520KB** và **Flash 4MB**, đủ khả năng chạy các chương trình điều khiển phức tạp hơn, bao gồm xử lý dữ liệu cảm biến, kết nối Wi-Fi, và truyền dữ liệu thời gian thực.
  Tuy nhiên, **dung lượng bộ nhớ vẫn có giới hạn**, nên hệ thống **không phù hợp cho việc lưu trữ dữ liệu lớn hoặc xử lý học máy trực tiếp trên chip**; thay vào đó, dữ liệu nên được gửi lên **nền tảng Cloud** để lưu trữ và phân tích.
- **Dung lượng pin hoặc nguồn điện:**
  Nếu hệ thống chạy bằng pin, thời gian hoạt động bị giới hạn do **module WiFi tiêu**

thụ điện năng cao khi truyền dữ liệu.
Do đó, giải pháp tối ưu là **dùng nguồn adapter cắm trực tiếp** để đảm bảo hoạt động liên tục.

- **Giới hạn mở rộng thiết bị:**
  Hệ thống cơ bản chỉ hỗ trợ **1 cảm biến ánh sáng và 1 relay**.
  Nếu muốn mở rộng cho nhiều phòng học, cần **nâng cấp lên vi điều khiển ESP32** (nhiều chân I/O hơn, RAM lớn hơn) hoặc tích hợp **hệ thống mạng mesh WiFi**.

3.4. Ràng buộc chức năng

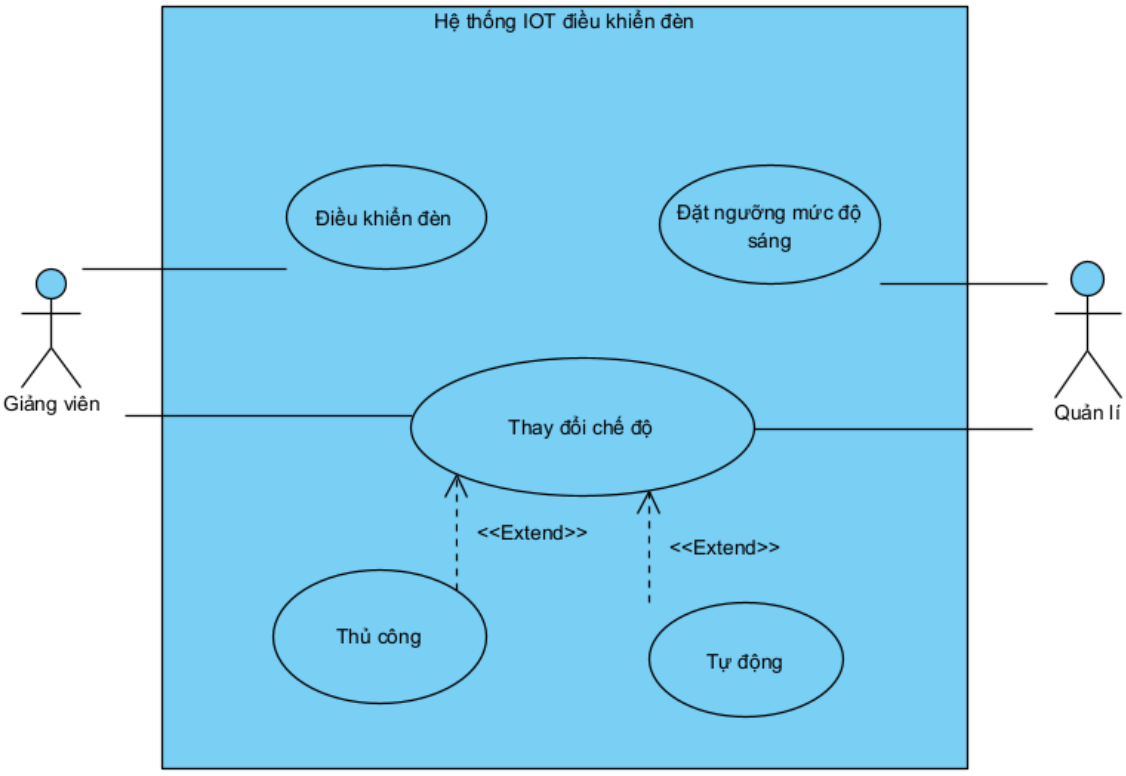
Hệ thống được thiết kế với các chức năng chính là giám sát ánh sáng và điều khiển đèn tự động. Tuy nhiên, phạm vi chức năng vẫn bị giới hạn trong một số khía cạnh:

STT	Chức năng	Giới hạn / Mô tả
1	**Giám sát ánh sáng**	Độ sáng đo được chỉ phản ánh khu vực đặt cảm biến, không đại diện toàn phòng.
2	**Điều khiển tự động**	Bật/tắt đèn dựa theo ngưỡng ánh sáng cố định, chưa hỗ trợ điều chỉnh thông minh hoặc học thói quen.
3	**Điều khiển từ xa**	Chỉ hoạt động khi có kết nối WiFi và Internet ổn định. Mất mạng thì chỉ chạy chế độ tự động cục bộ.
4	**Lưu trữ dữ liệu**	Vi điều khiển không đủ bộ nhớ để lưu dữ liệu dài hạn, chỉ có thể gửi dữ liệu lên server IoT.
5	**Thao tác thủ công**	Khi người dùng bật/tắt đèn thủ công, hệ thống phải tạm ngưng chế độ tự động để tránh xung đột.
6	**Phạm vi điều khiển**	Mỗi bộ điều khiển chỉ quản lý được một đèn hoặc một khu vực. Muốn mở rộng cần thêm thiết bị mới.

4. Mô hình yêu cầu

4.1. Use Case Diagram

- **Actor :**
  - **Giảng viên:** điều khiển đèn(bật/tắt), thay đổi chế độ điều khiển đèn(thủ công, tự động).
  - **Quản lí:** Đặt ngưỡng của mức độ sáng, thay đổi chế độ điều khiển đèn(thủ công, tự động).
- **Use case:**
  - Điều khiển đèn.
  - Đặt ngưỡng mức độ sáng.
  - Thay đổi chế độ.
- Biểu đồ



Các tính năng triển khai.

Hệ thống được thiết kế theo mô hình **Web Server cục bộ (Local Web Server)**, trong đó ESP32 đóng vai trò máy chủ lưu trữ và xử lý trung tâm, còn trình duyệt web đóng vai trò máy trạm hiển thị và điều khiển. Các chức năng được chia thành 4 nhóm chính:

1. Chức năng Thu thập và Giám sát dữ liệu (Monitoring)

Chức năng này đảm bảo việc số hóa các thông số môi trường và hiển thị trực quan cho người dùng.

- Thu thập độ rọi (Lux):
  - Mô tả: ESP32 đọc dữ liệu từ cảm biến BH1750 thông qua giao tiếp I2C (SDA-GPIO21, SCL-GPIO22).
  - Cơ chế cập nhật: Thay vì ESP32 tự gửi dữ liệu, hệ thống sử dụng cơ chế Polling (Hỏi vòng). Web Client sẽ gửi yêu cầu cập nhật (`GET /status`) định kỳ 1 giây/lần (`setInterval 1000ms`).
  - Dữ liệu trả về: Định dạng JSON bao gồm giá trị Lux thực tế và trạng thái của các thiết bị (Ví dụ: `{"lux": 320.5, ...}`).
- Biểu đồ hóa dữ liệu:
  - Mô tả: Giao diện Web sử dụng thư viện Chart.js để vẽ biểu đồ đường (Line Chart) thể hiện sự biến thiên của ánh sáng theo thời gian thực.
  - Lưu trữ tạm thời: Biểu đồ hiển thị cửa sổ dữ liệu của 20 điểm đo gần nhất để người dùng theo dõi xu hướng tăng/giảm độ sáng.

2. Chức năng Điều khiển Logic (Control Logic)

Hệ thống vận hành dựa trên biến trạng thái `autoMode` được lưu trong RAM của ESP32, chia làm 2 chế độ rõ rệt:

2.1. Chế độ Tự động (Auto Mode - Server Side Logic)

- Kích hoạt: Khi người dùng chọn chế độ "Tự động" trên Web (`/setMode?auto=1`).
- Nguyên lý hoạt động: Logic điều khiển nằm hoàn toàn dưới firmware của ESP32 trong vòng lặp `loop()`.

    - Ngưỡng Bật: Nếu giá trị cảm biến < 300 Lux → Hệ thống gọi hàm `setAll(1)` để bật toàn bộ 6 Relay.
    - Ngưỡng Tắt: Nếu giá trị cảm biến > 500 Lux → Hệ thống gọi hàm `setAll(0)` để tắt toàn bộ 6 Relay.
- Bảo vệ: Trong chế độ này, ESP32 sẽ từ chối các lệnh điều khiển đèn đơn lẻ từ người dùng (trả về lỗi HTTP 403 "Auto mode active").

2.2. Chế độ Thủ công

- Kích hoạt: Khi người dùng chọn chế độ "Thủ công" trên Web (`/setMode?auto=0`).
- Nguyên lý hoạt động: ESP32 vô hiệu hóa logic so sánh ngưỡng ánh sáng.
- Điều khiển chi tiết: Người dùng có quyền bật/tắt độc lập từng đèn trong số 6 đèn (tương ứng với 6 khu vực: Bàn GV, Giữa lớp, Cuối lớp) thông qua giao diện Web.

3. Chức năng Giao tiếp API (RESTful API Implementation)

Hệ thống không sử dụng Cloud trung gian mà giao tiếp trực tiếp qua mạng LAN (WiFi) thông qua các **API Endpoints** được định nghĩa trên ESP32:

Chức năng	Phương thức HTTP	API Endpoint	Tham số (Params)	Mô tả hành vi trong Code
Lấy trạng thái	GET	/status	Không	Trả về chuỗi JSON chứa: Lux, Mode, Mảng trạng thái 6 Relay.
Đổi chế độ	GET	/setMode	auto (0 hoặc 1)	Thay đổi biến toàn cục autoMode. Nếu auto=1 thì giao diện Web sẽ hiện lớp phủ khóa (Overlay).
Điều khiển đèn	GET	/toggle	id (0-5), state (0/1)	Thay đổi mức điện áp chân GPIO tương ứng (Active-LOW) để đóng/ngắt Relay.

4. Chức năng Giao diện người dùng

Giao diện được thiết kế dạng **Single Page Application (SPA)** nhúng trực tiếp hoặc chạy trên máy khách:

- **Đồng bộ trạng thái:** Hệ thống có tính năng "State Feedback". Nếu đèn bị tắt do logic Tự động, giao diện trên Web sẽ tự động chuyển icon bóng đèn sang màu xám (OFF) ngay lập tức nhờ cơ chế đồng bộ JSON.

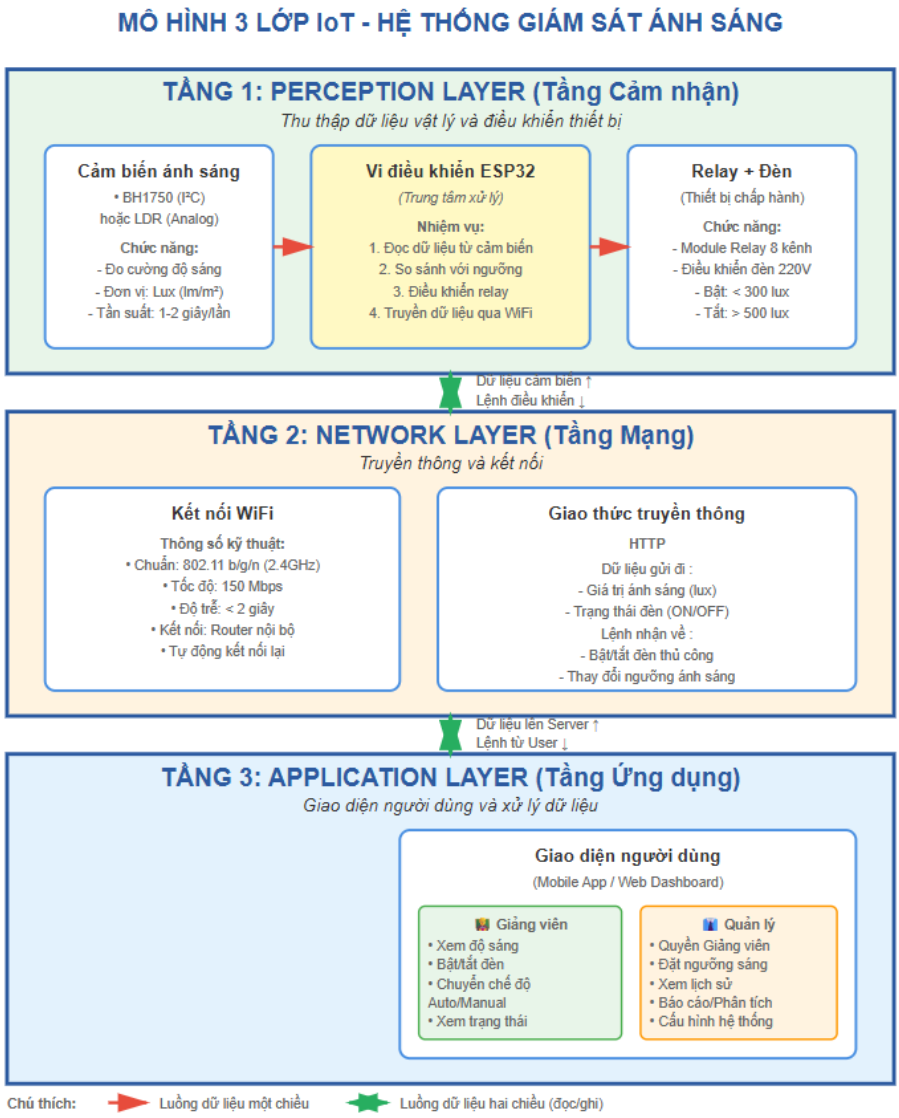
- **Cơ chế Khóa giao diện (Lock Overlay):**
    - Khi hệ thống nhận tín hiệu đang ở chế độ Tự động (`data.auto == 1`), JavaScript sẽ kích hoạt lớp phủ mờ (`overlay-lock`) đè lên các nút điều khiển đèn.
    - **Mục đích:** Ngăn chặn xung đột lệnh giữa người dùng và thuật toán tự động của ESP32.

IV. Phân tích thiết kế

1. Thiết kế kiến trúc hệ thống.

1.1. Mô hình 3 lớp IoT (Perception – Network – Application)

Tầng	Thành phần cụ thể	Chức năng chính
Tầng cảm nhận (Perception Layer)	- Cảm biến ánh sáng BH1750 - Relay điều khiển đèn - ESP32	Thu thập dữ liệu cường độ sáng trong phòng học và điều khiển bật/tắt đèn.
Tầng mạng (Network Layer)	- Kết nối Wi-Fi của ESP32 - Giao thức truyền thôngHTTP	Truyền dữ liệu từ cảm biến và trạng thái đèn lên nền tảng Cloud; nhận lệnh điều khiển từ người dùng.
Tầng ứng dụng (Application Layer)	- Web riêng	Giao diện để Giảng viên điều khiển đèn và Quản lý đặt ngưỡng cường độ sáng, thay đổi chế độ điều khiển.



2.2. Sơ đồ khối hệ thống (System Block Diagram)

Hệ thống hoạt động theo mô hình vòng kín (Closed-loop System) với ESP32 là trung tâm điều phối.

Chi tiết các khối chức năng:

- **Khối Nguồn (Power Block):**
  - Sử dụng Adapter 5V-2A cung cấp năng lượng nuôi vi điều khiển ESP32 và mạch kích của Module Relay.
  - Đảm bảo dòng điện ổn định, tách biệt nguồn động lực (Relay) và nguồn điều khiển (MCU) để tránh nhiễu.
- **Khối Cảm biến (Input Block):**

    - Sử dụng giao tiếp I2C để truyền dữ liệu độ rọi kỹ thuật số về ESP32.
- **Khối Xử lý trung tâm (MCU Block):**
    - **ESP32** thực hiện 3 nhiệm vụ song song: (1) Duy trì kết nối Wi-Fi, (2) Chạy máy chủ Web phục vụ API, (3) Chạy thuật toán so sánh ngưỡng ánh sáng (Auto Mode).
- **Khối Chấp hành (Output Block):**
    - Gồm 6 kênh Relay độc lập. Nhận tín hiệu kích mức thấp từ ESP32 để đóng mạch điện 220V cho đèn.
- **Khối Giao diện (User Interface):**
    - Thiết bị người dùng (Laptop/Smartphone) kết nối vào cùng mạng LAN với ESP32 để truy cập trang điều khiển.

1.3. Kiến trúc Truyền thông và Dữ liệu (Communication Architecture)

Hệ thống sử dụng mô hình **Client-Server** truyền thống qua giao thức HTTP:

- **Server (Máy chủ):** Vi điều khiển ESP32.
    - Lưu trữ giao diện Web (HTML/JS) trong bộ nhớ Flash.
    - Cung cấp các API RESTful (/status, /toggle, /setMode) để các thiết bị khác gọi vào.
- **Client (Máy khách):** Trình duyệt Web trên thiết bị người dùng.
    - Chịu trách nhiệm tải giao diện và thực thi mã JavaScript.
    - Sử dụng cơ chế **Polling (Hỏi vòng):** Chủ động gửi yêu cầu cập nhật trạng thái (GET /status) định kỳ mỗi 1 giây để đồng bộ dữ liệu, thay vì chờ Server gửi thông báo (Push).

1.4. Kiến trúc Triển khai (Deployment Architecture)

Hệ thống được triển khai theo mô hình **Mạng hình sao (Star Topology)** trong phạm vi lớp học:

- **Trung tâm kết nối:** Router Wi-Fi của phòng học (SSID: VIETTEL TANG 3).
- **Node thiết bị:** Bộ điều khiển ESP32 kết nối không dây tới Router và được cấp phát một địa chỉ IP tĩnh (Ví dụ: 192.168.1.39).
- **Node người dùng:** Các thiết bị của Giảng viên/Quản lý kết nối tới cùng Router để "nhìn thấy" ESP32.
- **Ưu điểm:** Không phụ thuộc vào đường truyền Internet quốc tế (mất mạng Internet vẫn điều khiển được miễn là Router còn điện), độ trễ thấp (< 150ms trong mạng LAN), bảo mật nội bộ.

2. Thiết kế phần cứng

2.1. Lựa chọn và Cấu hình linh kiện

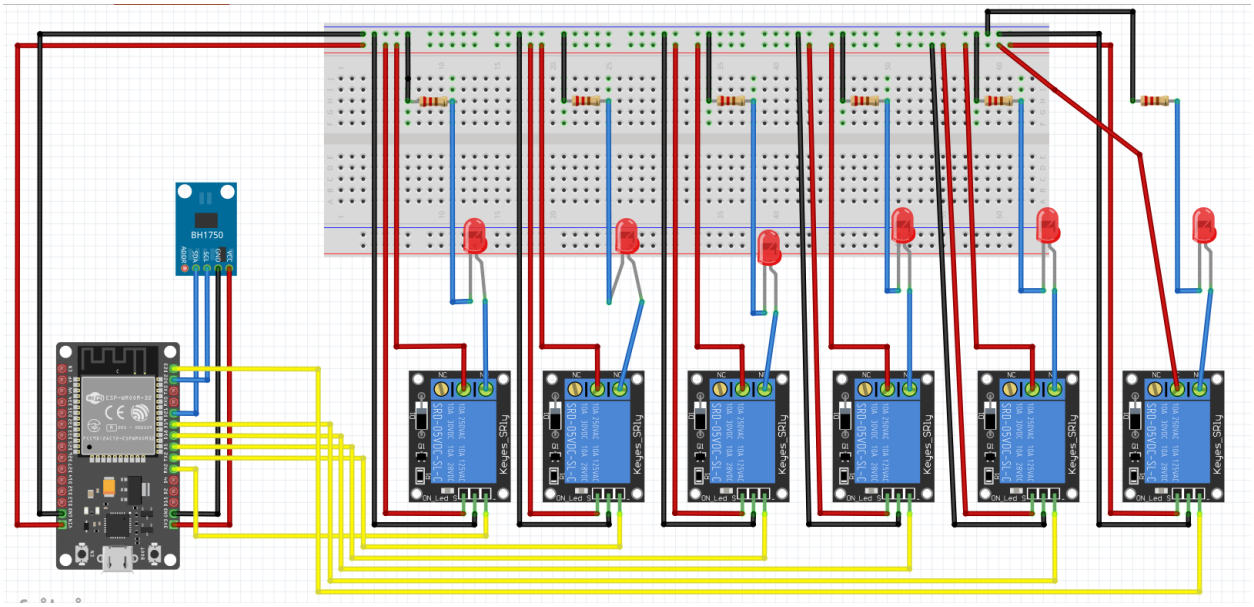
- **Vi điều khiển (MCU): ESP32 Development Board (30/38 pins)**
  - **Vai trò:** Đóng vai trò Web Server và bộ xử lý trung tâm.
  - **Cấu hình:** Sử dụng chế độ **Station (STA)** để kết nối vào mạng Wi-Fi của trường hoc..
  - **Điện áp:** Cấp nguồn 5V qua cổng MicroUSB hoặc chân VIN.
- **Cảm biến ánh sáng: BH1750 (Digital Light Sensor)**
  - **Giao tiếp:** Sử dụng chuẩn **I2C** (Inter-Integrated Circuit).
  - **Thư viện:** `BH1750.h` & `Wire.h`.
  - **Chế độ đo:** `CONTINUOUS_HIGH_RES_MODE` (Đo liên tục độ phân giải cao 1 Lux) để đảm bảo dữ liệu cập nhật mượt mà.
- **Thiết bị chấp hành: Module Relay 8 kênh (5V - Active LOW)**
  - **Logic kích hoạt:** Mức **THẤP (LOW/0V)** là BẬT Relay, mức **CAO (HIGH/5V)** là TẮT Relay.
  - **Cách ly:** Sử dụng Opto-coupler để cách ly tín hiệu điều khiển 5V của ESP32 với nguồn 5V của cuộn hút Relay.

2.2. Sơ đồ đấu nối chi tiết

Thiết bị	Chân thiết bị	Chân ESP32 (GPIO)	Loại tín hiệu
BH1750	SDA	GPIO 21	Dữ liệu I2C
	SCL	GPIO 22	Đồng hồ I2C
	VCC	3.3V	Nguồn
	GND	GND	Mass
Relay Module	IN1 (Đèn 1)	GPIO 16	Digital Output
	IN2 (Đèn 2)	GPIO 17	Digital Output
	IN3 (Đèn 3)	GPIO 5	Digital Output
	IN4 (Đèn 4)	GPIO 18	Digital Output
	IN5 (Đèn 5)	GPIO 19	Digital Output
	IN6 (Đèn 6)	GPIO 23	Digital Output

	VCC	5V (Nguồn ngoài)	Nguồn động lực

Sơ đồ mạch điện.



3. Thiết kế phần mềm

Hệ thống phần mềm được xây dựng theo kiến trúc **Embedded Web Server (Máy chủ Web nhúng)**, hoạt động hoàn toàn trên mạng cục bộ (LAN) mà không phụ thuộc vào Internet.

3.1. Thiết kế Firmware (Backend - ESP32)

Firmware được viết bằng C++ trên nền tảng Arduino IDE, chia thành 3 khối xử lý chính:

a. Khối khởi tạo (Setup)

- Khởi tạo giao tiếp Serial (115200 baud).
- Khởi tạo bus I2C cho cảm biến BH1750.
- Cấu hình 6 chân GPIO Relay là OUTPUT và đặt mức HIGH (Trạng thái tắt mặc định khi khởi động để an toàn).
- Kết nối Wi-Fi và in địa chỉ IP ra Serial Monitor.
- Định nghĩa các đường dẫn API (server.on(...)) và khởi chạy Web Server.

b. Khối Logic điều khiển (Loop)

Hệ thống hoạt động theo cơ chế **State Machine** dựa trên biến autoMode:

- **Đọc cảm biến:** Liên tục đọc giá trị lightMeter.readLightLevel().
- **Kiểm tra chế độ:**
  - **Nếu Auto Mode (1):** So sánh giá trị Lux với ngưỡng cứng (Hard-coded threshold):
    - Lux < 300: Gọi hàm setAll(1) $\rightarrow$ Kích mức LOW cho toàn bộ 6 Relay.
    - Lux > 500: Gọi hàm setAll(0) $\rightarrow$ Kích mức HIGH cho toàn bộ 6 Relay.
  - **Nếu Manual Mode (0):** Bỏ qua so sánh ngưỡng, chỉ duy trì kết nối Web Server để chờ lệnh từ người dùng.
- **Xử lý Client:** Gọi hàm server.handleClient() để liên tục lắng nghe các yêu cầu HTTP.

c. Thiết kế API (RESTful Endpoints)

ESP32 cung cấp các API hỗ trợ **CORS** (Cross-Origin Resource Sharing) để Web App truy cập:

Endpoint	Method	Tham số	Chức năng trong Code
/status	GET	Không	Trả về JSON chứa: giá trị Lux, trạng thái autoMode, mảng trạng thái 6 Relay.
/toggle	GET	id, state	Điều khiển bật/tắt từng đèn. Có kiểm tra điều kiện: Nếu autoMode == 1 sẽ trả về lỗi 403.
/setMode	GET	auto	Chuyển đổi biến toàn cục autoMode.

3.2. Thiết kế Ứng dụng Web (Frontend)

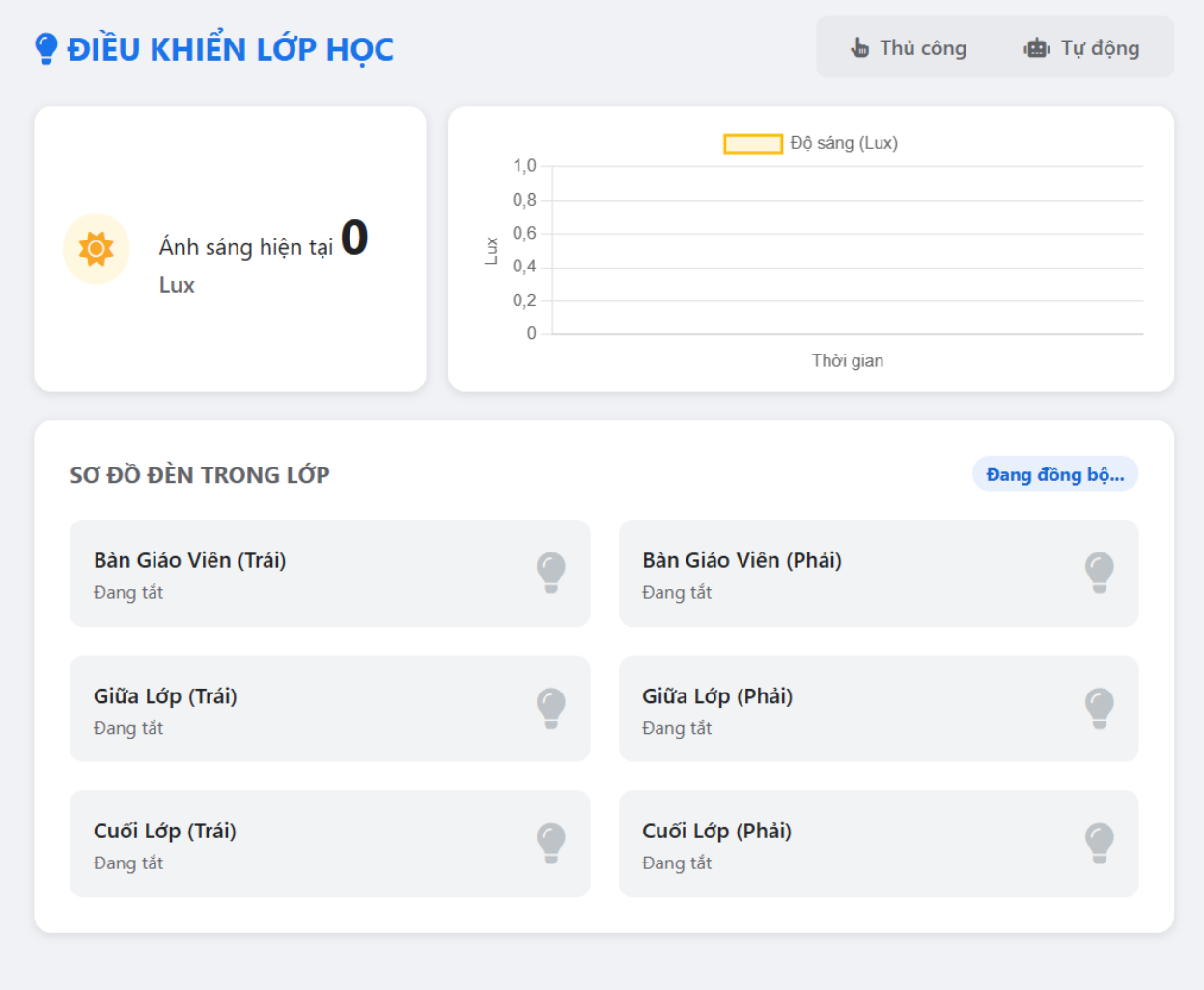
Giao diện người dùng là một trang đơn (Single Page Application) được xây dựng bằng HTML/CSS/JS thuần.

a. Giao diện người dùng (UI)

- **Dashboard:** Hiển thị giá trị Lux thời gian thực và biểu đồ đường (Line Chart) sử dụng thư viện `Chart.js`.
- **Control Grid:** Mô phỏng sơ đồ lớp học với 6 vị trí đèn. Mỗi đèn là một thẻ `<div>` có khả năng đổi màu (Vàng/Xám) và hiệu ứng đổ bóng `box-shadow` dựa trên trạng thái thực tế.
- **Mode Switch:** Nút chuyển đổi chế độ dạng Segmented Control.
- **Overlay Lock:** Một lớp phủ mờ (`<div id="lockOverlay">`) chứa icon ổ khóa, tự động hiện lên che khuất các nút đèn khi hệ thống ở chế độ Tự động.

b. Logic xử lý (JavaScript Client-side)

- **Cơ chế Polling (Hỏi vòng):** Sử dụng hàm `setInterval(updateStatus, 1000)` để gửi yêu cầu `GET /status` đến ESP32 mỗi 1 giây.
    - *Mục đích:* Đồng bộ hóa dữ liệu hiển thị (số Lux, trạng thái đèn) với thực tế mà không cần tải lại trang.
- **Xử lý sự kiện (Event Handling):**
    - Khi người dùng click vào đèn: Gửi `fetch(BASE + '/toggle?id=...')`.
    - Khi người dùng click đổi chế độ: Gửi `fetch(BASE + '/setMode?auto=...')`.
- **Trực quan hóa dữ liệu:** Hàm `addDataToChart()` đẩy dữ liệu Lux mới vào mảng dữ liệu của biểu đồ và xóa dữ liệu cũ (cửa sổ trượt 20 điểm) để biểu đồ luôn chạy theo thời gian thực.

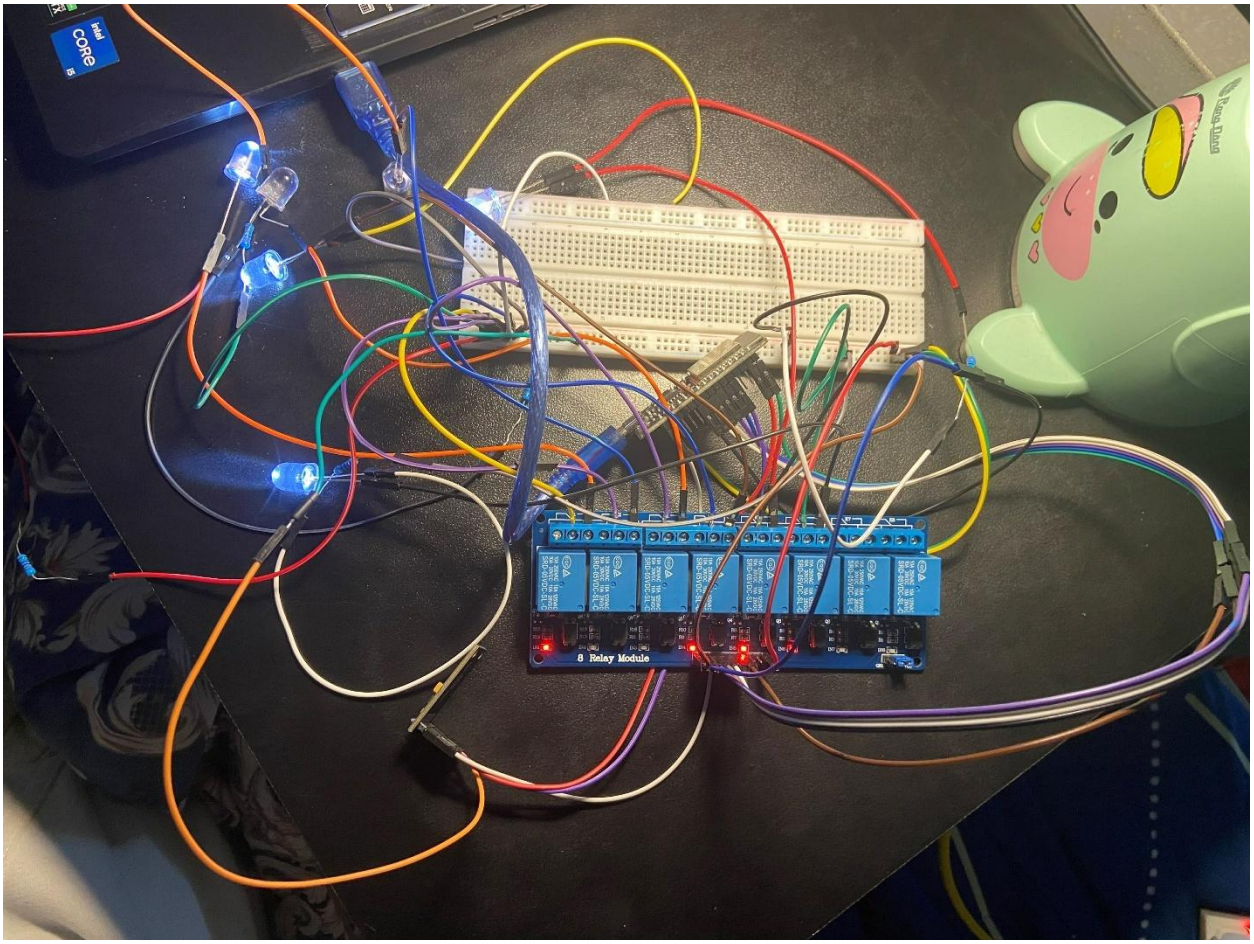


V. Kết quả đạt được

Sau quá trình thiết kế và thi công, nhóm đã hoàn thiện hệ thống "Giám sát ánh sáng và Điều khiển đèn phòng học thông minh" với các kết quả cụ thể như sau:

1. Về Phần cứng (Hardware)

- Hoàn thiện mô hình mạch điện thực tế gọn gàng, bao gồm: Vi điều khiển ESP32, Cảm biến BH1750 và Module Relay 6 kênh.
- Hệ thống hoạt động ổn định với nguồn cấp 5V-2A, không xảy ra hiện tượng sụt áp hay khởi động lại (Reset) khi kích hoạt đồng thời cả 6 Relay.
- Cảm biến BH1750 đo được giá trị độ rọi (Lux) với độ nhạy cao, phản ứng tức thời khi có thay đổi ánh sáng môi trường.



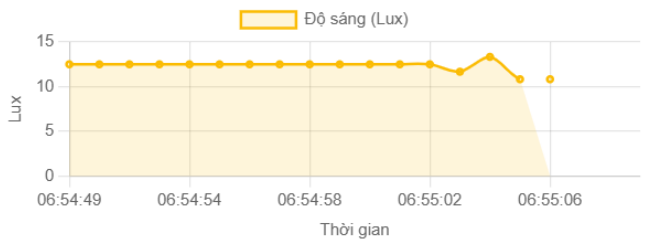
2. Về Phần mềm và Giao diện

- **Giao diện Web Dashboard:**
  - Truy cập ổn định qua địa chỉ IP tĩnh (Ví dụ: `192.168.1.39`) trên trình duyệt máy tính và điện thoại.
  - Biểu đồ (Chart.js) vẽ đường biến thiên ánh sáng mượt mà theo thời gian thực (cập nhật 1 giây/lần).
  - Các nút bấm điều khiển 6 đèn phản hồi tốt, có hiệu ứng đổi màu (Vàng/Xám) tương ứng với trạng thái thực của đèn.
- **Chức năng Logic:**
  - **Chế độ Thủ công:** Người dùng điều khiển bật/tắt độc lập từng đèn thành công. Độ trễ từ lúc nhấn nút đến khi Relay đóng ngắt là dưới 1 giây.
  - **Chế độ Tự động:** Hệ thống tự động kích hoạt chế độ khóa giao diện (Lock Overlay). Khi che cảm biến (Lux < 300), cả 6 đèn tự động bật. Khi chiếu sáng vào cảm biến (Lux > 500), cả 6 đèn tự động tắt.

ĐIỀU KHIỂN LỚP HỌC

Thủ công | Tự động

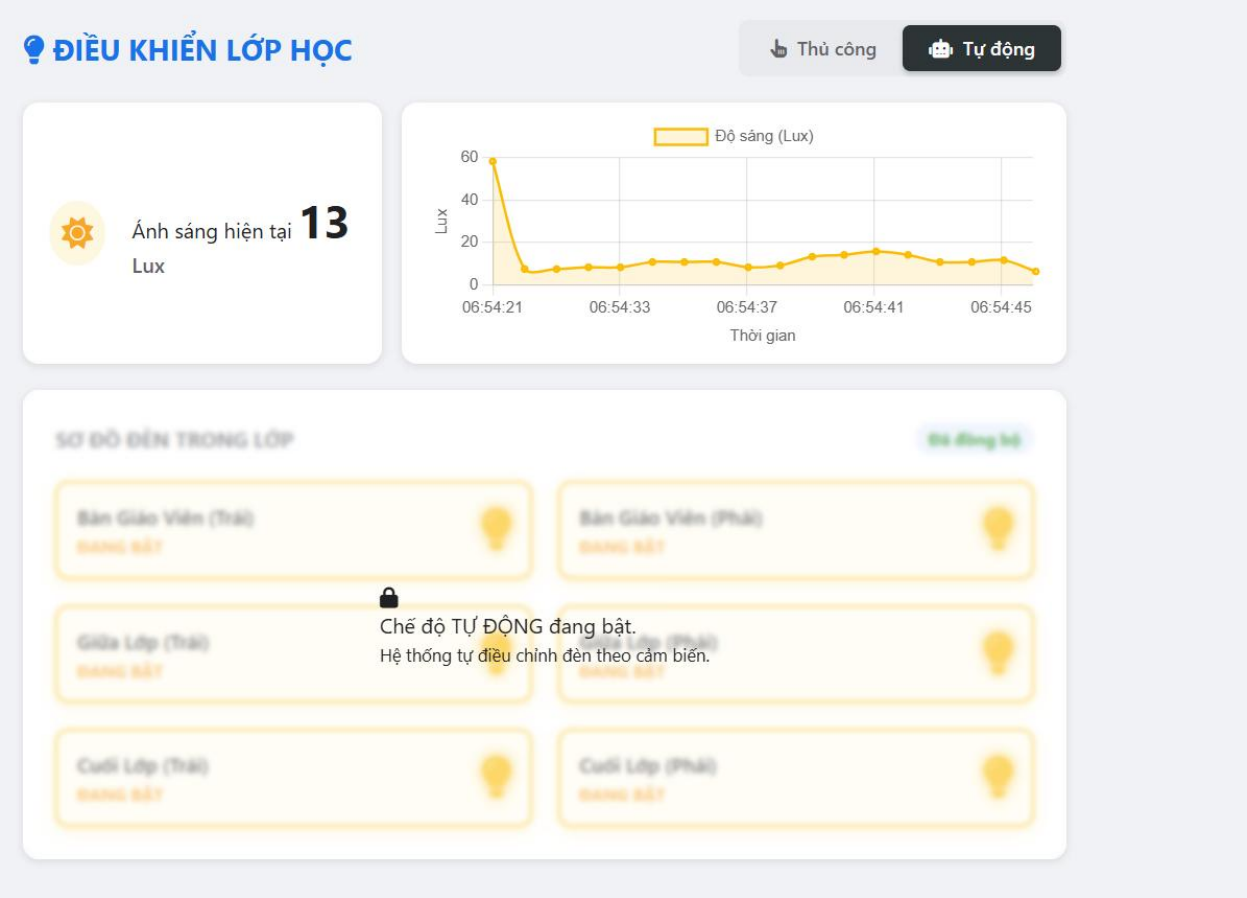
Ánh sáng hiện tại **12** Lux



SƠ ĐỒ ĐÈN TRONG LỚP

Đã đồng bộ

- Bàn Giáo Viên (Trái) — ĐANG BẬT
- Bàn Giáo Viên (Phải) — Đang tắt
- Giữa Lớp (Trái) — Đang tắt
- Giữa Lớp (Phải) — ĐANG BẬT
- Cuối Lớp (Trái) — ĐANG BẬT
- Cuối Lớp (Phải) — ĐANG BẬT



3. Đánh giá kết quả thử nghiệm.

Kịch bản kiểm thử (Test Case)	Hành động	Kết quả mong đợi	Kết quả thực tế	Đánh giá
TC-01: Giám sát	Mở Web Dashboard	Hiển thị đúng số Lux hiện tại	Hiển thị chính xác (sai số <5%)	**Đạt**
TC-02: Auto Mode (Bật)	Chọn Auto → Che cảm biến (< 300 Lux)	6 Relay đóng, Giao diện báo ON	6 đèn sáng, Web hiện ổ khóa	**Đạt**
TC-03: Auto Mode (Tắt)	Chọn Auto → Chiếu đèn (> 500 Lux)	6 Relay ngắt, Giao diện báo OFF	6 đèn tắt, Web hiện ổ khóa	**Đạt**
TC-04: Manual Mode	Chọn Manual $\rightarrow$ Bấm Đèn 1	Chỉ Đèn 1 sáng	Đèn 1 sáng, các đèn khác tắt	**Đạt**
TC-05: Chuyển chế độ	Chuyển từ Auto sang Manual	Mở khóa nút bấm	Giao diện cho phép bấm nút	**Đạt**

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Đề tài "Thiết kế hệ thống IoT giám sát ánh sáng và điều khiển đèn tự động trong phòng học" đã hoàn thành các mục tiêu đề ra ban đầu:

- **Xây dựng thành công** giải pháp giám sát và điều khiển thiết bị điện thông qua mạng Wi-Fi nội bộ (LAN) sử dụng vi điều khiển ESP32.
- **Ứng dụng mô hình Web Server nhúng:** Loại bỏ sự phụ thuộc vào các nền tảng Cloud bên thứ 3 (như Blynk), giúp hệ thống hoạt động độc lập, bảo mật và phản hồi nhanh.
- **Giải quyết bài toán thực tế:** Hệ thống giúp tiết kiệm điện năng nhờ cơ chế tự động tắt đèn khi trời sáng, đồng thời mang lại sự tiện lợi cho giảng viên qua giao diện điều khiển trên Smartphone.
- **Tối ưu chi phí:** Sử dụng các linh kiện phổ biến, giá thành thấp nhưng hiệu quả cao, phù hợp để triển khai nhân rộng cho nhiều phòng học.

2. Hạn chế

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số hạn chế:

- **Phạm vi kết nối:** Chỉ hoạt động trong mạng nội bộ (LAN). Người dùng không thể điều khiển khi ra khỏi trường (trừ khi cấu hình Port Forwarding hoặc VPN).
- **Bảo mật:** Chưa có cơ chế đăng nhập (Username/Password), bất kỳ ai kết nối vào WiFi đều có thể truy cập trang điều khiển nếu biết IP.
- **Cài đặt cứng:** Ngưỡng ánh sáng (300/500 Lux) đang được lập trình cố định trong Code, người dùng chưa thể tự thay đổi ngưỡng này trên giao diện Web.

3. Hướng mở rộng và phát triển

Để hoàn thiện sản phẩm và đưa vào ứng dụng thực tế quy mô lớn, nhóm đề xuất các hướng phát triển sau:

1. **Tích hợp cảm biến chuyển động (PIR/Radar):**
    - Bổ sung logic: Chỉ bật đèn khi "Trời tối" **VÀ** "Có người trong phòng". Tránh lãng phí điện khi trời tối nhưng không có lớp học.
2. **Mở rộng kết nối Internet (IoT Cloud):**
    - Sử dụng giao thức **MQTT** để kết nối hệ thống lên Cloud Server, cho phép quản lý phòng học từ bất kỳ đâu qua Internet.
3. **Nâng cấp giao diện quản lý:**

- Thêm trang Đăng nhập/Đăng ký.
- Thêm tính năng cài đặt ngưỡng ánh sáng (Threshold Slider) ngay trên Web App.
- Lưu trữ lịch sử dữ liệu vào thẻ nhớ SD hoặc Database để xuất báo cáo tiết kiệm điện hàng tháng.

4. **Phát triển tính năng học thói quen (AI/Machine Learning):**
   - Ghi nhận thói quen bật đèn thủ công của giảng viên để tự động điều chỉnh ngưỡng sáng phù hợp nhất cho từng lớp học.