**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**

**A red circle with a yellow star and a book

AI-generated content may be incorrect.**

**BÀI TẬP LỚN CUỐI KỲ**

**MÔN: QUẢN LÝ DỰ ÁN PHẦN MỀM**

|  |  |
| --- | --- |
| **GIẢNG VIÊN** | **: Ts. KIM NGỌC BÁCH** |
| **SINH VIÊN** | **: NGUYỄN ANH TUẤN – B22DCCN757** |
| **: NGUYỄN QUÝ HẠNH – B22DCCN277** |
| **NHÓM** | **: 16** |

**Hà Nội – 11/2025**

# CHƯƠNG 1: TỔNG QUAN

## 1.1 Giới thiệu tổng quan về IOT

Thuật ngữ Internet of things (viết tắt là IoT) được hiểu một cách đơn giản là một mạng lưới vạn vật kết nối với nhau thông qua Internet. Chúng bao gồm các đồ vật, con người được cung cấp một định danh của riêng mình và tất cả có khả năng truyền tải hay trao đổi thông tin hay dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính.

IoT đã phát triển từ sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và quan trọng hơn là sự có mặt của Internet. Nói đơn giản IoT là một tập hợp các thiết bị có khả năng kết nối mọi thứ lại với nhau với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

Có thể nói IoT có mặt trong hầu hết các lĩnh vực đời sống của chúng ta hiện nay từ văn hóa, du lịch, giáo dục, y học, truyền thông,...Tất cả các lĩnh vực trên đều được ứng dụng từ các thành tựu của IoT cả. Như trước đây nếu chúng ta muốn thông tin liên lạc với nhau thì không còn cách nào khác đó là viết thư và truyền thư thông qua người giao nhận. Còn bây giờ thì thông qua các thiết bị thông minh, chúng được kết nối với nhau thông qua hệ thống wifi, 3G hay các ứng dụng riêng của từng hãng. Chưa đầy 10s chúng ta đã có thể gửi tin nhắn đến một người hay nhiều người khác. Bên cạnh đó thì còn có nhiều ứng dụng khác như:

* Quản lý môi trường Quản lý hệ thống máy móc
* Hệ thống mua sắm trực tuyến
* Hệ thống kiểm soát an ninh
* Nhà thông minh
* Ứng dụng quản lý toàn bộ thiết bị cá nhân thông qua việc động bộ

## 1.2 Giới thiệu hệ thống

Hệ thống giàn phơi/mái che thông minh mini là một mô hình mô phỏng việc kéo ra/thu vào giàn phơi ở ban công căn hộ,với khả năng dự đoán thời tiết và điều khiển thông minh. Người dùng có thể bấm nút trên app/web (MQTT qua Wi-Fi nhà) để Mở/Đóng/Stop thủ công. Ngoài ra, hệ thống còn tự động phân tích dữ liệu cảm biến cục bộ, kết hợp với dữ liệu dự báo thời tiết từ OpenWeather API và dự đoán từ mô hình RNN, để đưa ra lịch đóng mở phù hợp và tối ưu hơn, dựa theo thời tiết hiện tại và dự kiến.

## 1.3 Mục đích nghiên cứu

**Mục tiêu đề tài:**

* Giải bài toán “phơi tự động” cơ bản với chi phí thấp: cảm biến mưa–nắng → quyết định → điều khiển motor có giới hạn hành trình, tích hợp OpenWeather API và mô hình RNN để đưa ra quyết định dự đoán.
* Đánh giá độ tin cậy trong bối cảnh thực: tránh “đóng–mở liên tục” khi mưa/nắng chập chờn (dùng trễ xác nhận mưa vài giây, ngưỡng lux đủ nắng), kết hợp với logic dự đoán từ RNN để đưa ra quyết định ổn định hơn.
* Thử nghiệm điều khiển từ xa qua Wi-Fi + MQTT: gửi lệnh, nhận trạng thái tức thời; làm nền cho việc thêm lịch hẹn hoặc thông báo sau này.
* Giao diện người dùng thân thiện: Người dùng có thể theo dõi và điều khiển hệ thống từ xa qua ứng dụng di động hoặc website.

**Đối tượng sử dụng:**

Hệ thống được sử dụng cho các hộ chung cư hay các hộ gia đình có không gian khiêm tốn, không thường xuyên có người ở nhà, không có điều kiện hoặc không muốn đầu tư máy sấy.

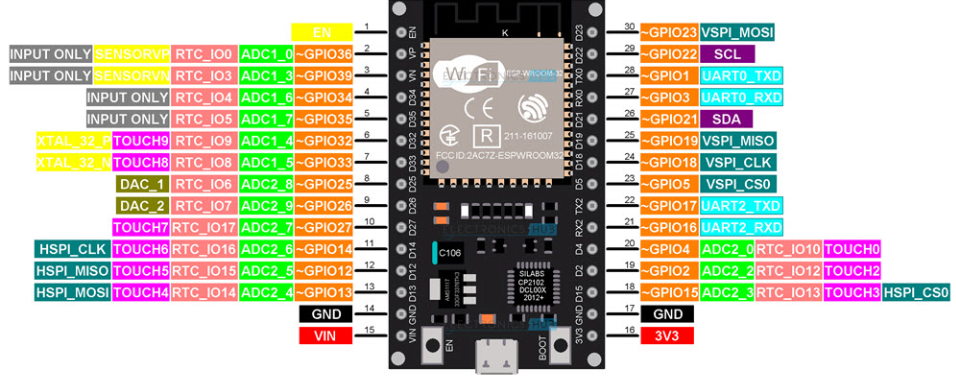
**Lý thuyết:**

* Hệ thống mô phỏng thao tác kéo ra/thu vào giàn phơi
* Hệ thống đọc cảm biến mưa (raindrop) và cảm biến ánh sáng BH1750 để quyết định trạng thái vận hành (mở khi nắng đủ, thu khi mưa)
* Thiết bị giao tiếp MQTT qua Wi-Fi để nhận lệnh từ xa (Open/Close/Stop), lưu trữ lịch sử đóng/mở và trạng thái/telemetry lên nền tảng đám mây, sau đó gửi về về dashboard/app.

## 1.4 Thiết bị phần cứng và nền tảng phần mềm

### 1.4.1 Các thiết bị phần cứng

#### a. Bộ kit ESP32 CP3102



ESP32 là SoC Wi-Fi 2.4 GHz + Bluetooth v4.2 (BR/EDR + BLE) tích hợp MCU Xtensa LX6 hai nhân (lên đến 240 MHz), SRAM ~520 KB, dùng module ESP32-WROOM-32(E/UE) gắn trên bo.

CP2102 là chip USB-to-UART của Silicon Labs, biến cổng USB máy tính thành cổng UART để nạp firmware và log serial. Trên DevKit, CP2102 còn nối DTR/RTS → EN/IO0 để tự vào chế độ bootloader khi nạp.

**Thông số chính:**

**CPU**: Xtensa LX6 2 nhân, 80–240 MHz.

**Bộ nhớ:** ~520 KB SRAM trong chip; Flash ngoài (thường 4 MB) trong module.

**Kết nối**: Wi-Fi 802.11 b/g/n 2.4 GHz; Bluetooth BR/EDR + BLE v4.2.

**GPIO:** tối đa 34 GPIO mức 3.3 V (một số chỉ input).

**Analog/TouchS:** ADC 12-bit (tối đa 18 kênh), DAC 8-bit (GPIO25, 26), 10 kênh cảm ứng chạm.

**Peripherals:** UART, SPI (VSPI/HSPI), I²C, I²S, PWM (LEDC), SD/SDIO, CAN (TWAI).

**Nguồn vào**: USB/VIN 5 V; I/O chỉ 3.3 V (không 5 V trực tiếp vào GPIO).

**Khái quát các chân của kit:**

**Nguồn:** VIN(5V), 3V3, nhiều GND.

**UART:** TX0/RX0 (nối CP2102 để nạp & log), có thêm UART1, UART2 remap linh hoạt.

**I²C:** thường dùng GPIO21 (SDA) và GPIO22 (SCL) (có thể remap).

SPI (VSPI mặc định): SCK(GPIO18), MOSI(GPIO23), MISO(GPIO19), CS(GPIO5).

**ADC:** nhóm GPIO32–39…; GPIO34–39 chỉ input.

**DAC:** GPIO25, GPIO26.

**Touch:** ví dụ GPIO4, 12–15, 27, 32, 33…

**Boot/Reset:** EN (reset), IO0 (giữ LOW khi reset để vào bootloader).

GPIO6–11 nối flash QSPI trong module, không dùng làm GPIO.

**Nguyên lý hoạt động:**

**Cấp nguồn & nhận cổng COM:** cắm USB → CP2102 xuất hiện trên máy tính như cổng Serial.

**Nạp firmware:** công cụ nạp (Arduino IDE/ESP-IDF esptool) kéo DTR/RTS để đặt IO0=LOW rồi reset EN → ESP32 vào UART bootloader → ghi flash → tự reset chạy ứng dụng.

**Chạy ứng dụng:** lõi CPU thực thi, Wi-Fi/BLE hoạt động song song; ngoại vi (UART/I²C/SPI/ADC/DAC/PWM…) gán ra GPIO; hỗ trợ sleep/deep-sleep đánh thức bằng timer/GPIO.

**Giao tiếp với các thiết bị ngoại vi:** Bộ kit hỗ trợ nhiều giao thức giao tiếp như UART, I2C và SPI. Các chân GPIO (General Purpose Input/Out- put) có thể được lập trình để đọc dữ liệu từ cảm biến hoặc điều khiển thiết bị khác. Ví dụ, chân ADCO có thể đọc tín hiệu analog từ cảm biến và chuyển đổi nó thành tín hiệu số để xử lý.

**Xử lý dữ liệu:** Dữ liệu nhận được từ các cảm biến hoặc thiết bị ngoại vi được vi điều khiển xử lý thông qua mã lập trình đã nạp vào bộ nhớ Flash. Bộ kit có khả năng thực hiện các phép toán logic và quyết định dựa trên dữ liệu nhận được.

**Gửi và nhận dữ liệu:** Sau khi xử lý, BỘ kit có thể gửi dữ liệu lên máy chủ hoặc nhận dữ liệu từ máy chủ thông qua giao thức HTTP, MQTT hoặc WebSocket. Điều này cho phép bộ kit hoạt động trong các ứng dụng IoT như giám sát, điều khiển từ xa và thu thập dữ liệu.

**Tương tác người dùng:** Bộ kit có thể được kết nối với các thiết bị như màn hình LCD, cảm biến và động cơ, cho phép người dùng tương tác với hệ thống thông qua các ứng dụng di động hoặc web.

#### b. Động cơ DC giảm tốc

Là motor DC nhỏ có hộp số, lực tốt ở điện áp 6–9 V, phù hợp kéo/thu mô hình. (trong thực tế nên sử dụng động cơ có công suất lớn hơn)



**Thông số kỹ thuật:**

**Điện áp danh định:** 3–9 V

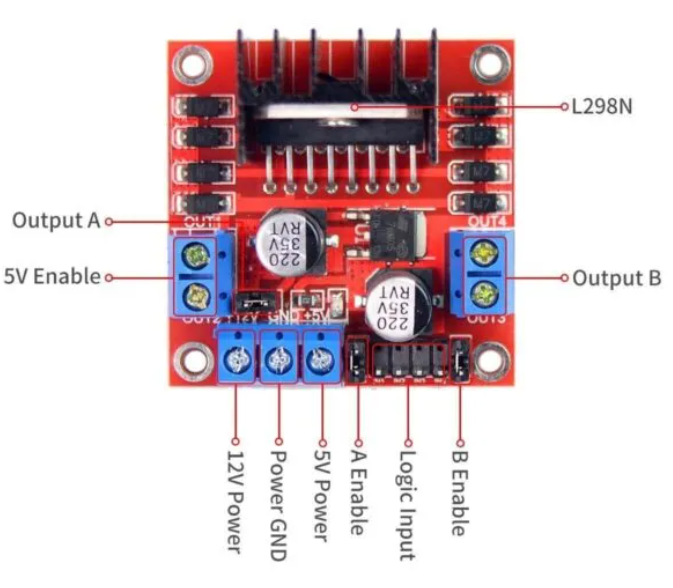
**Tốc độ không tải:** phụ thuộc tỷ số truyền (ví dụ 30–300 rpm).

**Dòng không tải:** 100–250 mA; dòng kẹt/stall: 0.8–1.5 A (tuỳ loại).

**Trục:** đường kính 3–5 mm (gắn puly/ống cuộn).

Cấp điện 9V qua module L298N

#### c. Module L298N



Là driver cầu H 2 kênh cho motor DC, nhận tín hiệu điều khiển từ ESP32.

**Thông số chính:**

Điện áp phần công suất (VS): 5–35 V

Dòng: tối đa ~2 A/kênh (đỉnh ngắn; thực tế nên ≤1–1.5 A liên tục, có tản nhiệt).

Điện áp logic (Vss):5 V (lấy từ on-board regulator hoặc nguồn 5 V riêng).

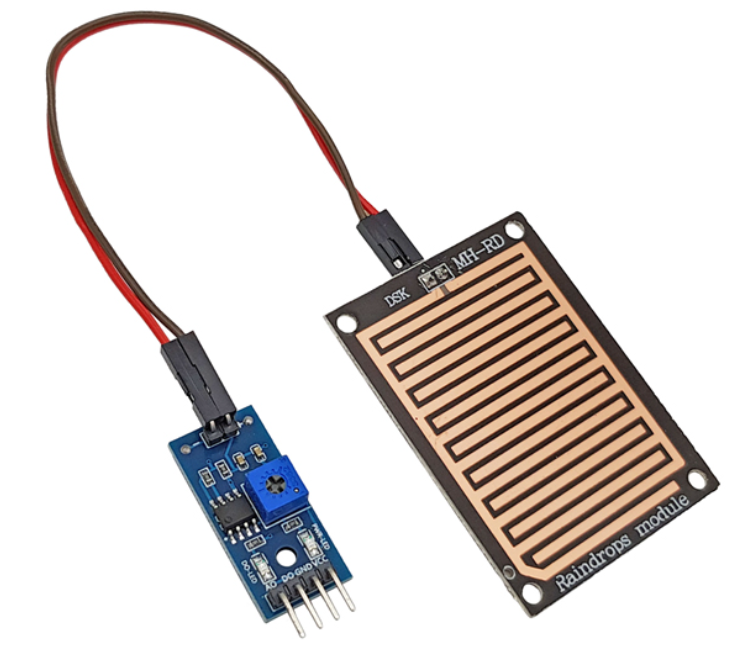
Sụt áp lớn (~1.5–2 V) → motor 9 V sẽ nhận ~7–7.5 V khi tải.

**Cấp nguồn:**

VS: từ buck 9 V; GND chung với ESP32.

5 V logic: dùng buck 5 V cấp cho ESP32; không dùng 5 V của L298N nuôi ESP32.

#### d. Cảm biến mưa



Là tấm dò mưa + board so sánh (LM393), xuất D0 (digital) theo ngưỡng chỉnh biến trở, và A0 (analog).

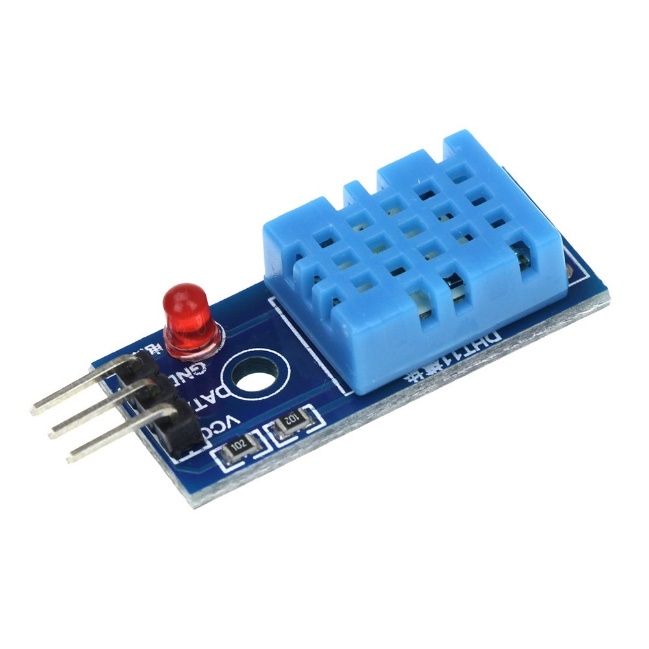
**Thông số chính:**

Điện áp hoạt động: 3.3–5 V (nhiều bản ghi 5 V, nhưng vẫn chạy 3.3 V).

Ngõ ra D0: mức logic theo ngưỡng; A0: điện áp tỷ lệ ướt/khô.

**Cấp nguồn:** dùng 3.3 V để an toàn mức logic với ESP32, hoặc 5 V nhưng chỉ dùng D0 nếu bảo đảm tương thích 3.3 V

#### e. DHT11



Là cảm biến nhiệt độ/độ ẩm giá rẻ để hiển thị môi trường (phục vụ thống kê/DryScore đơn giản).

Thông số chính

Điện áp: 3.3–5.5 V.

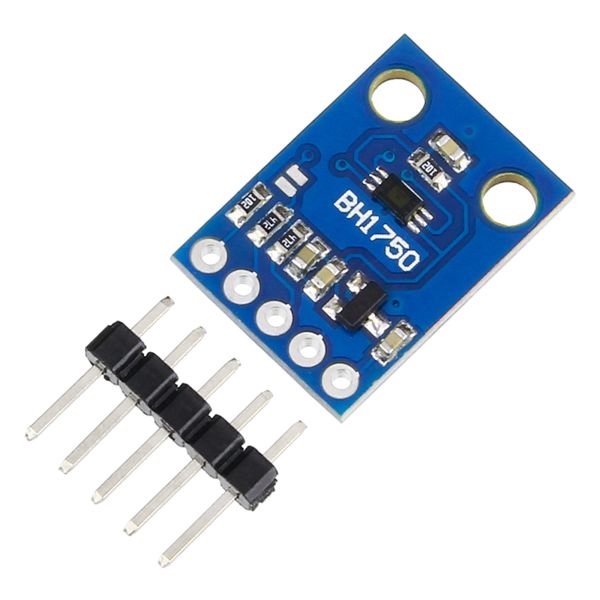
Nhiệt độ: 0–50 °C, độ chính xác khoảng ±2 °C.

Độ ẩm: 20–80 %RH, độ chính xác khoảng ±5 %RH.

Tần suất đo: ~1 Hz (1 lần/giây).

Cấp nguồn 3.3 V (hoặc 5 V); GND chung.

#### f. BH1750



Là cảm biến ánh sáng đo **lux** trực tiếp, dùng để biết khi nào “đủ nắng” để có thể mở/đóng

**Thông số chính:**

Dải đo: ~1–65,535 lux (nhiều mode).

I²C địa chỉ mặc định: 0x23 (0x5C tùy module).

Điện áp: 3.3–5 V; logic I²C 3.3 V-tương thích.

Cấp nguồn 3.3 V (hoặc 5 V); GND chung.

#### g. Các linh kiện khác:

- Nguồn DC 12V: nguồn cấp chính cho thiết bị

- Module hạ áp Lm2596: Dùng để hạ áp từ 12V về 9V (nuôi motor) và 5V (cho ESP32)

- Breadboard, dây dupont...

### 1.4.2 Các nền tảng phần mềm

#### a. Arduino IDE

Arduino IDE là một phần mềm soạn thảo văn bản chính hãng, giúp viết code để nạp vào bo mạch Arduino một cách nhanh chóng, dễ dàng và hoàn toàn miễn phí.Tuy là phần mềm mã nguồn mở nhưng khả năng bảo mật thông tin của Arduino IDE là vô cùng tuyệt vời, khi phát hiện lỗi nhà phát hành sẽ vá nó và cập nhật rất nhanh khiến thông tin của người dùng không bị mất hoặc rò rỉ ra bên ngoài.

Khi người dùng viết mã và biên dịch, IDE sẽ tạo file Hex cho mã. File Hex là các file thập phân Hexa được Arduino hiểu và gửi đến bo mạch bằng cáp USB. Mỗi bo Arduino đều được tích hợp một bộ vi điều khiển, bộ vi điều khiển sẽ nhận file Hex và chạy theo mã được viết.

Arduino IDE sử dụng ngôn ngữ lập trình C/C++ rất phổ biến trong giới lập trình. Bất kỳ đoạn code nào của C/C++ thì Arduino IDE đều có thể nhận dạng, giúp các lập trình viên thuận tiện trong việc thiết kế chương trình lập cho các bo mạch Arduino.

Arduino IDE tích hợp với hơn 700 thư viện, được viết và chia sẻ bởi nhà phát hành Arduino Software và thành viên trong cộng đồng Arduino.

Arduino IDE có một giao diện đơn giản, dễ sử dụng giúp người dùng thuận tiện hơn trong thao tác. Dưới đây là một số tính năng chúng ta thường sử dụng:

Nút kiểm tra chương trình (Verify): giúp dò lỗi phần code định truyền xuống bo mạch Arduino.

Nút tải đoạn code vào bo mạch Arduino (Upload): giúp nhập đoạn code vào bo mạch Arduino.

Vùng lập trình: người dùng sẽ viết chương trình tại khu vực này.

Thanh Menu: gồm những thẻ chức năng nằm trên cùng như File, Edit, Sketch, Tools, Help

#### b. Firebase

Giới thiệu về firebase

Firebase là nền tảng phát triển ứng dụng di động của Google giúp chúng ta xây dựng, cải thiện và phát triển ứng dụng của mình. Nó có nhiều dịch vụ được sử dụng để quản lý dữ liệu từ bất kỳ ứng dụng Android, iOS hoặc web nào. Firebase cung cấp các dịch vụ:

Xác thực: đăng nhập / đăng xuất và nhận dạng người dùng.

Cơ sở dữ liệu thời gian thực: cơ sở dữ liệu NoSQL theo thời gian thực, được lưu trữ trên đám mây, dữ liệu được lưu trữ dưới dạng cấu trúc JSON;

Cloud Functions: Được sử dụng để định kỳ gọi OpenWeather API (mục e), xử lý dữ liệu dự báo thời tiết và lưu trữ vào Firestore. Đồng thời, Cloud Functions cũng là nơi triển khai mô hình RNN (mục f) để phân tích dữ liệu cảm biến và dự đoán trạng thái môi trường, sau đó ra quyết định điều khiển motor thông minh.

Cloud Firestore: Lưu trữ dữ liệu cảm biến, trạng thái thiết bị, lệnh điều khiển, dữ liệu dự báo thời tiết, ...

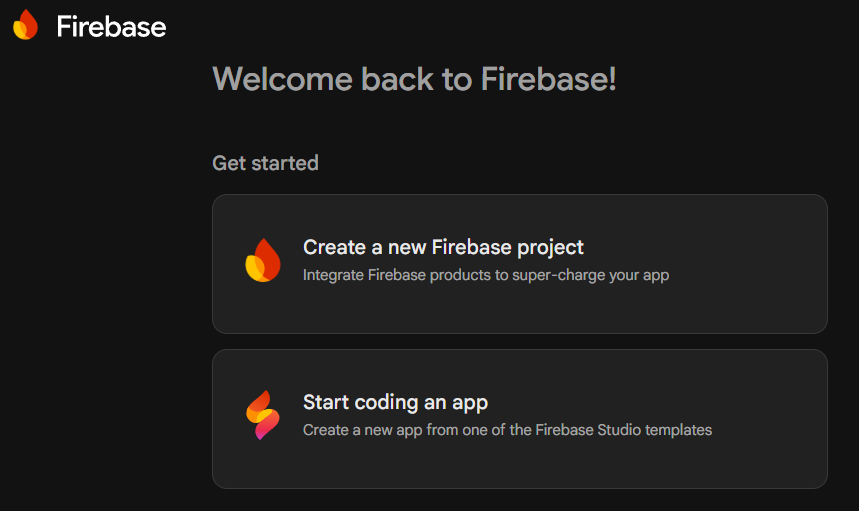
Lưu trữ đám mây: lưu trữ tệp có thể mở rộng để tải lên và tải xuống tệp

Chức năng đám mây: chương trình phụ trợ hướng sự kiện, không có máy chủ;

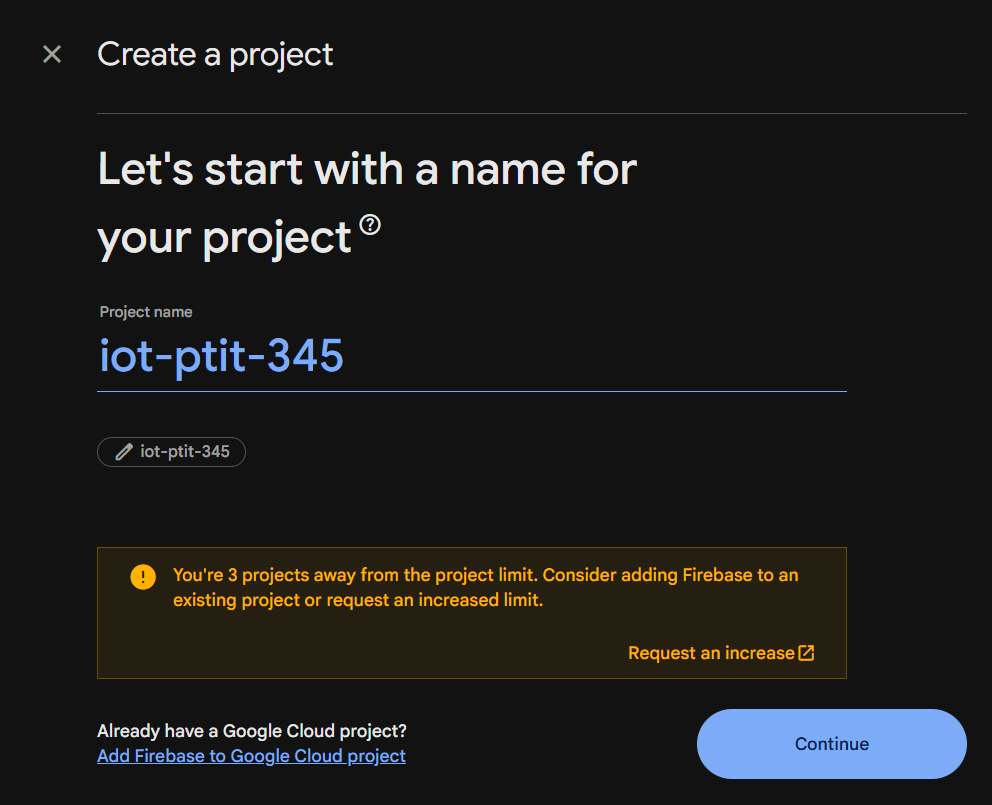
Firebase hosting: lưu trữ web toàn cầu có hoặc không có tên miền tùy chỉnh

Tạo dự án Firebase

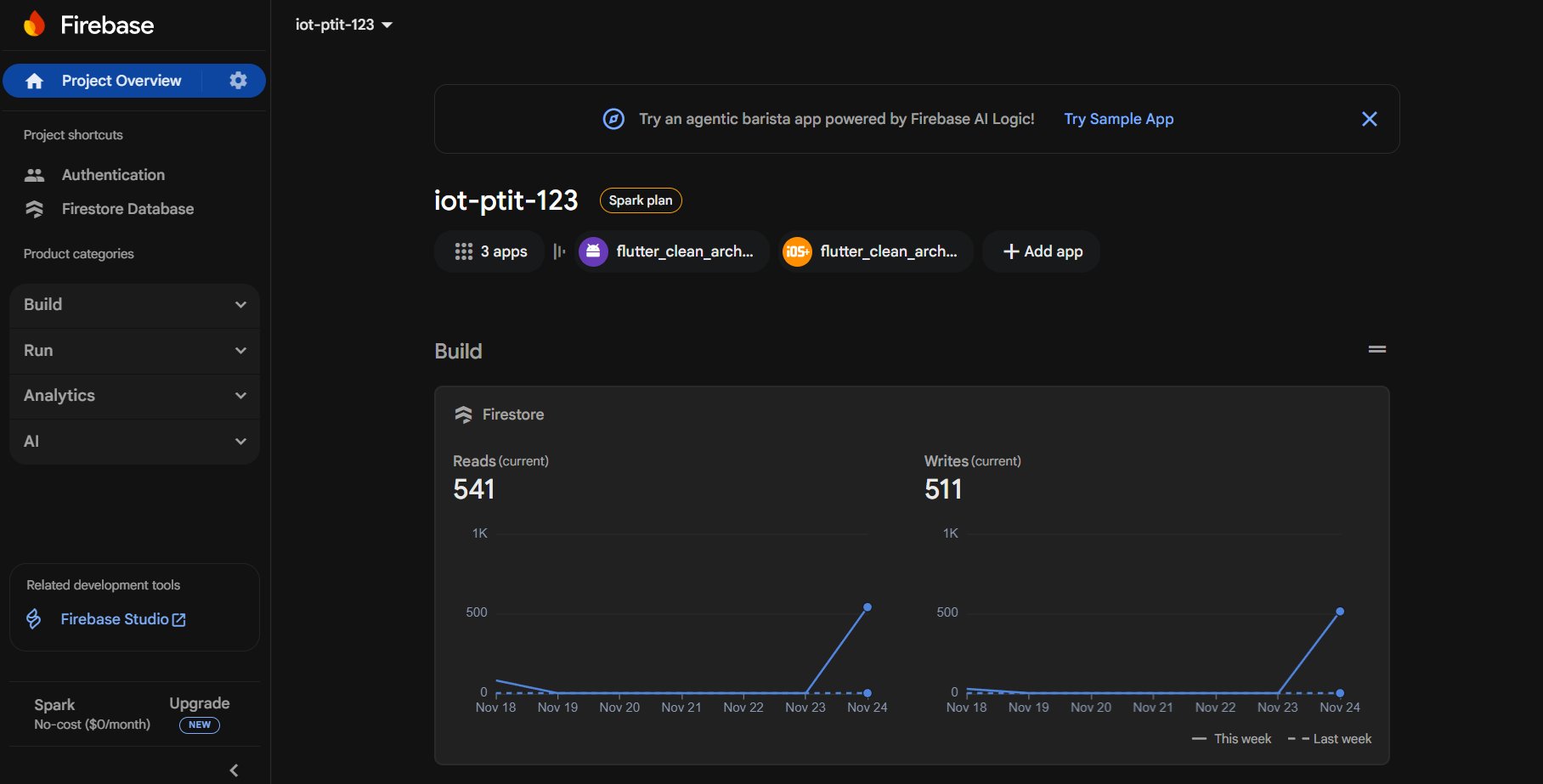
Truy cập vào firebase.google.com, đăng nhập bằng tài khoản google và nhấn vào “get started”



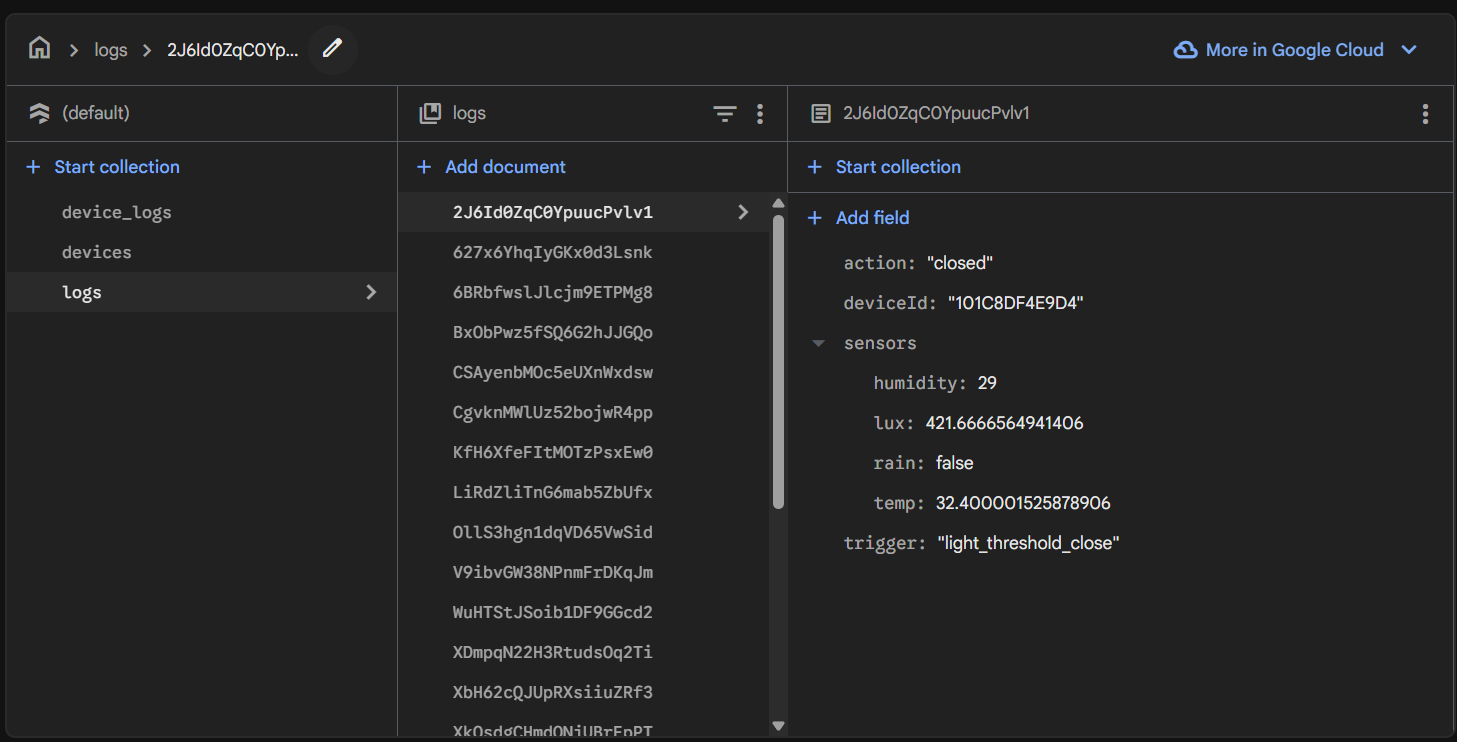
Chọn “Create a new Firebase project”, chọn tên project và tiếp tục

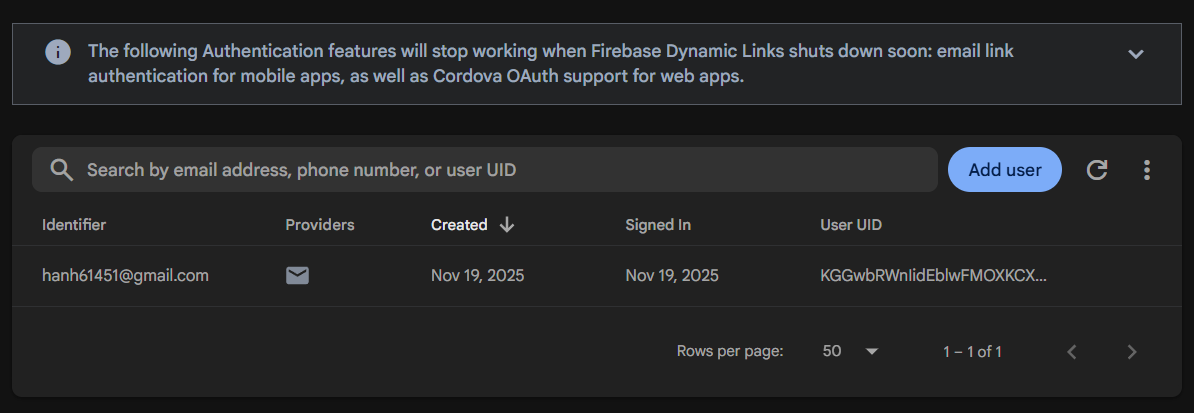


Sau khi tạo dự án, bạn sẽ được chuyển hướng đến trang bảng điều khiển dự án của mình



Với Firebase, chúng ta có thể sử dụng Firestore để lưu trữ dữ liệu về người dùng, về thông tin các thiết bị, thực hiện đọc, ghi dữ liệu log của từng thiết bị khi hoạt động đóng, mở

Về đăng nhập, đăng ký, chúng ta sử dụng Authentication có sẵn trong Firebase

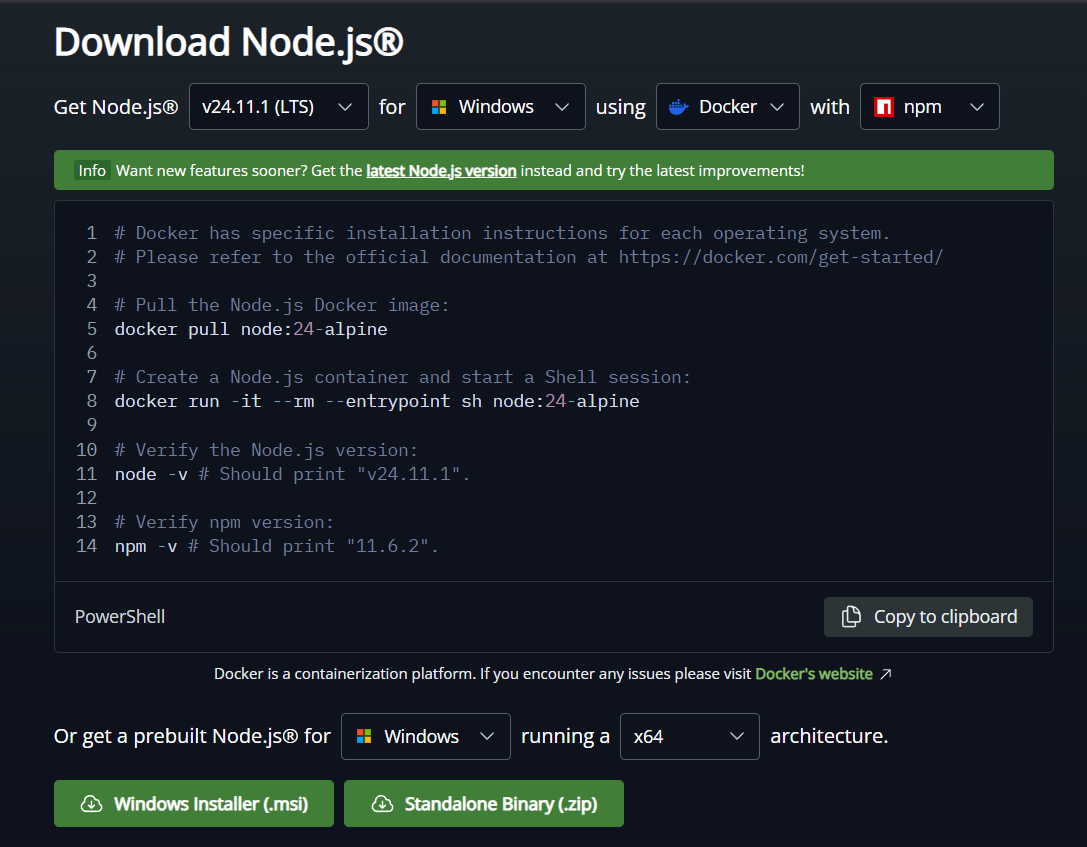


#### c. React:

React là một thư viện JavaScript phổ biến do Facebook phát triển, dùng để xây dựng giao diện người dùng (UI). React cho phép phát triển các thành phần giao diện độc lập, có thể tái sử dụng, giúp đơn giản hóa quá trình quản lý và cập nhật trạng thái của ứng dụng một cách hiệu quả. Với cơ chế "Virtual DOM", React giúp tăng tốc độ render và giảm tải cho trình duyệt, giúp ứng dụng trở nên mượt mà hơn. React cũng có một hệ sinh thái phong phú, hỗ trợ cả cho phát triển frontend và server-side rendering.

**Các bước để chạy dự án web bằng react:**

- Trước tiên, cần cài đặt nodejs và npm trên máy tính của mình nếu chưa có.



- Sau khi cài đạt node.js và npm, chạy lệnh “npm install -g create-react-app” để cải đặt base của react app trên máy tính

- Sau khi cài đạt create react app, có thể tự tạo dự án react bằng các mở terminal hoặc cmd và chạy lệnh “create-react-app my-app” để tạo ra dự án mới có tên là “my-app” với cấu trúc chuẩn

- Tiếp tục chạy “cd my-app” và “npm start” sẽ mở trình duyệt để chạy ứng dụng react hiện tại tại địa chỉ http://localhost:3000

#### d. Flutter

Flutter là bộ SDK giao diện đa nền tảng do Google phát triển, dùng ngôn ngữ Dart. Điểm mạnh của Flutter:

UI nhất quán, hiệu năng cao (render thẳng bằng engine Skia, không phụ thuộc native UI).

Hot reload giúp dev nhanh.

Một codebase cho Android, iOS, Web, Desktop (Windows/macOS/Linux).

Hệ sinh thái plugin phong phú, dễ gắn với API/h phần cứng qua Platform Channels.

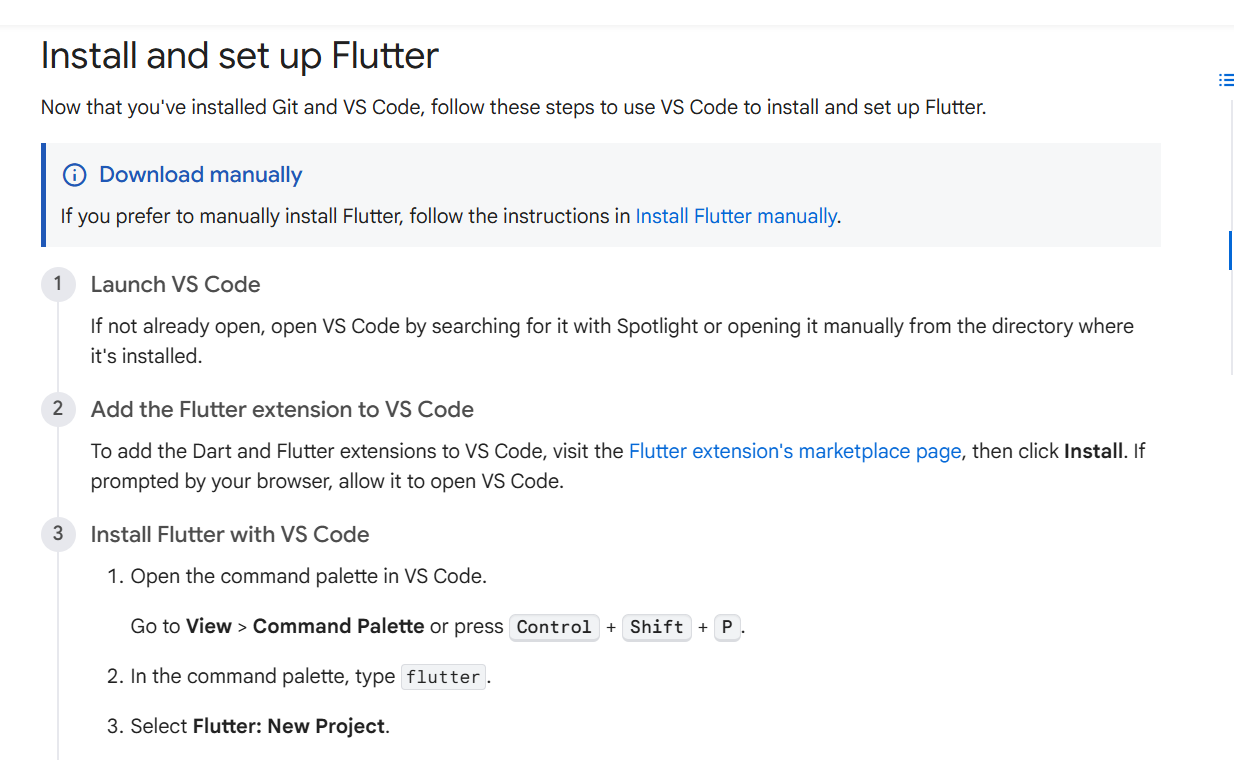
Ứng dụng IoT thường cần giao diện đẹp + realtime (dashboard, biểu đồ, điều khiển), kết nối đa giao thức (BLE, Wi-Fi, MQTT/WebSocket, HTTP), chạy trên nhiều thiết bị (điện thoại, tablet, web kiosk). Flutter đáp ứng tốt cả 3, lại có nhiều package sẵn sàng dùng.

Về cách cài đặt flutter để có thể build và chạy app mobile, nó khá là phức tạp:

Trước hết, bạn cần đảm bảo đã cài đặt các công cụ sau:

**Flutter SDK:**

Tải về phiên bản **Flutter SDK** mới nhất từ trang web chính thức của Flutter.



Giải nén tệp tin đã tải về và đặt nó vào một thư mục dễ nhớ (ví dụ: C:\src\flutter trên Windows, hoặc thư mục cá nhân trên macOS/Linux).

Thêm đường dẫn đến thư mục con bin của Flutter vào biến môi trường **PATH** của hệ thống.

**Trình Soạn Thảo Mã (IDE):**

Nên sử dụng **Visual Studio Code (VS Code)** hoặc **Android Studio** vì chúng có hỗ trợ và plugin tốt nhất cho Flutter.

**Cấu Hình Nền Tảng:**

**Android Development:** Cài đặt **Android Studio** và cấu hình **Android SDK** cùng với các công cụ nền tảng cần thiết.

**iOS/macOS Development (Chỉ trên Mac):** Cài đặt **Xcode** (cho phát triển iOS/macOS) và **CocoaPods**.

Sau khi cài đặt Flutter SDK và IDE, bạn mở **Terminal** (hoặc Command Prompt/PowerShell) và chạy lệnh “flutter doctor” sau để kiểm tra xem môi trường đã sẵn sàng chưa.

Khi môi trường đã sẵn sàng, bạn có thể tạo dự án bằng một trong hai cách:

Chạy lệnh “flutter create ten\_du\_an\_cua\_ban” với tên dự án được viết thường và gạch dưới.

Sử dụng IDE để tạo dự án.

Sau khi tạo dự án, thực hiện chạy “flutter run” để ứng dụng build và chạy trên thiết bị đã chọn

### 1.4.3 Luồng hoạt động cơ bản

#### a. Khởi tạo & kết nối

Cấp nguồn: ESP32 khởi động, kiểm tra limit switch & I/O.

Kết nối Wi-Fi: vào STA, kết nối SSID đã lưu (hoặc mở cổng cấu hình nếu chưa có).

Kết nối MQTT: kết nối broker, đăng ký cmd/<deviceId>, gửi ack/<deviceId> = online + thông tin phiên bản.

#### b. Vòng lặp cảm biến → quyết định → chấp hành

Đọc cảm biến:

BH1750 (lux) lấy trung bình N mẫu.

Raindrop (D0) với bộ đếm mưa liên tục ≥ T xác nhận (mặc định 3 giây).

DHT11: nhiệt độ/độ ẩm (phục vụ hiển thị/ghi log).

Ra quyết định (FSM):

Nếu mưa xác nhận → trạng thái CLOSING (thu).

Nếu không mưa & lux > ngưỡng nắng → OPENING (mở).

Timeout hành trình (ví dụ 15s) → STOP + phát event TIMEOUT.

Điều khiển motor:

Gửi PWM qua ENA (GPIO25), chiều quay bằng IN1/IN2; dừng mềm/hard stop khi cần.

Phát dữ liệu:

Telemetry định kỳ (5–10s): telemetry/<deviceId> chứa {lux,rain,temp,rh,state,ts}.

Event tức thời (limit, raining, timeout) lên event/<deviceId>.

#### 3. Điều khiển từ app/web

Người dùng nhấn Open/Close/Stop → app publish cmd/<deviceId> = OPEN|CLOSE|STOP.

ESP32 nhận lệnh, ACK lên ack/<deviceId> (ví dụ OPENING), đồng thời cập nhật trạng thái cho dashboard.

# CHƯƠNG 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG

## 2.1. Tổng quan

Hệ thống giàn phơi/thiết bị thu quần áo thông minh được xây dựng nhằm tự động hóa việc đóng/mở dựa trên điều kiện thời tiết thực tế và dự báo, đồng thời cho phép điều khiển từ xa qua Internet.

Hệ thống kết hợp cảm biến tại chỗ nhằm ra quyết định thông minh và chủ động hơn trong việc bảo vệ quần áo khi trời mưa hoặc thời tiết chuyển xấu.

## 2.2. Người dùng và các bên liên quan (Stakeholders)

| **Đối tượng** | **Vai trò** |
| --- | --- |
| Người dùng cuối | Điều khiển thiết bị, nhận thông báo, xem trạng thái và lịch sử |
| Kỹ sư phát triển firmware | Lập trình ESP32, xử lý cảm biến, MQTT |
| Backend/Cloud engineer | Giám sát broker MQTT, cơ sở dữ liệu, uptime |
| Frontend/App developer | Xây dựng giao diện mobile/web |

Bảng 2.2: Stakeholders

## 2.3. Mục tiêu hệ thống

* Tự động mở/thu giàn phơi dựa trên tín hiệu cảm biến tại chỗ (mưa, ánh sáng)
* Điều khiển từ xa qua MQTT và nhận phản hồi theo thời gian thực.
* Theo dõi trạng thái, lịch sử cảm biến và sự kiện.

## 2.4 Yêu cầu chức năng (Functional Requirements)

#### F1 – Điều khiển thủ công

Người dùng có thể gửi lệnh OPEN, CLOSE, STOP từ ứng dụng; thiết bị phản hồi trạng thái ngay lập tức qua MQTT.

#### F2 – Tự động theo cảm biến

Thiết bị đọc cảm biến mưa và ánh sáng:

Khi mưa thật → tự động thu.

Khi ánh sáng cao và không mưa → mở ra phơi.

#### F3 – Telemetry realtime

Gửi định kỳ các thông số:

* Lux (BH1750)
* Mưa (Raindrop)
* Nhiệt độ/độ ẩm
* Trạng thái giàn phơi

#### F5 – Cấu hình từ xa

Cho phép thay đổi: ngưỡng mưa, ngưỡng ánh sáng, thời gian timeout, chế độ tự động...

## 2.5 Yêu cầu phi chức năng (Non-functional Requirements)

#### N1 – Tính ổn định

Uptime ≥ 99%; tự reconnect Wi-Fi & MQTT.

#### N2 – Độ tin cậy

Firmware phải đảm bảo không tự động đóng mở liên tục (debounce, hysteresis).

#### N3 – Tính thời gian thực

ACK MQTT phải phản hồi trong ≤ 2 giây.

#### N4 – Bảo mật

MQTT username/password hoặc TLS.

#### N5 – Khả năng mở rộng

Hỗ trợ nhiều thiết bị.

Hỗ trợ bổ sung thêm cảm biến hoặc mô hình ML tương lai.

## 2.6 Yêu cầu phần cứng

* ESP32 DevKit (Wi-Fi + I2C + GPIO)
* Cảm biến ánh sáng BH1750
* Cảm biến mưa (LM393)
* DHT11/22 (tùy chọn)
* Driver motor L298N hoặc mô-đun H-bridge khác
* Motor kéo/đẩy
* Nguồn 12V + module hạ áp LM2596
* Limit switch hai đầu hành trình

## 2.7 Yêu cầu phần mềm

**Firmware ESP32**

* Wi-Fi manager + captive portal
* MQTT client kết nối broker
* FSM điều khiển động cơ
* Đọc cảm biến BH1750, Raindrop, DHT
* Gửi & nhận MQTT
* Xử lý ngưỡng tự động

**Backend/Cloud**

* Xử lý và lưu dữ liệu thời tiết
* Database NoSQL (Firestore)

**Frontend/App**

* Dashboard hiển thị realtime
* Cấu hình thiết bị
* Biểu đồ dữ liệu cảm biến

## 2.8 Use Cases chính

**UC1 – Điều khiển thủ công**

**UC2 – Tự động theo cảm biến tại chỗ**

**UC3 – Telemetry & Logging**

## 2.9 Ràng buộc & Giả định hệ thống

### 2.9.1 Giả định hệ thống

* Wi-Fi trong nhà ổn định trong phần lớn thời gian.
* Cơ cấu cơ khí đủ mạnh để kéo giàn phơi trong thời gian dài.
* Người dùng có nguồn điện ổn định cho thiết bị.
* Ban công có không gian lắp đặt cảm biến mưa và BH1750.

### 2.9.2 Ràng buộc kỹ thuật & môi trường

#### 2.9.2.1. Ràng buộc môi trường

* Nhiệt độ môi trường 15–55°C, độ ẩm cao → cảm biến cần chống nước/ẩm.
* Có thể gặp mưa tạt và nắng gắt → cảm biến cần che chắn.
* Motor ở khu nóng → cần timeout bảo vệ**.**

#### 2.9.2.2. Ràng buộc nguồn

* ESP32 dễ sụt áp → cần nguồn 5V ổn định và lọc nhiễu.
* Motor gây nhiễu → tách nguồn motor và nguồn MCU.

#### 2.9.2.3. Ràng buộc mạng

* Hệ thống phụ thuộc Wi-Fi → cần cơ chế tự reconnect và chế độ offline (nhờ cảm biến).
* Broker MQTT có thể tắt → thiết bị cần queue dữ liệu đơn giản.

#### 2.9.2.4. Ràng buộc phần mềm

* RAM hạn chế → tối ưu JSON, giảm kích thước packet MQTT.

**2.9.2.5. Ràng buộc cơ khí**

* Motor DC lực kéo giới hạn → không được kéo quá tải.
* Dây cuốn có thể kẹt → cần limit switch hoặc timeout.

## 2.10. Tiêu chí thành công

### 2.10.1. Hiệu năng

* **Độ trễ phản hồi lệnh:** ≤ 2 giây từ khi người dùng bấm trên App đến khi ESP32 thực thi.
* **Độ trễ cập nhật telemetry:** 5–10 giây/lần, dữ liệu không bị thiếu gói.

### 2.10.2. Độ chính xác cảm biến

* Cảm biến mưa: nhận diện mưa sai lệch không quá **±5% độ ướt bề mặt**.
* Cảm biến ánh sáng BH1750: sai số không quá **±10% giá trị lux**.

### 2.10.3. Độ tin cậy

* **Tỷ lệ truyền MQTT thành công > 98%.**
* **Hệ thống tự hoạt động liên tục ≥ 7 ngày** mà không cần reset thủ công.

### 2.10.4. Khả năng mở rộng

* Cho phép mở rộng lên **≥ 20 thiết bị** trong tương lai mà không cần thay đổi kiến trúc Cloud.

### 2.10.5. An toàn vận hành

* Không xảy ra hiện tượng **đóng/mở liên tục** khi mưa – nắng thất thường.
* Motor không vượt quá **thời gian hành trình tối đa (timeout)**.

### 2.10.6. Chi phí

* Tổng chi phí phần cứng cho 1 thiết bị ≤ **800.000 – 1000.000 VNĐ** (ESP32, cảm biến, motor, driver, khung giá đỡ).

## 2.11 Tiêu chí nghiệm thu

### 2.11.1. Nghiệm thu chức năng điều khiển

* Khi người dùng nhấn **OPEN**, giàn phơi phải **bắt đầu mở** và chuyển sang trạng thái OPENING.
* Khi nhấn **CLOSE**, giàn phơi phải **thu vào** và chuyển sang trạng thái CLOSING.
* Khi nhấn **STOP**, động cơ **phải dừng ngay lập tức**.
* Trạng thái thiết bị hiển thị trên app phải **khớp với trạng thái ESP32** trong mọi trường hợp.

### 2.11.2. Nghiệm thu chức năng tự động theo cảm biến

* Nếu cảm biến mưa xác nhận mưa liên tục theo logic debounce, thiết bị phải **tự động chuyển sang chế độ thu**.
* Nếu trời khô ráo và cường độ sáng vượt ngưỡng cài đặt, hệ thống phải **tự động mở phơi**.
* Khi người dùng tắt chế độ AUTO, **tất cả hành động tự động phải dừng lại** và chỉ chạy manual.

### 2.11.3. Nghiệm thu chức năng an toàn cơ khí

* Khi giàn phơi chạm limit switch hoặc hết hành trình, động cơ phải **tự động dừng**.
* Nếu motor bị kẹt, hệ thống phải **phát hiện timeout** và không được tiếp tục cố kéo.
* Thiết bị không được chạy motor khi nguồn không đủ điện (voltage drop – được firmware xử lý bằng dừng motor).

### 2.11.4. Nghiệm thu luồng dữ liệu và giao tiếp

* Thiết bị phải **gửi đầy đủ telemetry** (mưa, lux, temp, humidity, state) về cloud theo đúng chu kỳ (5–10s, tùy cấu hình).
* Khi app gửi lệnh, ESP32 phải **nhận đúng nội dung lệnh** (OPEN/CLOSE/STOP/AUTO/MANUAL).
* App phải luôn hiển thị **trạng thái mới nhất** sau mỗi lần nhận dữ liệu từ MQTT.

### 2.11.5. Nghiệm thu khả năng hoạt động ngoại tuyến (Offline Mode)

* Khi **mất Wi-Fi**, hệ thống vẫn phải:
  + Tự thu khi mưa.
  + Không để giàn phơi mở khi trời mưa.
* Sau khi Wi-Fi kết nối lại, hệ thống phải **đồng bộ trạng thái** và tiếp tục gửi dữ liệu bình thường.

### 2.11.6. Nghiệm thu giao diện người dùng

* App/Web phải hiển thị đúng các thông số cảm biến thời gian thực.
* Người dùng phải xem được **lịch sử hoạt động** của thiết bị.
* Có thể thay đổi cấu hình ngưỡng (lux, cảm biến mưa, timeout…) và thiết bị **phải nhận đúng cấu hình mới**.

### 2.11.7. Nghiệm thu khả năng quản lý nhiều thiết bị

* Mỗi người dùng có thể thêm/xóa nhiều thiết bị.
* App phải hiển thị **danh sách thiết bị**, với trạng thái realtime của từng cái.
* Lệnh gửi đến đúng deviceId, không nhầm lẫn giữa nhiều thiết bị.

## ****2.12. Rủi ro & đối sách****

### ****2.12.1. Rủi ro cảm biến****

| **Rủi ro** | **Nguyên nhân** | **Đối sách** |
| --- | --- | --- |
| Cảm biến mưa nhiễu | Nước đọng lâu | Debounce + xác nhận mưa 3 giây |
| BH1750 đọc sai khi nắng gắt | Bão hòa lux | Giới hạn max + filter trung bình mẫu |
| DHT11 sai số lớn | Độ ẩm cao | Chỉ dùng để hiển thị, không dùng cho quyết định chính |

Bảng 2.12.1: Rủi ro cảm biến

### ****2.12.2. Rủi ro kết nối****

| **Rủi ro** | **Nguyên nhân** | **Giải pháp** |
| --- | --- | --- |
| Mất Wi-Fi | Router lỗi | Tự reconnect, tự động chạy chế độ offline |
| MQTT mất kết nối | Cloud downtime | Retry backoff + báo trạng thái offline |

Bảng 2.12.2: Rủi ro kết nối

### ****3. Rủi ro cơ khí – động cơ****

| **Rủi ro** | **Nguyên nhân** | **Giải pháp** |
| --- | --- | --- |
| Motor kẹt | Vật cản, dây cuốn | Timeout + event TIMEOUT + tự dừng |
| Motor nóng | Quá tải | Hạn chế thời gian mở/đóng ≤15s |

Bảng 2.12.3: Rủi ro cơ khí – động cơ

## 2.12. Case Study đặc thù: Hộ gia đình chung cư – Thu quần áo thông minh

### 2.12.1. Bối cảnh thực tế

Một hộ gia đình sống tại chung cư thường phơi quần áo ở ban công. Khi có mưa bất chợt, người dùng không có mặt tại nhà và quần áo bị ướt. Họ muốn một hệ thống có thể tự thu quần áo khi mưa, mở khi nắng và vẫn cho phép điều khiển từ bất cứ đâu.

### 2.12.2. Nhu cầu người dùng

* Nhận thông báo ngay khi bắt đầu mưa.
* Tự thu giàn phơi mà không cần ở nhà.
* Có thể mở lại khi nắng trở lại.
* Điều khiển qua app bất cứ lúc nào.
* Hệ thống hoạt động yên lặng, tiết kiệm điện và ít bảo trì.

### 2.12.3. Quy trình hoạt động lý tưởng

1. Trời đang nắng → giàn phơi mở.
2. Mưa nhẹ 1–2 giây → hệ thống đợi xác nhận.
3. Nếu mưa liên tục > 3 giây → **tự động thu vào**.
4. Nếu mưa tắt, trời nắng > ngưỡng lux → **tự động mở lại**.
5. Người dùng có thể override bằng lệnh manual.

# CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

## 3.1 Tổng quan hệ thống

Hệ thống giàn phơi/mái che thông minh bao gồm 3 thành phần chính: Bộ kit phần cứng (ESP32, cảm biến, động cơ), Nền tảng đám mây (Firebase/MQTT Broker), và Ứng dụng điều khiển (App Mobile/Web). Cụ thể:

**Vi mạch điều khiển, cảm biến và cơ cấu chấp hành (Hardware Layer):**

* Bộ vi điều khiển trung tâm là ESP32 có nhiệm vụ thu thập dữ liệu môi trường từ cảm biến mưa (Raindrop sensor) và cảm biến ánh sáng (BH1750) để xác định điều kiện thời tiết thực tế.
* Dựa trên dữ liệu cảm biến hoặc lệnh điều khiển nhận được từ Cloud, ESP32 sẽ điều khiển động cơ DC thông qua mạch cầu H (L298N) để thực hiện thao tác cơ học: Kéo giàn phơi ra (khi trời nắng) hoặc Thu giàn phơi vào (khi trời mưa).
* Dữ liệu trạng thái (đang mở/đang đóng, thông số ánh sáng, trạng thái mưa) được gửi vào thiết bị theo chu kỳ 2 giây 1 lần, đồng thời sẽ gửi các chỉ số lên server khi nhận được tín hiệu đóng mở.

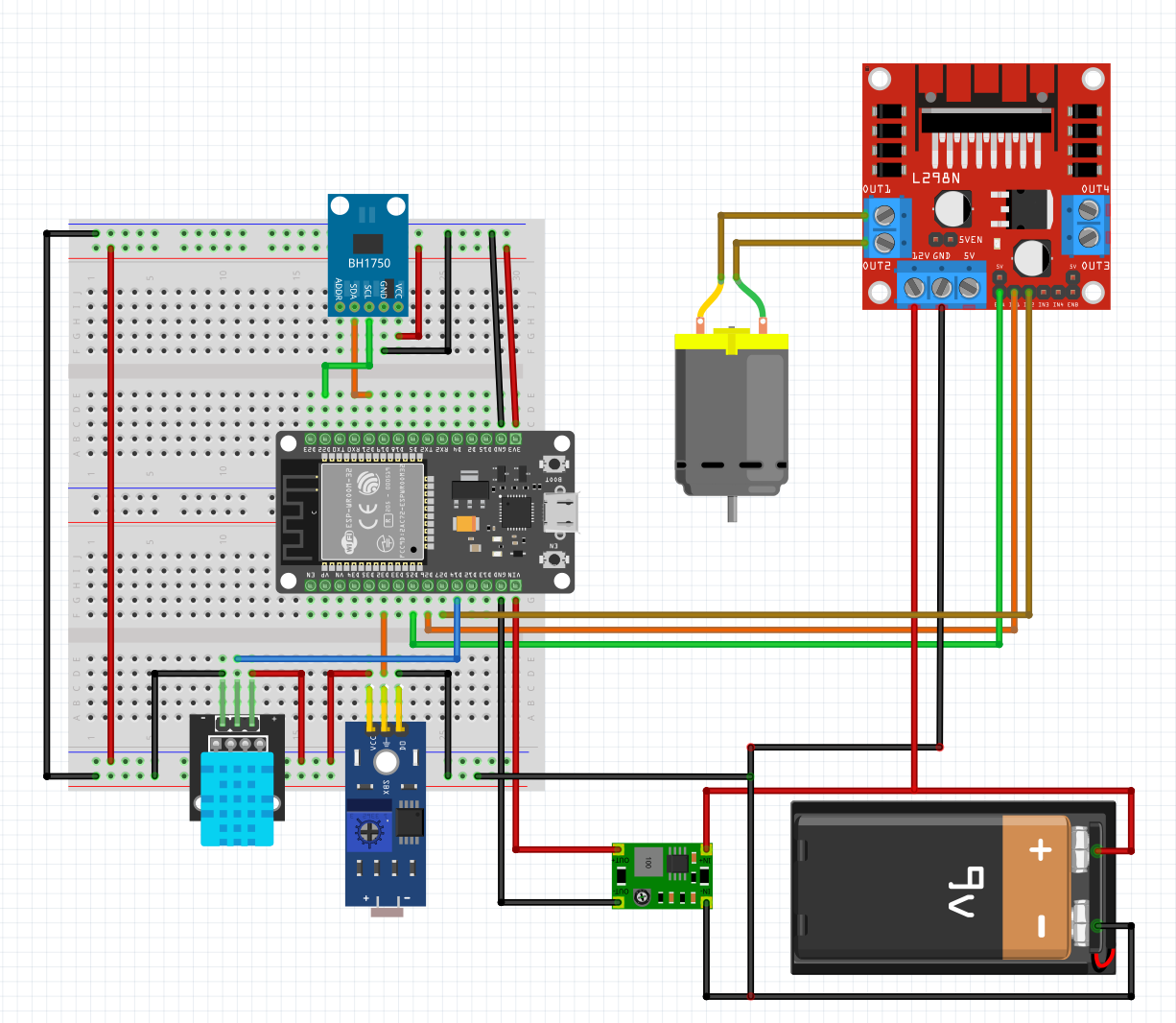
**Nền tảng đám mây (Cloud Layer - Firebase/MQTT):**

* Đóng vai trò là trung tâm lưu trữ và trung chuyển dữ liệu thời gian thực.
* Mọi dữ liệu từ cảm biến và trạng thái của giàn phơi gửi từ ESP32 sẽ được cập nhật và lưu trữ tại đây.
* Đồng thời, Cloud đóng vai trò cầu nối để nhận lệnh từ ứng dụng người dùng và đồng bộ trạng thái xuống ESP32 ngay lập tức để thiết bị thực thi.

**Giao diện người dùng (Application Layer - Web/Mobile App):**

* Người dùng giám sát hệ thống thông qua ứng dụng (được xây dựng trên Flutter hoặc Web React). Giao diện hiển thị trực quan các thông số môi trường (cường độ sáng Lux, trạng thái mưa) và vị trí hiện tại của giàn phơi.
* Người dùng có thể gửi các lệnh điều khiển thủ công (Manual Mode) như: Kéo ra, Thu vào, hoặc Dừng khẩn cấp. Các lệnh này sẽ được gửi tới Cloud và chuyển tiếp xuống thiết bị.
* Ngoài ra, ứng dụng còn hỗ trợ xem lại lịch sử hoạt động hoặc thiết lập các ngưỡng tự động (ví dụ: ngưỡng ánh sáng bao nhiêu thì tự động phơi).

## 3.2. Thiết kế sơ đồ mạch



Ảnh 3.2.1: Sơ đồ mạch

**Mô tả thiết kế chi tiết:**

*1. Khối nguồn (Power Supply)*

* Nguồn động cơ: Sử dụng nguồn DC 9V bên ngoài cấp trực tiếp vào chân +12V và GND của module L298N để đảm bảo đủ công suất kéo tải cho động cơ.
* Nguồn cảm biến: Các cảm biến (BH1750, Raindrop, DHT11) sử dụng nguồn 3.3V được lấy từ chân 3V3 của bo mạch ESP32.
* Mass (GND): Tất cả các chân GND của nguồn 9V, nguồn ESP32, module L298N và các cảm biến phải được nối chung (common ground) để đảm bảo tín hiệu điều khiển hoạt động chính xác.

2*. Kết nối giữa ESP32 và Driver động cơ L298N:*

Module L298N được sử dụng để điều khiển chiều quay (thu/kéo giàn phơi) và tốc độ động cơ.

* Chân ENA (Enable A): Kết nối với GPIO 25. Chân này dùng để điều khiển tốc độ động cơ thông qua tín hiệu PWM (nếu cần điều chỉnh tốc độ) hoặc bật/tắt động cơ.
* Chân IN1: Kết nối với GPIO 26.
* Chân IN2: Kết nối với GPIO 27.
  + Khi IN1 High / IN2 Low: Động cơ quay thuận (Ví dụ: Kéo giàn phơi ra).
  + Khi IN1 Low / IN2 High: Động cơ quay nghịch (Ví dụ: Thu giàn phơi vào).
  + Khi IN1 Low / IN2 Low: Động cơ dừng.

*3. Kết nối giữa ESP32 và Cảm biến mưa (Raindrop Sensor)*

* VCC: Nối với chân 3.3V của ESP32.
* GND: Nối với chân GND của ESP32.
* AO (Analog Output): Kết nối với GPIO 32. ESP32 sẽ đọc giá trị Analog từ chân này để xác định mức độ mưa (hoặc dùng chân DO nếu chỉ cần xác định có mưa/không mưa, nhưng GPIO 32 hỗ trợ ADC nên đọc Analog sẽ chính xác hơn cho việc đặt ngưỡng).

*4. Kết nối giữa ESP32 và Cảm biến ánh sáng (BH1750)*

Sử dụng giao tiếp I2C để đọc cường độ ánh sáng (Lux).

* VCC: Nối với chân 3.3V.
* GND: Nối với chân GND.
* SDA (Serial Data): Kết nối với GPIO 21.
* SCL (Serial Clock): Kết nối với GPIO 22.

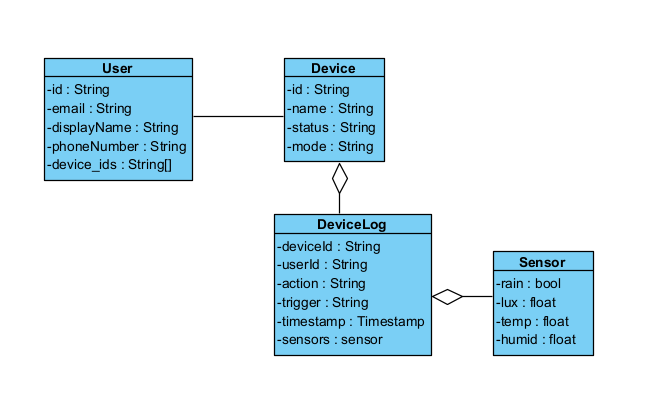
*5. Kết nối giữa ESP32 và Cảm biến nhiệt độ - độ ẩm (DHT11)*

* VCC: Nối với chân 3.3V.
* GND: Nối với chân GND.
* DATA: Kết nối với GPIO 14 để truyền nhận dữ liệu nhiệt độ và độ ẩm môi trường.

Hoạt động tổng quát của mạch:

1. Thu thập dữ liệu: ESP32 liên tục đọc dữ liệu từ các chân GPIO tương ứng: nhiệt độ/độ ẩm từ DHT11 (GPIO 14), cường độ sáng từ BH1750 (qua I2C GPIO 21, 22), và tín hiệu mưa từ chân GPIO 32.
2. Xử lý: Vi điều khiển so sánh dữ liệu thu được với các ngưỡng cài đặt (ví dụ: nếu tín hiệu mưa < ngưỡng khô ráo hoặc ánh sáng < ngưỡng tối).
3. Điều khiển chấp hành: Dựa trên kết quả xử lý hoặc lệnh từ App (qua Wifi), ESP32 xuất tín hiệu logic ra các chân GPIO 25, 26, 27 để kích hoạt module L298N, từ đó điều khiển động cơ quay đúng chiều mong muốn.

## 3.3. Thiết kế cơ sở dữ liệu



Ảnh 3.2.2: Cơ sở dữ liệu

**1. User (Người dùng)**

* id (String): Mã định danh duy nhất (UID).
* email (String): Email đăng nhập.
* displayName (String): Tên hiển thị.
* phoneNumber (String): Số điện thoại.
* devices (List<String>): Danh sách ID các thiết bị thuộc quyền quản lý.

**2. Device (Thiết bị - Giàn phơi)** Đại diện cho phần cứng, chỉ chứa thông tin định danh và trạng thái vận hành hiện tại.

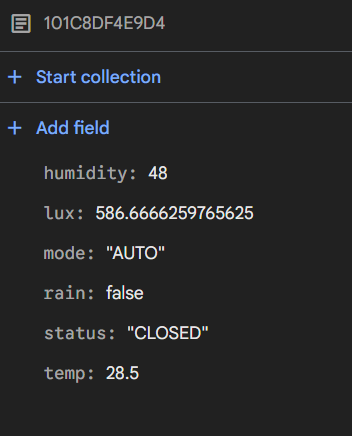
* id (String): Mã định danh thiết bị (ví dụ: "ESP32\_001").
* name (String): Tên thiết bị (ví dụ: "Giàn phơi tầng 2").
* status (String): Trạng thái hiện tại (OPENING, CLOSING, STOPPED, OPENED, CLOSED).
* mode (String): Chế độ hoạt động (AUTO/MANUAL).

**3. DeviceLog (Nhật ký hoạt động)** Lưu trữ lịch sử tác động và **dữ liệu môi trường** tại thời điểm đó.

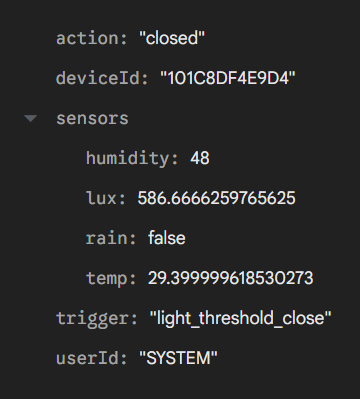
* deviceId (String): ID thiết bị.
* userId (String): ID người dùng (hoặc "SYSTEM" nếu tự động).
* action (String): Hành động (opened, closed, stopped).
* trigger (String/Enum): Nguyên nhân (manual, rain\_detected, light\_threshold).
* timestamp (Timestamp): Thời gian thực hiện.
* **sensors (Map/Object)**: Dữ liệu cảm biến tại thời điểm ghi log.
  + rain (Boolean): Có mưa hay không.
  + lux (Float): Cường độ sáng.
  + temp (Float): Nhiệt độ.
  + humidity (Float): Độ ẩm.

**Quan hệ giữa các thực thể:**

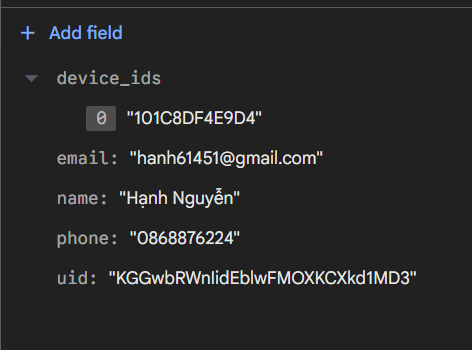
* Mỗi user quản lý nhiều devices
* Mỗi device sẽ có nhiều deviceLogs mỗi khi thông báo
* Biểu diễn dưới dạng JSON ở firebase



Ảnh 3.3.1: Cấu trúc devices

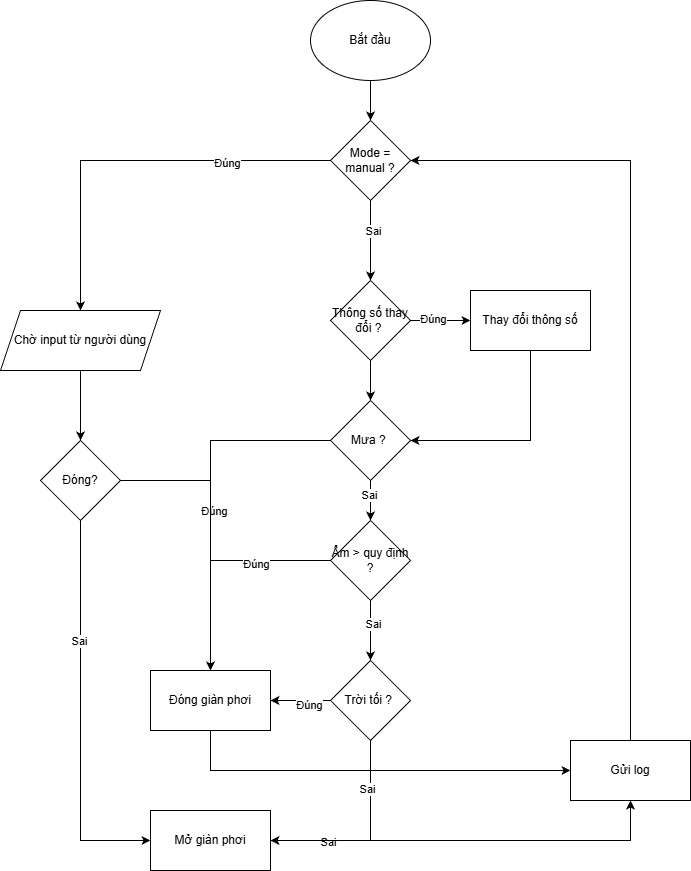


Ảnh 3.3.2: Cấu trúc Logs



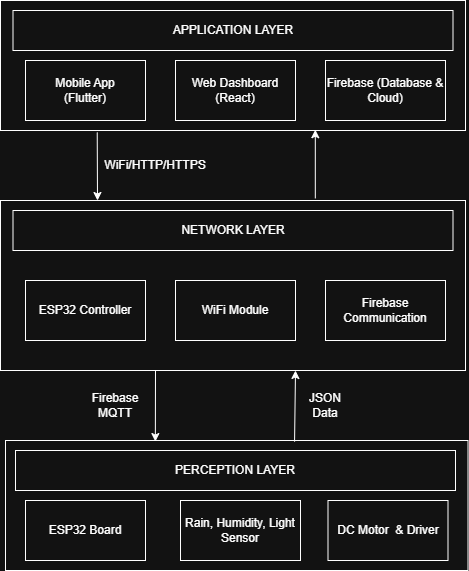
Ảnh 3.3.3: Cấu trúc users

## 3.4. Lưu đồ thuật toán



Ảnh 3.2.5: Lưu đồ thuật toán

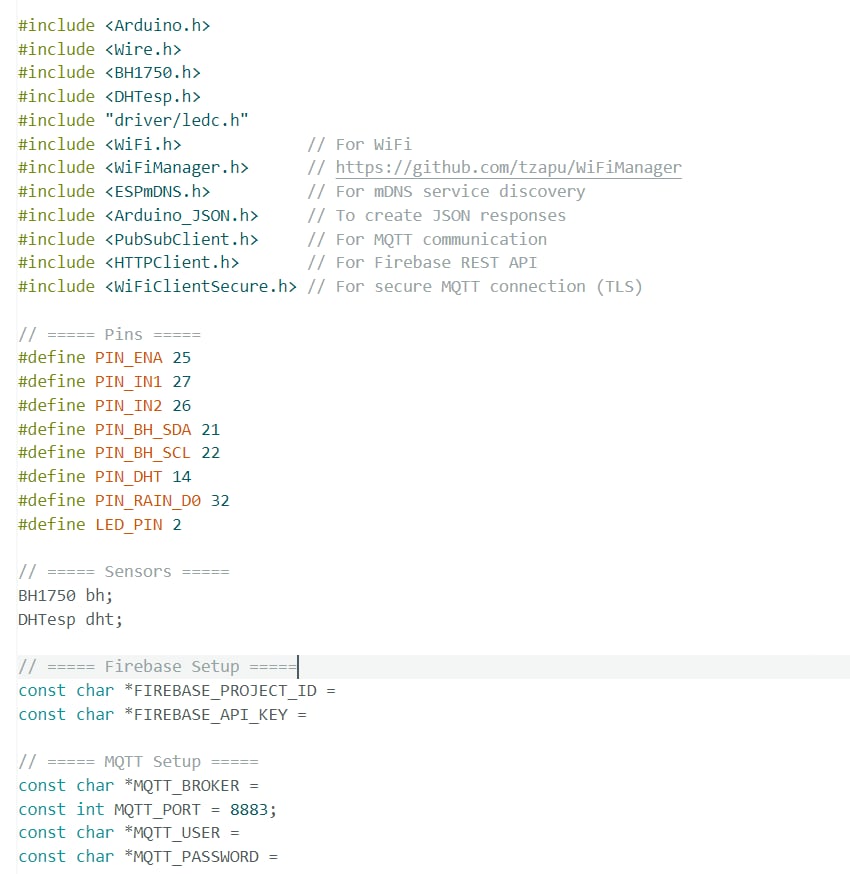
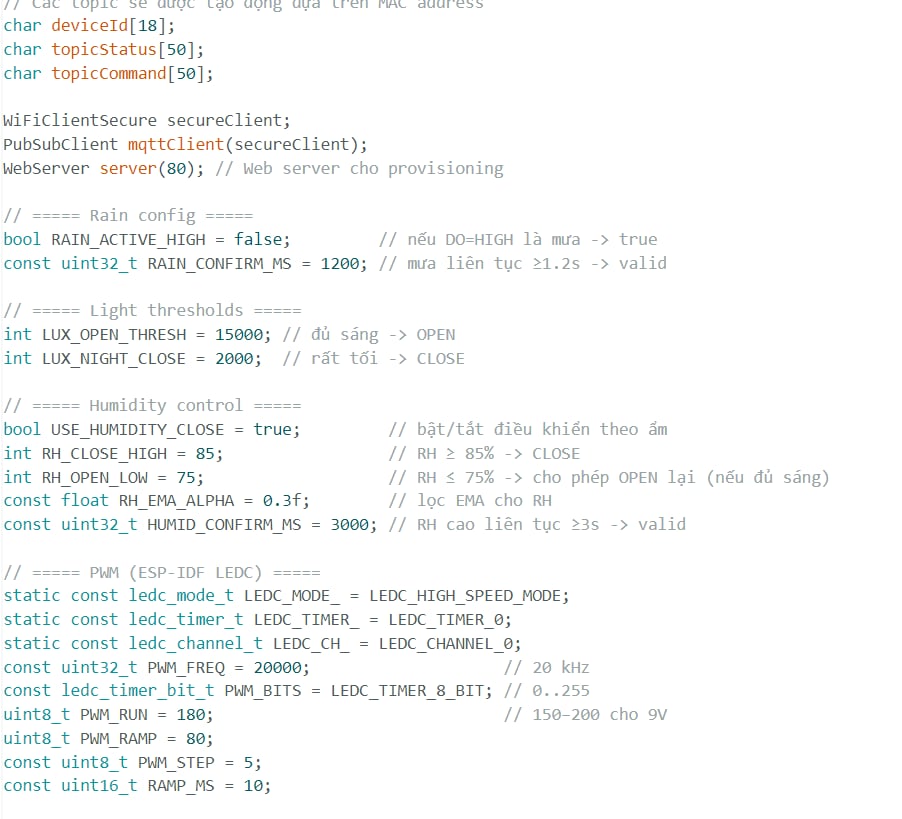
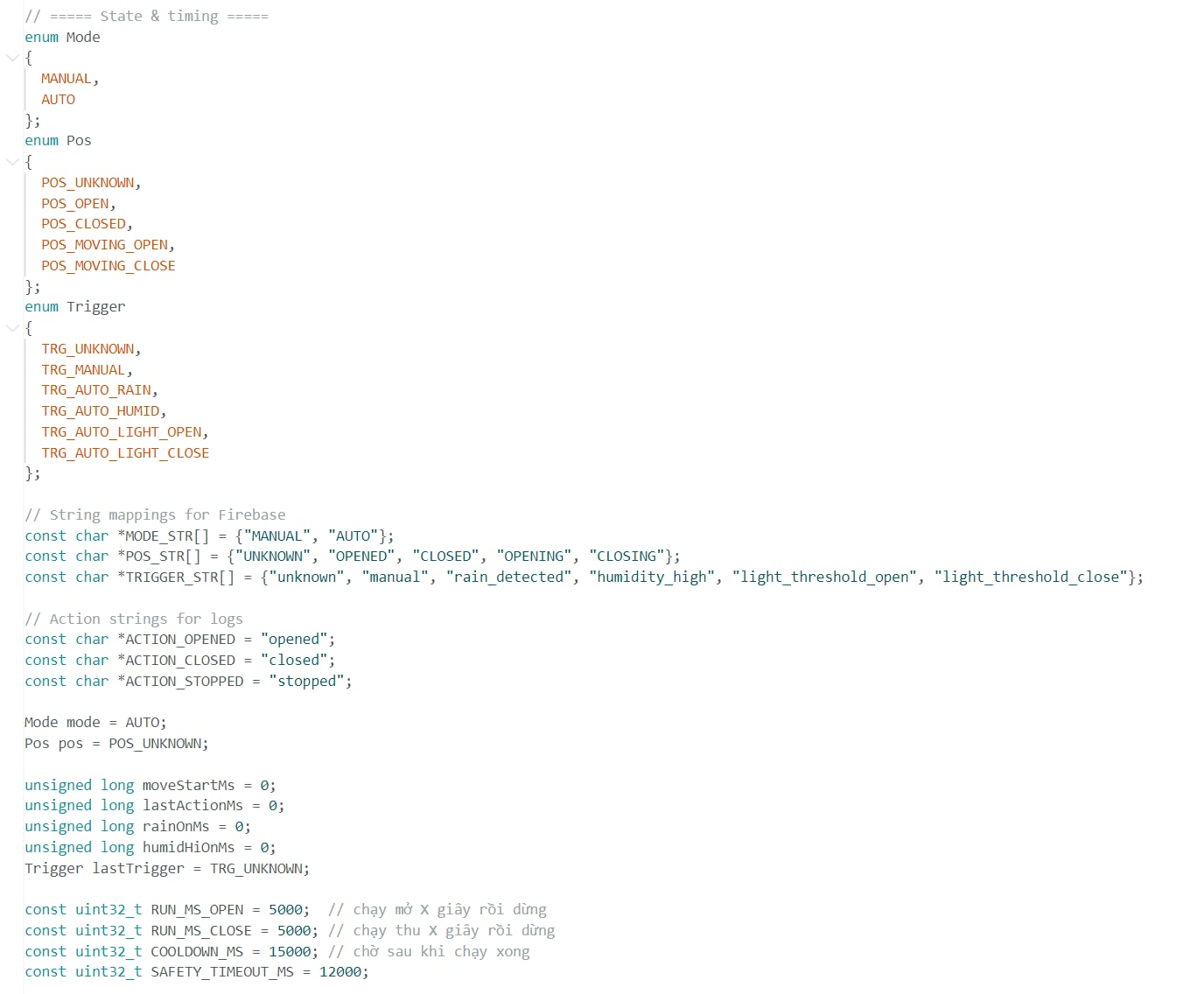
## 3.5. Đặc tả cấp độ IOT:



Hình 3.5: Đặc tả cấp độ IoT

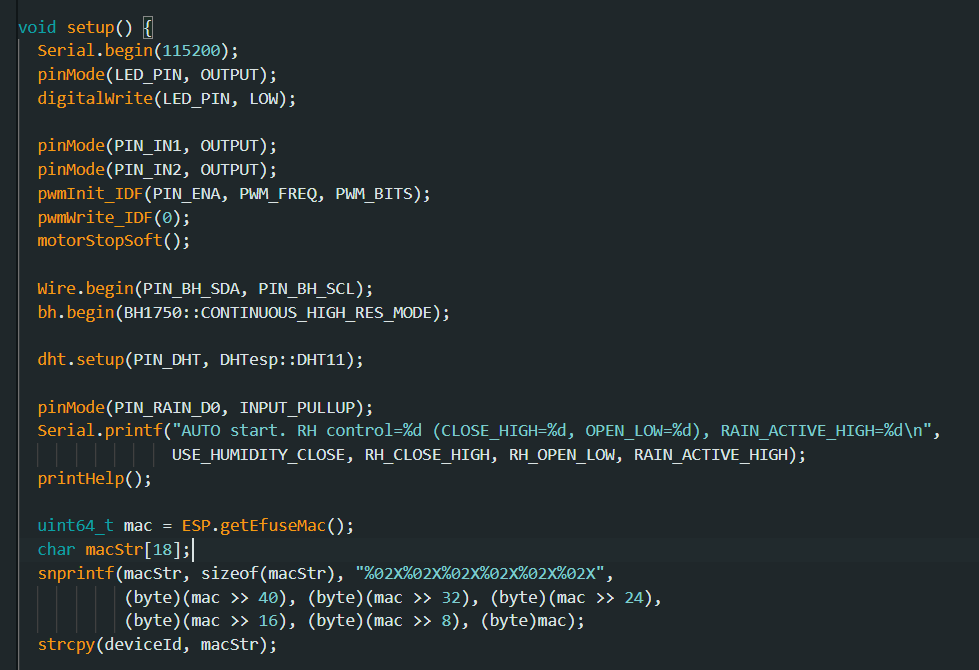
## 3.6. Chương trình điều khiển

Khai báo thư viện và các hằng số cần thiết: Khai báo các thứ viện Wifi, firebase, mqtt, định nghĩa các chân cảm biến cần thiết.

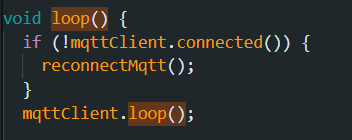
Hàm setup(): dùng để thiết lập các thông số cơ bản của hệ thống:

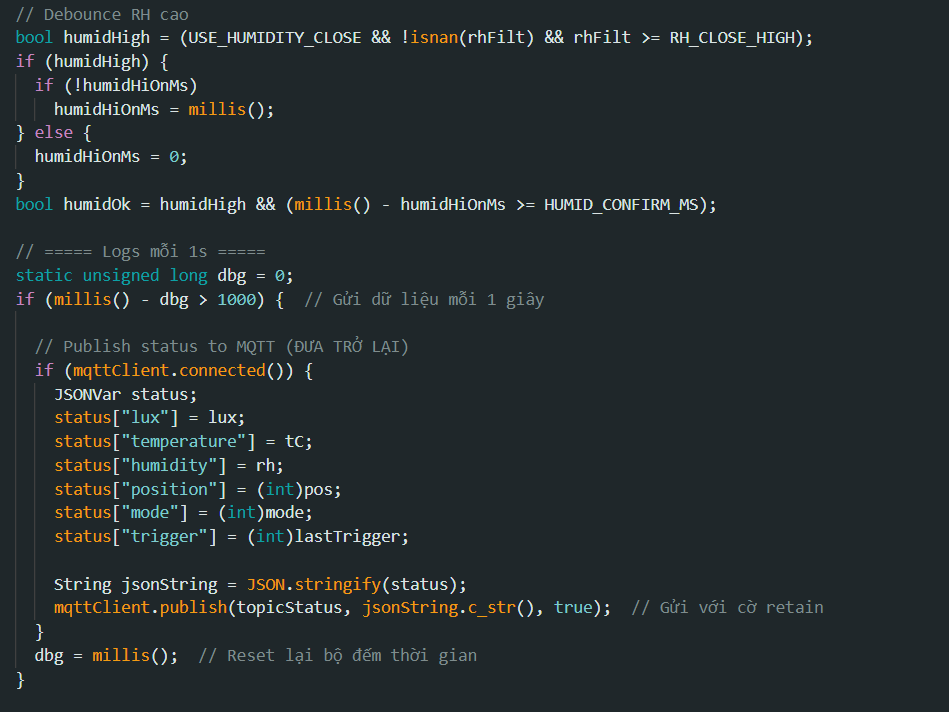
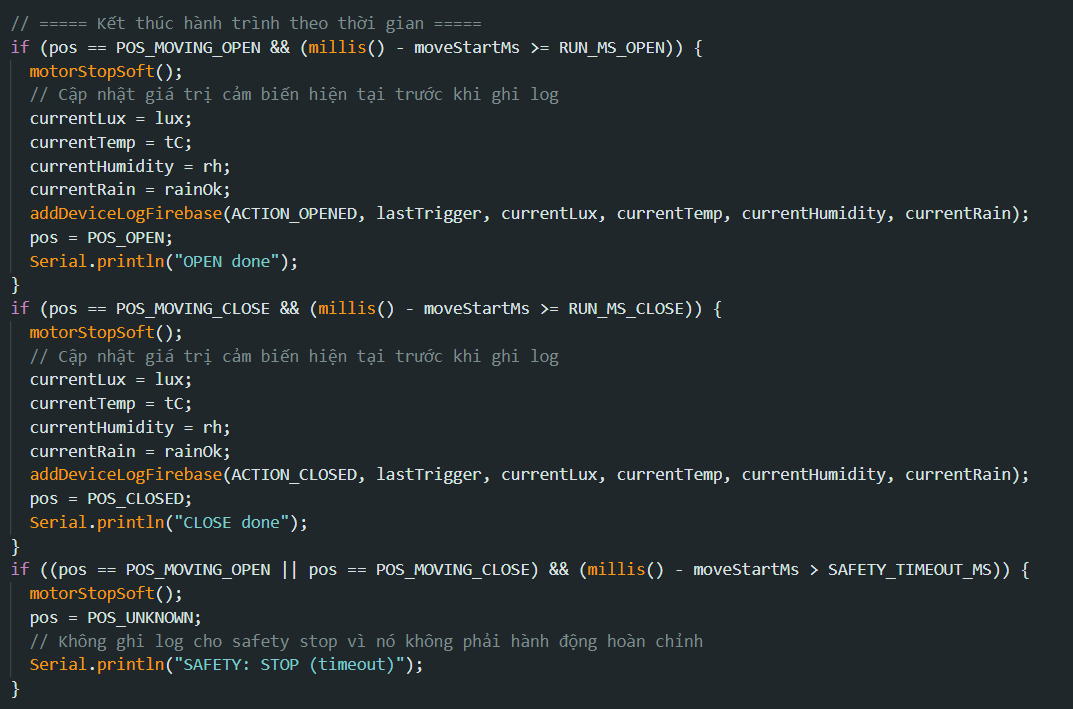
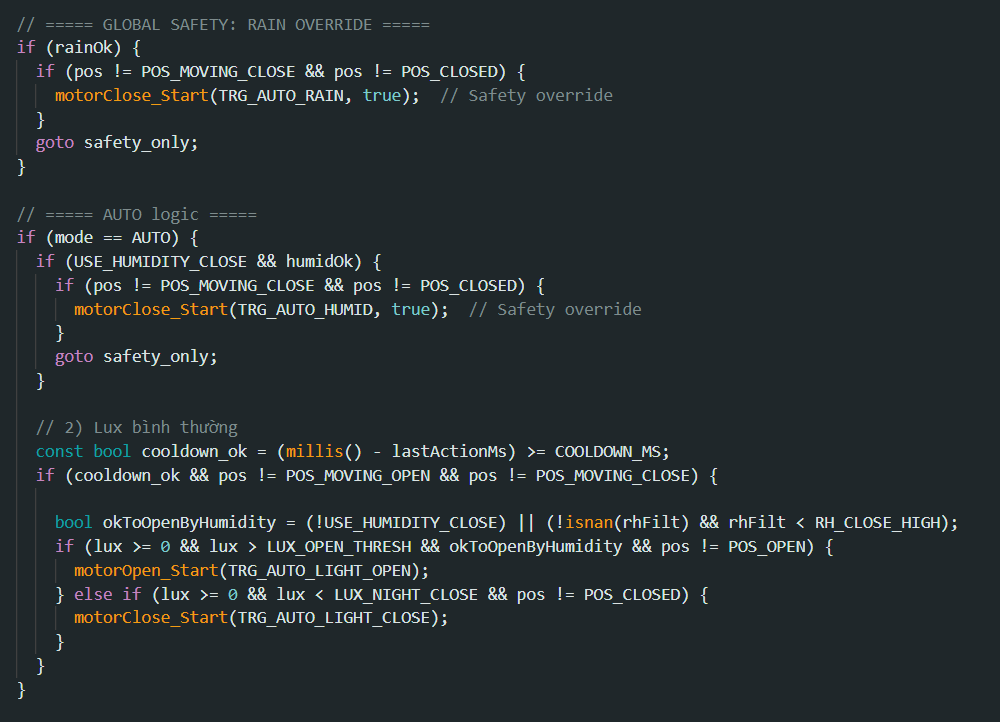
* Hệ thống khởi tại kết nối wifi, mqtt và firebase
* Cài đặt các chân đầu vào và đầu ra cho cảm biến
* Tạo id của thiết bị bằng cách lấy địa chỉ MAC hiện tại

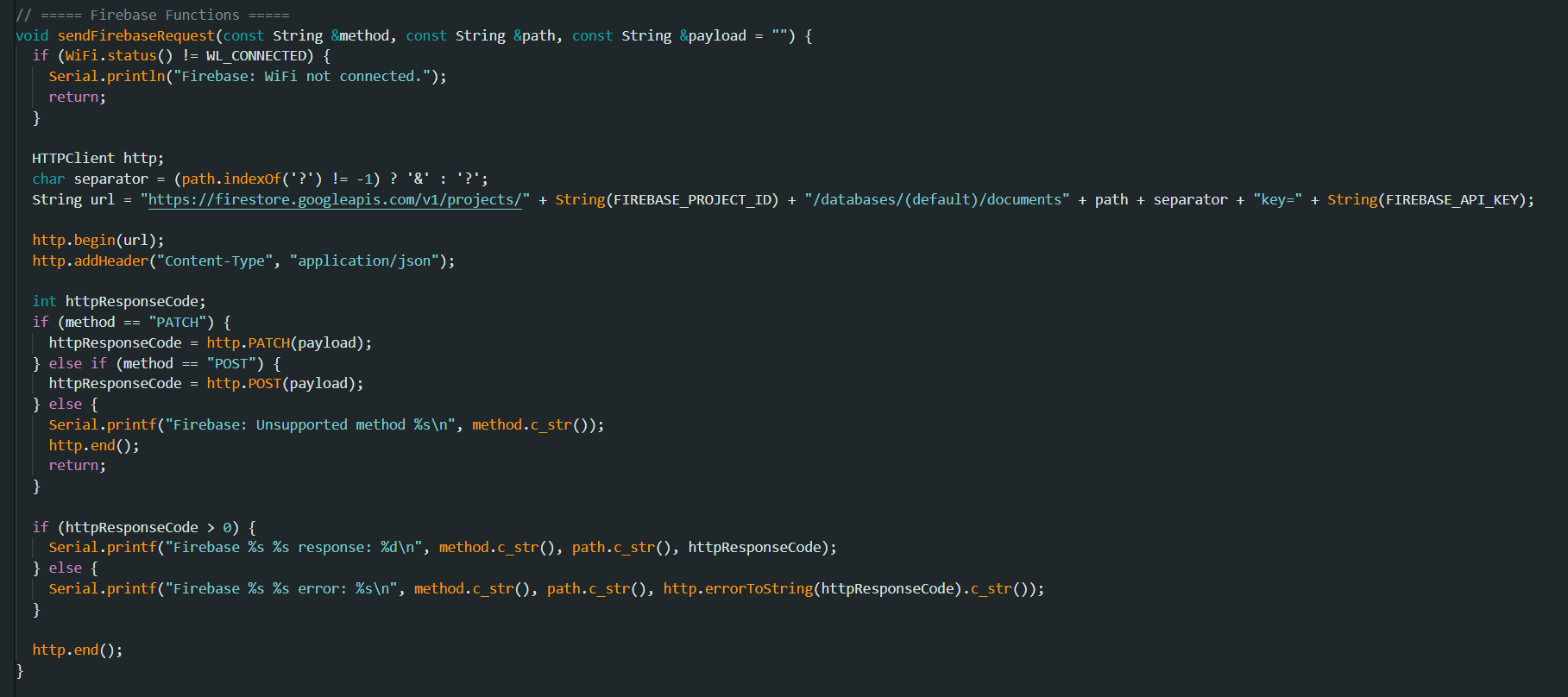
Hàm loop(): thực hiện lặp lại các công việc sau  
- Kiểm tra mqtt còn kết nối hay không, nếu không thì thực hiện kết nối lại

* Đọc giá trị từ các cảm biến
* Gửi dữ liệu từ cảm biến và trạng thái động cơ thẳng về thiết bị điều khiển
* Nhận tín hiệu từ thiết bị điều khiển và xử lý
* Tự động nhận diện trạng thái của các cảm biến, nếu khớp với yêu cầu và ở trạng thái tự động thì sẽ thực hiện đóng/mở động cơ

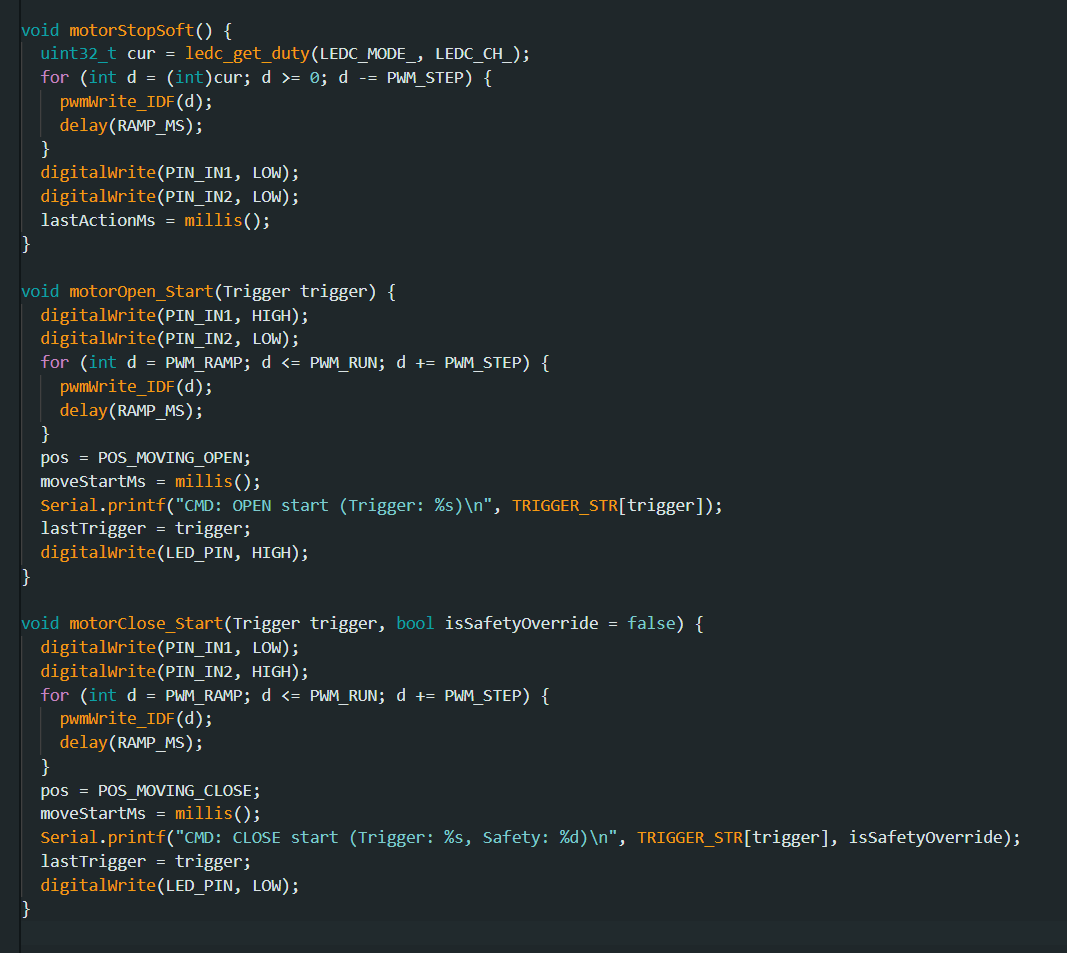
  

Các hàm helper khác:

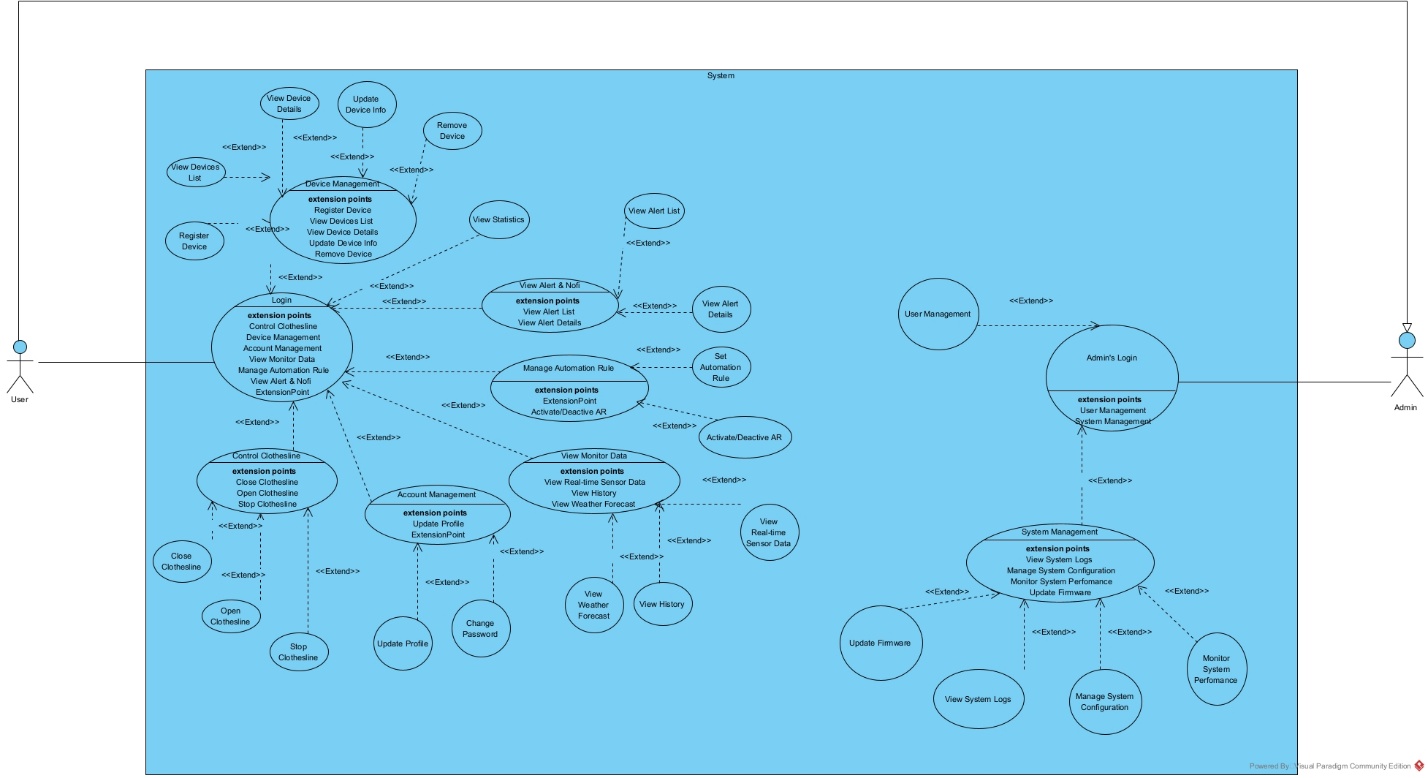
* Hàm để kết nối và cấu hình mqtt 
* Các hàm liên quan đến firebase như gửi log, đăng ký thiết bị, ...
*  



Hàm điều khiển động cơ

* 

## 3.7: Sơ đồ UseCase



Sơ đồ usecase tổng quan toàn hệ thống

***1. Chức năng của User (Người dùng)***

Người dùng đóng vai trò là người vận hành trực tiếp, giám sát thiết bị và cấu hình các hoạt động cá nhân hóa.

* **Đăng nhập/Đăng ký:** Người dùng cần đăng nhập để truy cập vào các chức năng bên trong.
* **Quản lý thiết bị (Device Management):**
  + Xem danh sách các thiết bị đang sở hữu (View Devices List).
  + Xem chi tiết thông số (View Device Details), cập nhật thông tin (Update Device Info)
  + Xem thống kê hoạt động của thiết bị (View Statistics).
* **Điều khiển giàn phơi (Control Clothesline):**
  + Thực hiện các lệnh điều khiển thủ công bao gồm: Đóng giàn phơi (Close), Mở giàn phơi (Open), và Dừng hoạt động (Stop).
  + Chức năng này tương ứng với yêu cầu cho phép người dùng bấm nút trên app/web để điều khiển thủ công.
* **Giám sát dữ liệu (View Monitor Data):**
  + Xem dữ liệu cảm biến thời gian thực (View Real-time Sensor Data) như nhiệt độ, độ ẩm, ánh sáng.
  + Xem lịch sử hoạt động và dữ liệu quá khứ (View History).
  + Xem dự báo thời tiết (View Weather Forecast) để chủ động phơi phóng.
* **Quản lý luật tự động (Manage Automation Rule):**
  + Thiết lập các luật tự động (Set Automation Rule), ví dụ như cài đặt ngưỡng ánh sáng hoặc mưa để thiết bị tự hành động.
  + Kích hoạt hoặc vô hiệu hóa các luật này (Activate/Deactivate AR).
* **Quản lý tài khoản (Account Management):**
  + Cập nhật thông tin cá nhân (Update Profile) và thay đổi mật khẩu (Change Password).
* **Xem thông báo (View Alert & Noti):**
  + Xem danh sách các cảnh báo (View Alert List) và chi tiết cảnh báo (View Alert Details) từ hệ thống gửi đến.

**2. Chức năng của Admin (Quản trị viên)**

Admin chịu trách nhiệm quản lý vận hành tổng thể, đảm bảo hệ thống hoạt động ổn định và cập nhật phần mềm.

* **Đăng nhập quản trị (Admin's Login):** Cổng đăng nhập riêng dành cho quyền quản trị.
* **Quản lý người dùng (User Management):** Admin có quyền quản lý thông tin của các người dùng trong hệ thống (như thêm, sửa, xóa hoặc phân quyền).
* **Quản lý hệ thống (System Management):**
  + **Xem nhật ký hệ thống (View System Logs):** Theo dõi các log hoạt động để phát hiện lỗi hoặc kiểm tra lịch sử vận hành.
  + **Quản lý cấu hình hệ thống (Manage System Configuration):** Thiết lập các thông số kỹ thuật chung cho toàn hệ thống.
  + **Giám sát hiệu năng (Monitor System Performance):** Theo dõi trạng thái server, broker MQTT, database để đảm bảo uptime.

# CHƯƠNG 4: KIỂM THỬ VÀ KẾT QUẢ

## 4.1. Kiểm thử ứng dụng di động

### 4.1.1. Chức năng đăng nhập

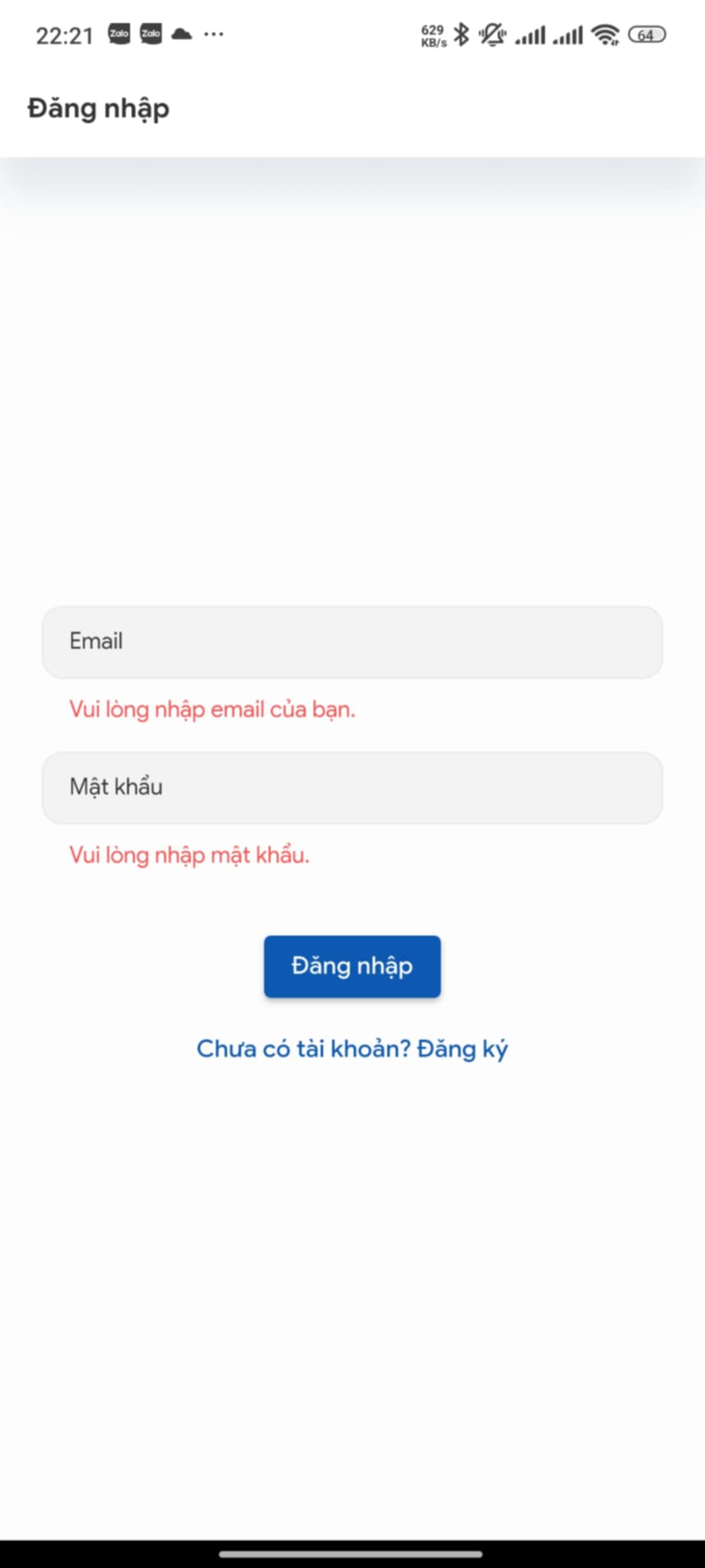
#### 4.1.1.1: Giao diện đăng nhập



Hình 4.1.1.1: Giao diện đăng nhập

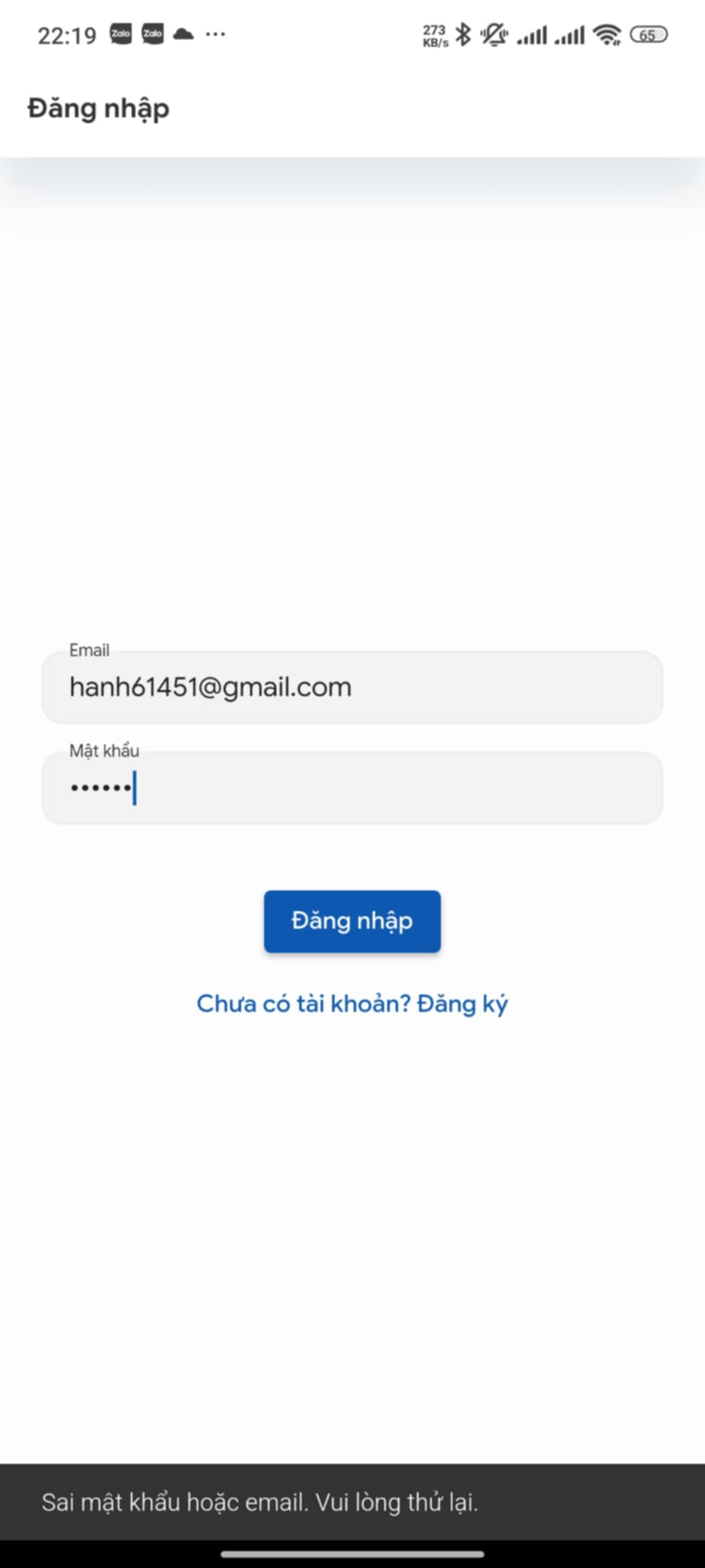
#### 4.1.1.2: Kiểm tra các trường hợp đăng nhập

*TH1: Không nhập tên đăng nhập hoặc mật khẩu nhưng bấm đăng nhập*



*Hình 4.1.1.2: Không nhập tên đăng nhập hoặc mật khẩu nhưng bấm đăng nhập*

*TH2: Nhập sai mật khẩu hoặc email*



Hình 4.1.1.3: Nhập sai mật khẩu hoặc email

*TH3: Đăng nhập thành công*



Hình 4.1.1.4: Đăng nhập thành công, vào trang chọn thiết bị

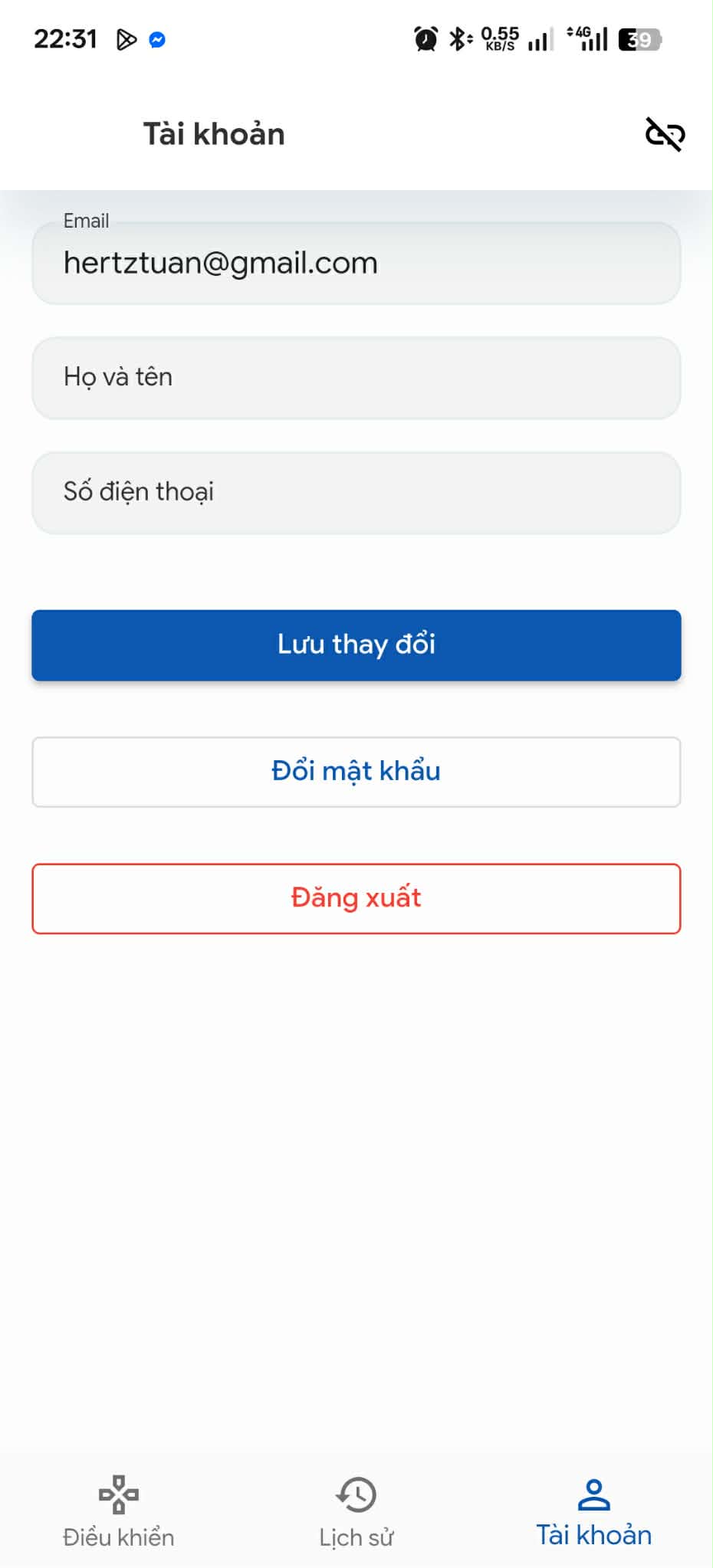
### 4.1.2: Chức năng đăng ký

#### 4.1.2.1: Giao diện trang đăng ký



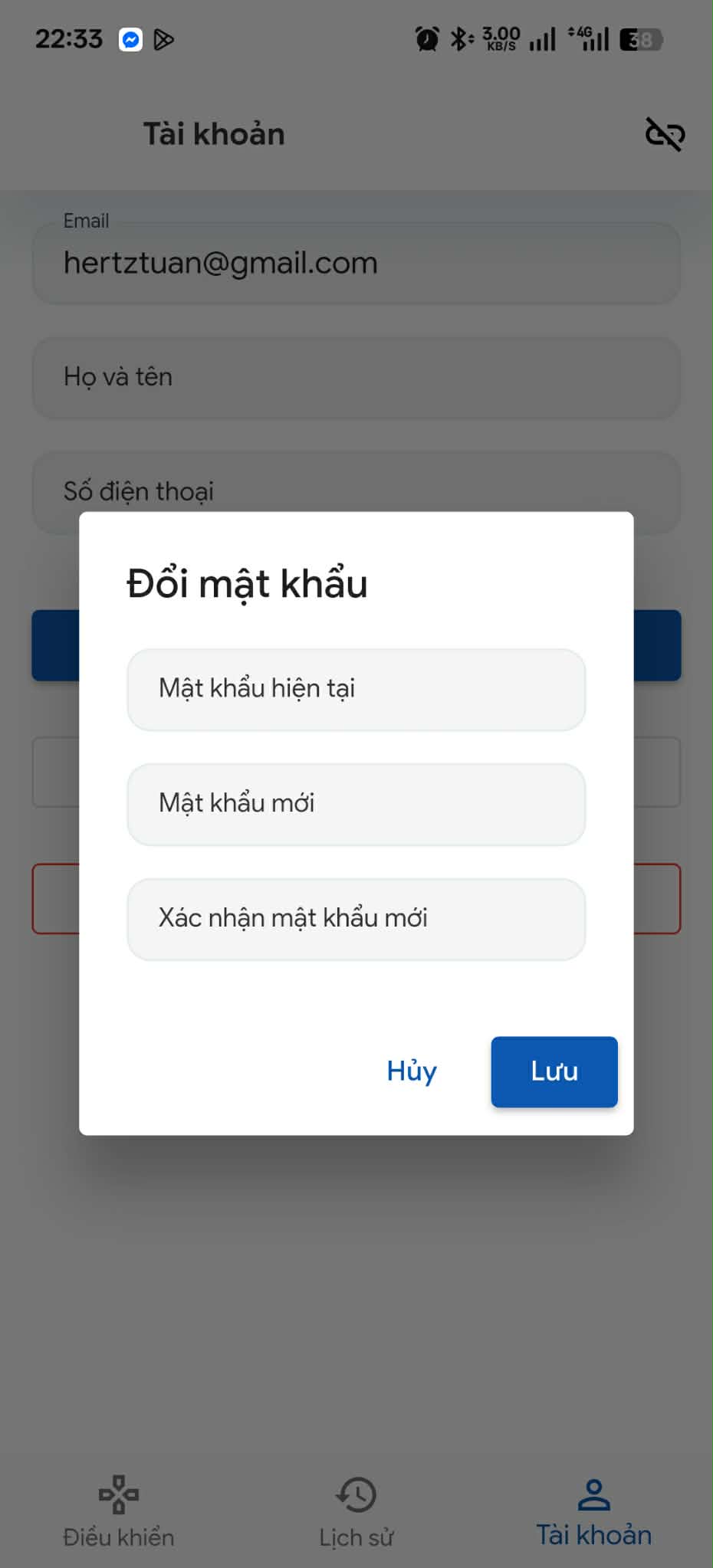
Hình 4.1.2.1: Giao diện trang đăng ký

### 4.1.3: Chức năng đăng xuất

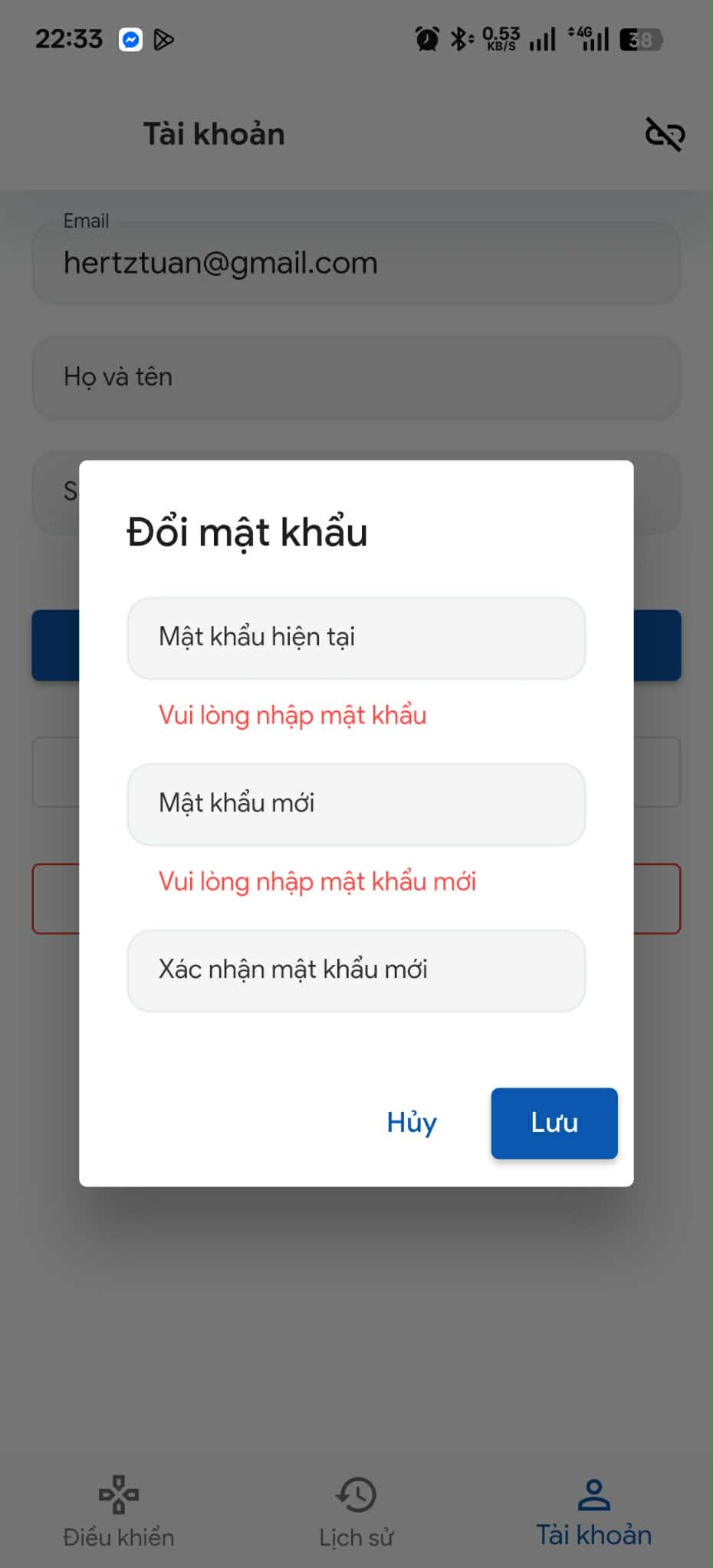


Hình 4.1.3.1: Giao diện chức năng đăng xuất

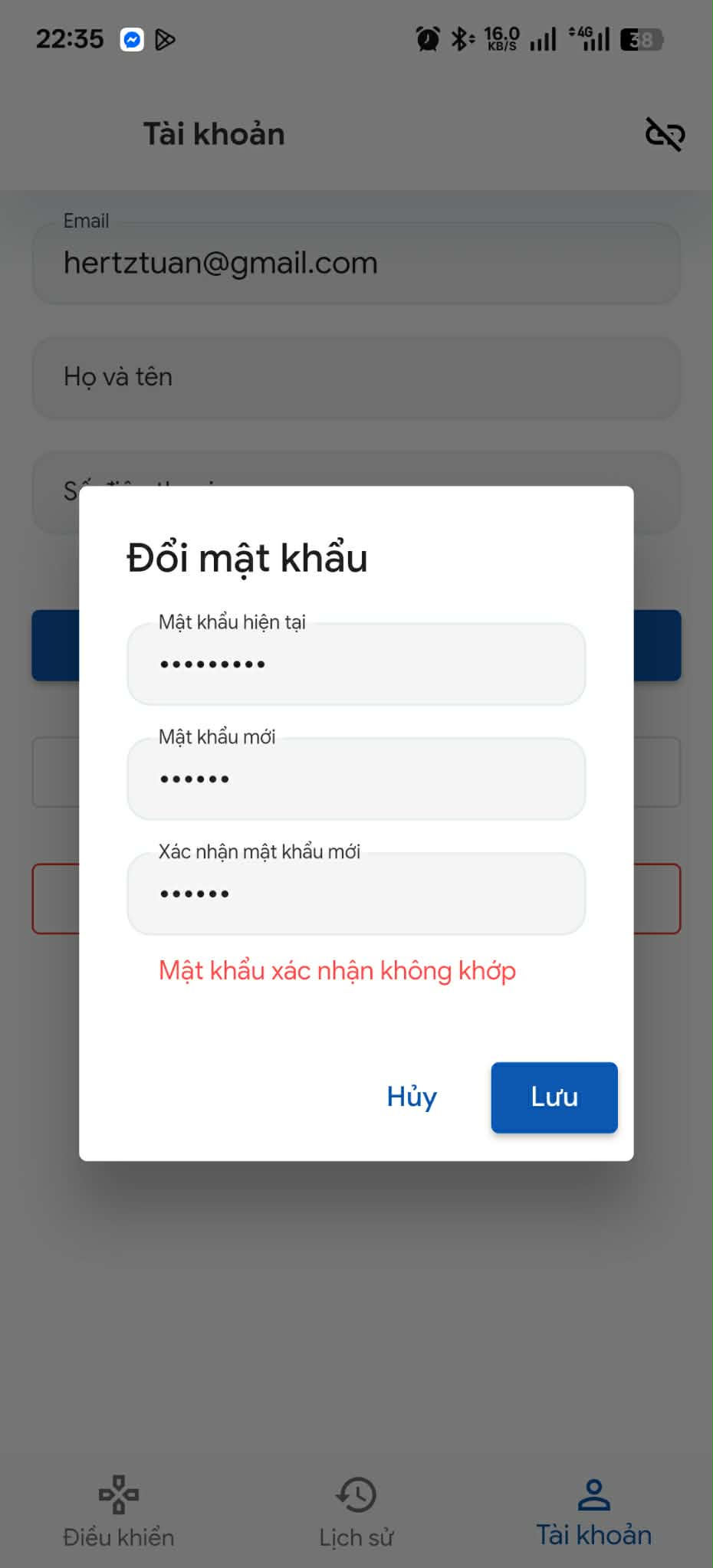
### 4.1.4: Chức năng đổi mật khẩu



Hinh 4.1.4.1: Giao diện đổi mật khẩu

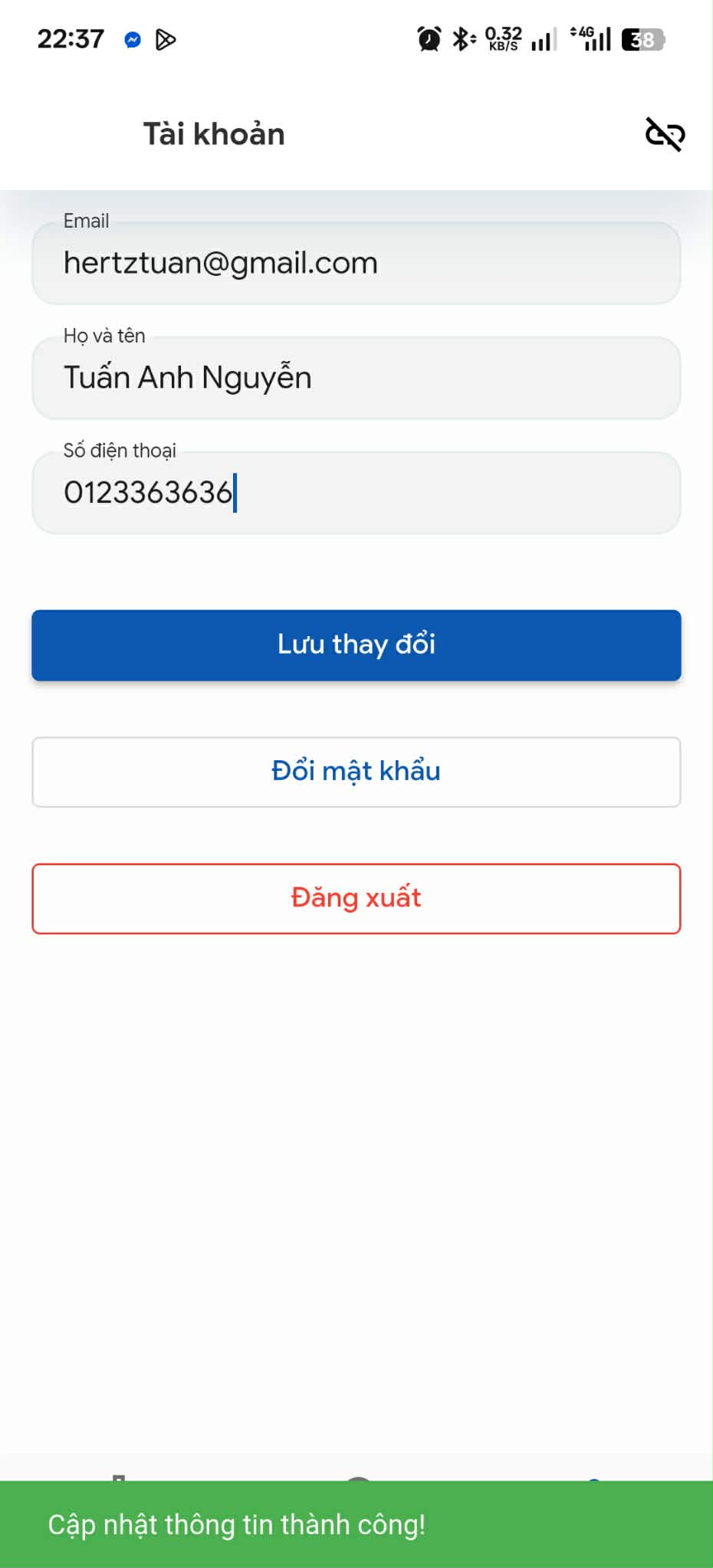


Hình 4.1.4.2: Khi không điền mật khẩu



Hình 4.1.4.3: Mật khẩu xác nhận không khớp

### 4.1.5: Chức năng cập nhật hồ sơ



Hình 4.1.5: Cập nhật thông tin hồ sơ

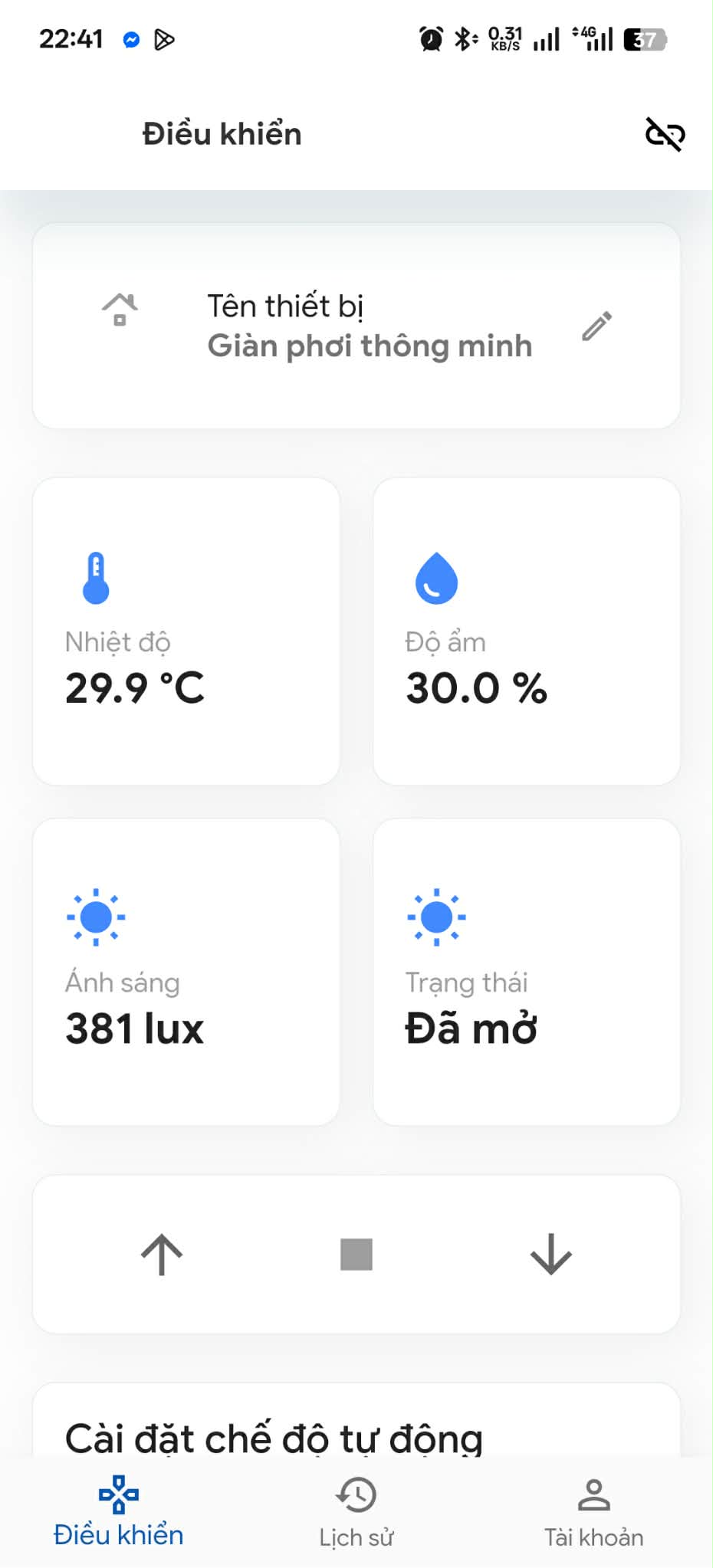
### 4.1.6: Giao diện danh sách thiết bị

A white background with black dots

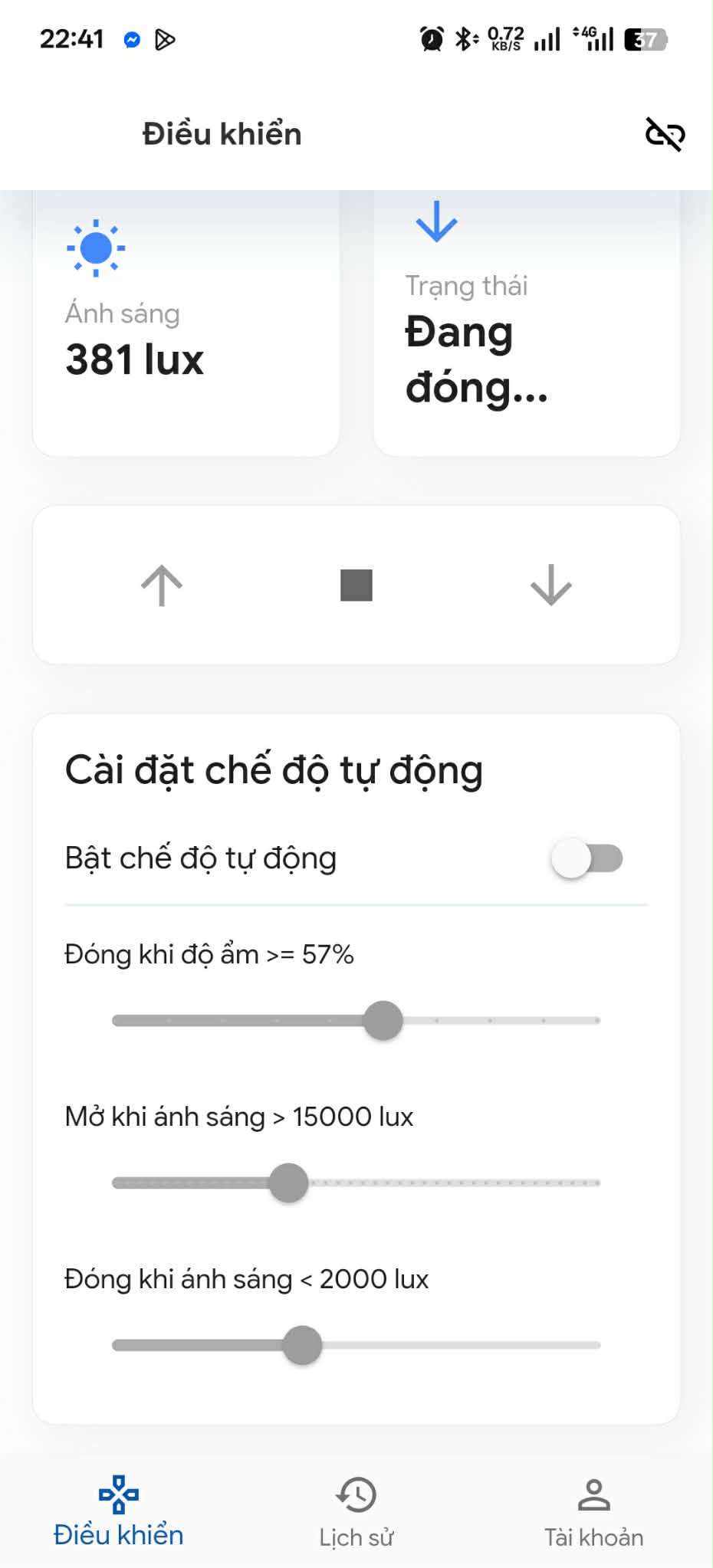
AI-generated content may be incorrect.

Hình 4.1.6: Giao diện danh sách thiết bị

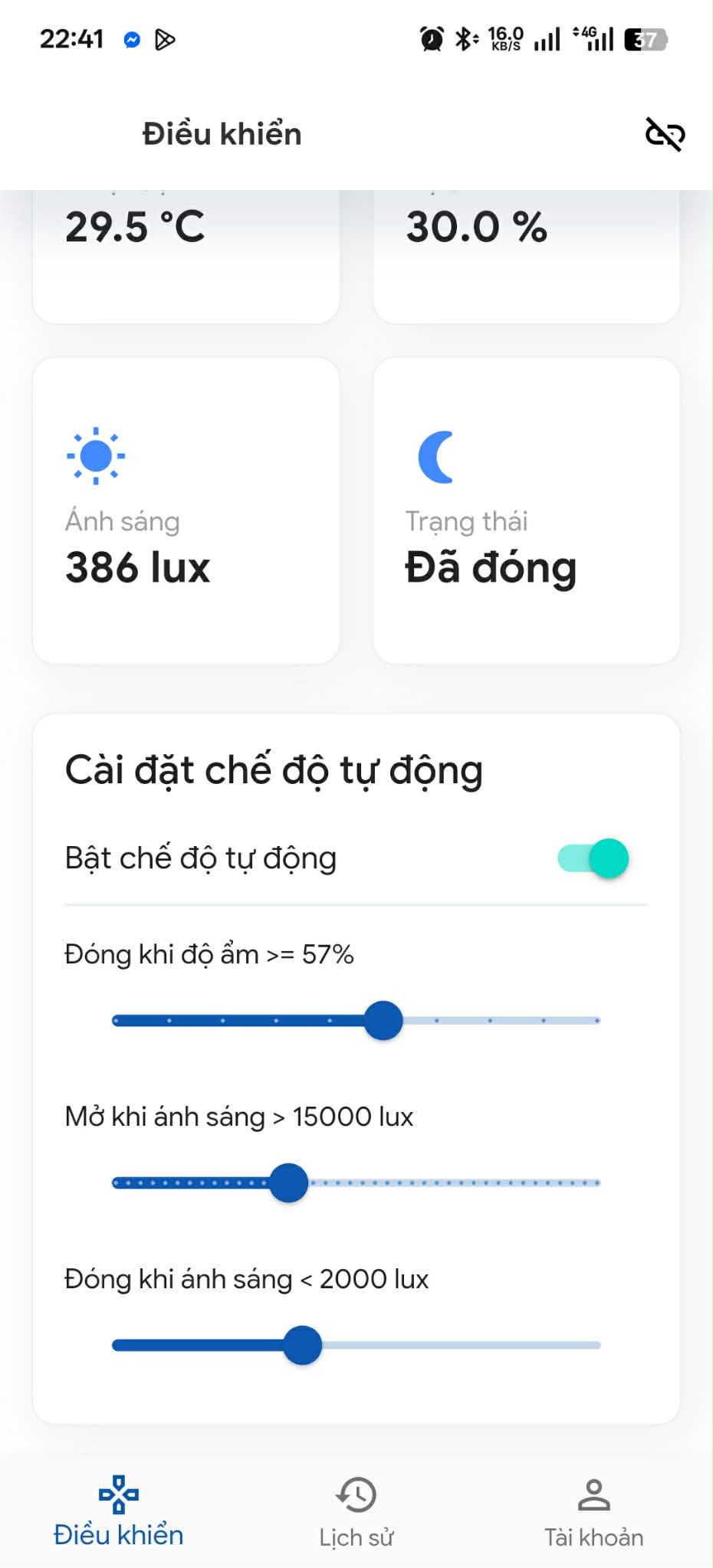
### 4.1.7: Giao diện điều khiển



Hình 4.1.7.1: Giao diện điều khiển

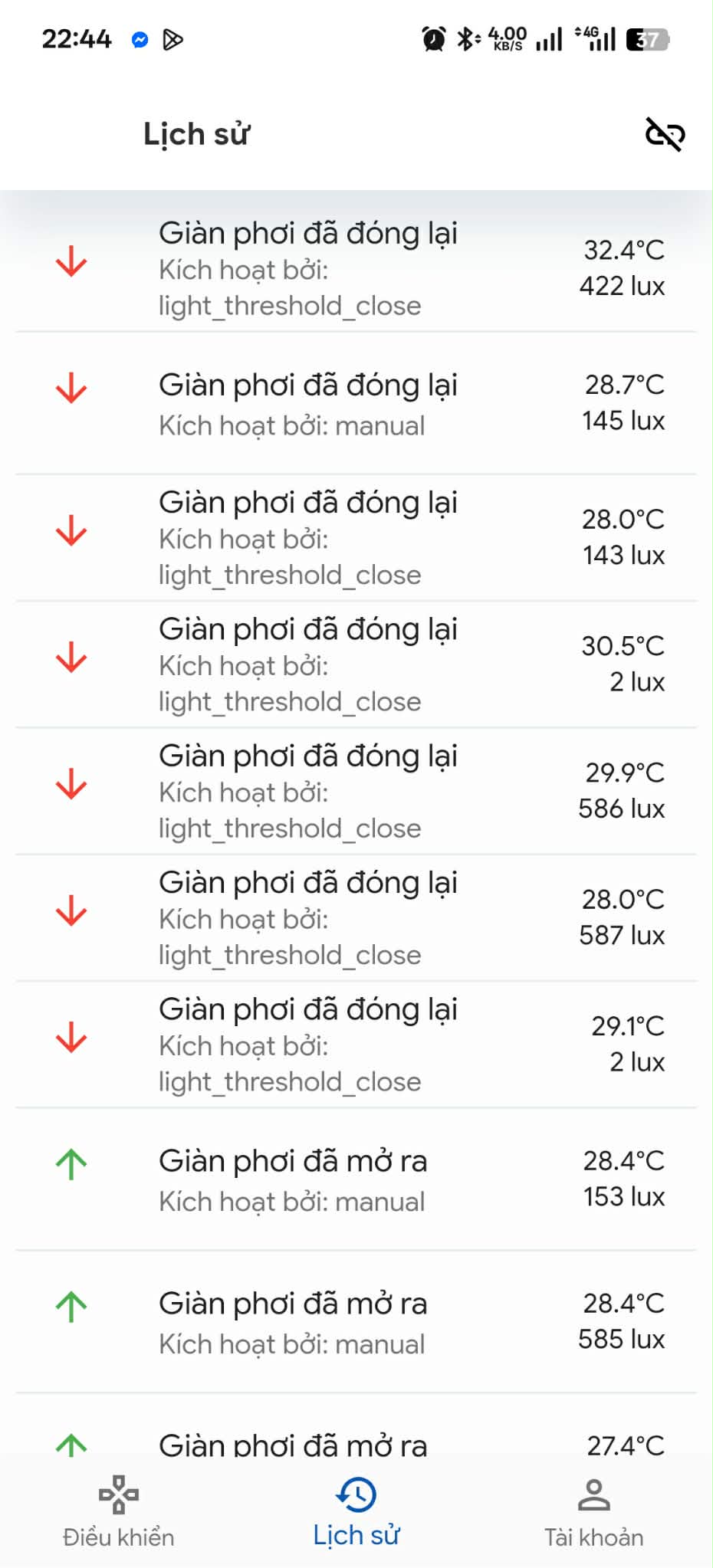


Hình 4.1.7.2: Giao diện điều khiển thủ công (Chế độ tự động tắt)



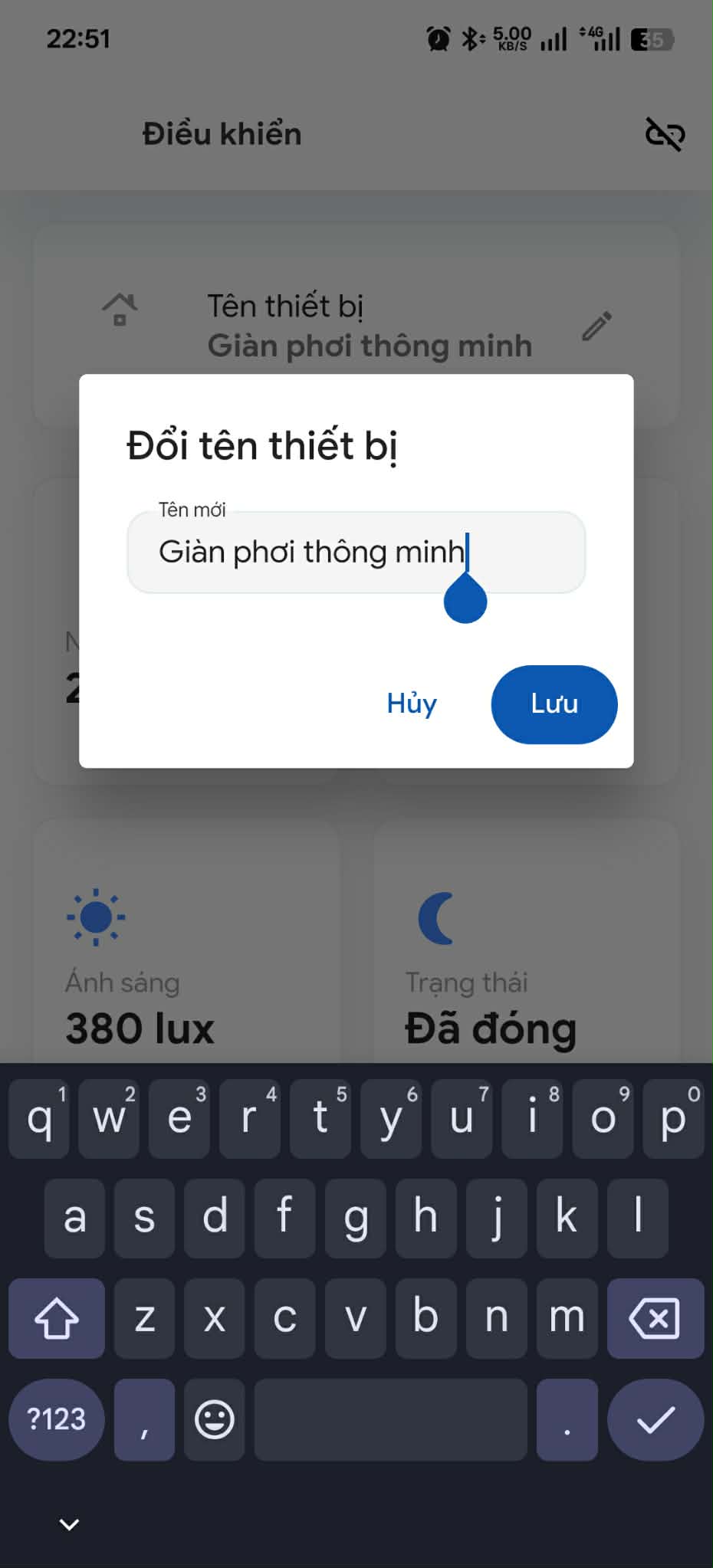
Hình 4.1.7.3: Giao diện điều khiển tự động

### 4.1.8: Chức năng xem lịch sử thiết bị



Hình 4.1.8: Giao diện xem lịch sử thiết bị

### 4.1.9: Chức năng sửa thông tin thiết bị



Hình 4.1.9: Chức năng sửa thông tin thiết bị

## 4.2: Kiểm thử chức năng trang web

### 4.2.1: Chức năng đăng nhập

A screenshot of a login form

AI-generated content may be incorrect.

Hình 4.2.1.1: Giao diện đăng nhập

A screenshot of a login form

AI-generated content may be incorrect.

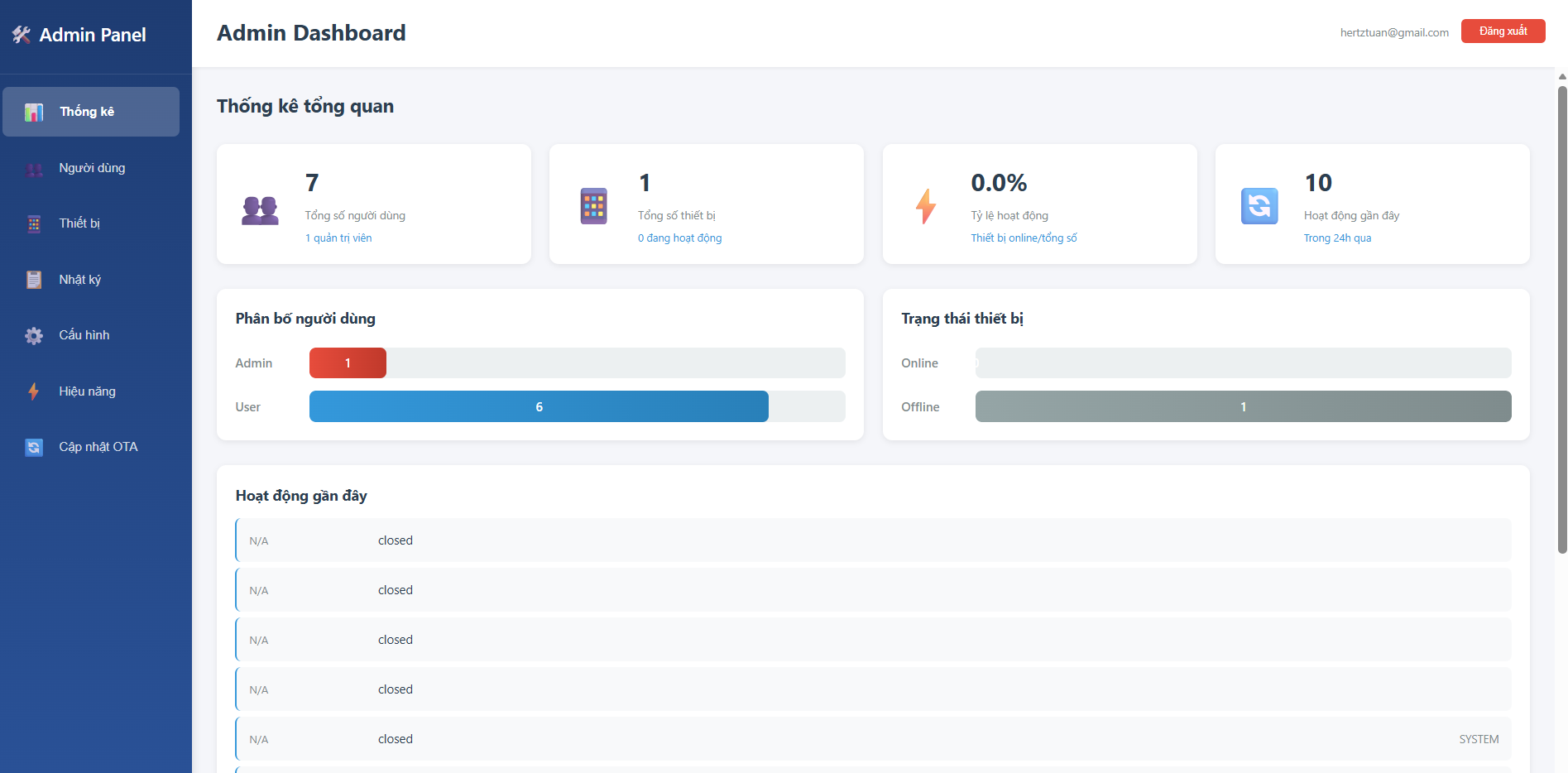
Hình 4.2.1.2: Thông báo khi không điền đủ thông tin

A screenshot of a login form

AI-generated content may be incorrect.

Hình 4.2.1.3: Thông báo sai mật khẩu

### 4.2.2: Chức năng thống kê



Hình 4.2.2.1: Giao diện thống kê

A green rectangle with white text

AI-generated content may be incorrect.

Hình 4.2.2.2: Giao diện thao tác nhanh

### 4.2.3: Chức năng quản lý người dùng

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.3: Giao diện quản lý người dùng

### 4.2.4: Chức năng quản lý thiết bị



Hình 4.2.4.1: Giao diện quản lý thiết bị

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.4.2: Giao diện chi tiết thiết bị

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.4.3: Thông báo xóa thiết bị

### 4.2.5: Chức năng quản lý nhật ký thiết bị

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.5: Giao diện quản lý nhật ký

### 4.2.6: Chức năng quản lý cấu hình hệ thống

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.6.1: Giao diện quản lý cấu hình hệ thống

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.6.2: Giao diện quản lý cấu hình MQTT

A screenshot of a video chat

AI-generated content may be incorrect.

Hình 4.2.6.3: Giao diện quản lý cấu hình mặc định thiết bị

### 4.2.7: Chức năng quản lý hiệu năng hệ thống

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.7.1: Giao diện giám sát hiệu năng hệ thống

### 4.2.8: Chức năng Update Firmware từ xa

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2.8: Chức năng update OTA

## 4.3: Kết quả kiểm thử

| **Công việc** | **Số lần thao tác** | **Số lần thành công** | **Thời gian đáp ứng** | **Ghi chú** |
| --- | --- | --- | --- | --- |
| **Phía người dùng** |  |  |  |  |
| Đăng ký tài khoản | 10 | 10 | <1s | Giao tiếp với Firebase Auth. |
| Đăng nhập | 10 | 10 | <1s | Giao tiếp với Firebase Auth. |
| Đổi mật khẩu | 5 | 5 | <1s | Giao tiếp với Firebase Auth. |
| Sửa thông tin thiết bị | 5 | 5 | <1s | Cập nhật thông tin trên Firestore. |
| Bật/Tắt chế độ tự động | 10 | 10 | <1s | Cập nhật trạng thái cấu hình trên Firestore để Cloud Function đọc. |
| Điều khiển giàn phơi (Mở/Đóng/Dừng) | 20 | 20 | 1-2s | Gửi lệnh qua MQTT Broker (qua Firebase). |
| Giám sát trạng thái cảm biến (Real-time) | 20 | 20 | <1s | Đọc dữ liệu từ Firestore/MQTT. |
| Xem lịch sử dữ liệu | 10 | 10 | 1-2s | Truy vấn và hiển thị dữ liệu lịch sử từ Firestore. |
| **Phía Admin** |  |  |  |  |
| Đăng nhập Admin | 5 | 5 | <1s | Giao tiếp với Firebase Auth (kiểm tra vai trò). |
| Thêm người dùng | 5 | 5 | 1s | Giao tiếp với Firebase Auth và Firestore (tạo profile). |
| Sửa thông tin người dùng | 5 | 5 | 1s | Cập nhật Firebase Auth/Firestore. |
| Xóa người dùng | 5 | 5 | <1s | Xóa người dùng khỏi Firebase Auth/Firestore. |
| Xem/Quản lý tất cả thiết bị | 5 | 5 | 1s | Truy vấn Firestore (quyền Admin). |
| Xem nhật ký hệ thống (Logs) | 5 | 5 | 1-2s | Truy vấn log từ Cloud Functions/Firebase. |
| Chỉnh sửa cấu hình | 10 | 10 | 2-3s |  |
| Update OTA | 5 | 5 | 2-3’ |  |
| **Phía Hệ thống tự động** |  |  |  |  |
| Tự động đóng/mở giàn phơi (Dựa trên cảm biến) | 20 | 19 | 1-2s | Phản ứng trực tiếp với cảm biến mưa/nắng (đã xử lý trễ). |
| Gửi cảnh báo (phát hiện mưa) | 10 | 10 | <1s | Kích hoạt Push Notification qua FCM. |
| Gửi cảnh báo (thiết bị Offline) | 5 | 5 | 1-2s | Kích hoạt Push Notification qua FCM. |

Bảng 4.3: Kết quả kiểm thử

**a. Về phần cứng:**

* Đã nghiên cứu và lựa chọn thành công các thiết bị phần cứng phù hợp, bao gồm kit ESP32 CP3102, động cơ DC giảm tốc, module driver L298N, cảm biến mưa, cảm biến nhiệt độ/độ ẩm DHT11 và cảm biến ánh sáng BH1750.
* Thực hiện lắp đặt và đấu nối mạch điện hoàn chỉnh cho mô hình giàn phơi/mái che mini, đảm bảo các linh kiện hoạt động ổn định và tương thích.
* Nắm vững kiến thức về nguyên lý hoạt động của các linh kiện điện tử, cách đọc và xử lý tín hiệu từ cảm biến, cũng như điều khiển động cơ thông qua module driver.

**b. Về phần mềm:**

* Xây dựng chương trình điều khiển cho ESP32 trên nền tảng Arduino IDE, sử dụng ngôn ngữ lập trình C/C++ để đọc dữ liệu cảm biến, điều khiển motor và giao tiếp Wi-Fi thông qua giao thức MQTT.
* Phát triển hệ thống backend sử dụng Firebase (Cloud Firestore, Firebase Authentication, Cloud Functions) để lưu trữ dữ liệu cảm biến, trạng thái thiết bị, quản lý người dùng và thực hiện các logic tự động hóa trên đám mây.
* Phát triển ứng dụng di động Flutter và web dashboard React với giao diện trực quan, cho phép người dùng giám sát dữ liệu cảm biến thời gian thực, xem lịch sử, và điều khiển giàn phơi từ xa.
* Đã thiết lập thành công giao tiếp MQTT giữa ESP32 và Firebase (hoặc MQTT Broker trung gian) để gửi nhận lệnh và dữ liệu telemetry theo thời gian thực.

## 4.4 Nhận xét, Kết luận và Hướng phát triển

### a. Nhận xét:

* **Ưu điểm:**
  + Hệ thống có thiết kế gọn nhẹ, các thành phần phần cứng được lựa chọn tối ưu về chi phí và dễ dàng lắp đặt, sử dụng.
  + Khả năng hoạt động ổn định và hiệu quả các chức năng cơ bản như thu thập dữ liệu cảm biến, điều khiển giàn phơi thủ công và tự động theo điều kiện mưa/nắng hiện tại.
  + Giao diện người dùng trên ứng dụng di động và web trực quan, dễ sử dụng, cung cấp khả năng theo dõi dữ liệu thời gian thực và lịch sử thông qua biểu đồ.
  + Chức năng tự động đóng/mở giàn phơi theo mưa/nắng hoạt động tốt, đã có cơ chế xử lý trễ để tránh các hành động đóng/mở liên tục không cần thiết khi thời tiết chập chờn.
* **Nhược điểm:**
  + Hệ thống vẫn còn một số hạn chế về khả năng dự đoán và tối ưu hóa quyết định trong các tình huống thời tiết phức tạp.
  + Thời gian đáp ứng cho một số tác vụ có thể cần tối ưu thêm, đặc biệt là các quy trình liên quan đến xử lý trên Cloud Functions.
  + Mô hình hiện tại tập trung vào việc phản ứng với thời tiết tại chỗ, chưa hoàn toàn chủ động dựa trên dự báo dài hạn.

### b. Kết luận:

Đề tài đã hoàn thành việc xây dựng một hệ thống giàn phơi/mái che thông minh mini hoạt động ổn định, đáp ứng được các mục tiêu ban đầu về điều khiển tự động theo mưa/nắng và điều khiển từ xa qua ứng dụng/web. Hệ thống sử dụng các công nghệ IoT phổ biến như ESP32, Firebase, MQTT, Flutter và React, chứng minh khả năng ứng dụng thực tế trong việc tự động hóa các tác vụ gia đình, đặc biệt hữu ích cho các căn hộ có không gian ban công hạn chế. Việc tích hợp thành công cả phần cứng và phần mềm cho thấy tiềm năng lớn để phát triển thêm các tính năng thông minh hơn.

### c. Hướng phát triển:

Để nâng cao tính thông minh và hiệu quả của hệ thống, chúng tôi đề xuất một số hướng phát triển như sau:

* **Tích hợp Trí tuệ Nhân tạo (AI) / Học máy (ML):**
  + **Mô hình dự đoán thời tiết**: Dựa vào dữ liệu lịch sử từ các cảm biến và dữ liệu dự báo từ các API thời tiết (ví dụ: OpenWeather API), xây dựng và triển khai mô hình Học máy (đặc biệt là RNN cho dữ liệu chuỗi thời gian) trên Cloud Functions (hoặc Vertex AI) để dự đoán sớm khả năng mưa/nắng trong tương lai gần. Điều này cho phép giàn phơi/mái che đưa ra quyết định chủ động hơn (ví dụ: tự động thu vào 30 phút trước khi trời mưa).
  + **Tối ưu hóa hành trình:** Sử dụng ML để học hỏi hành vi người dùng và điều kiện môi trường, từ đó tối ưu hóa các quyết định đóng/mở, giúp tiết kiệm năng lượng và tăng tuổi thọ thiết bị.
* **Mở rộng khả năng thông báo:**
  + Ngoài thông báo đẩy (push notifications) qua Flutter, có thể tích hợp gửi email cảnh báo hoặc tin nhắn SMS trong các trường hợp khẩn cấp (ví dụ: thiết bị offline dài ngày, lỗi hệ thống nghiêm trọng) để đảm bảo người dùng nhận được thông tin kịp thời.
* **Tích hợp thêm cảm biến:**
  + Thêm cảm biến gió để tự động thu giàn phơi khi gió quá mạnh, bảo vệ đồ đạc và thiết bị.
  + Tích hợp cảm biến chất lượng không khí để đưa ra khuyến nghị phơi đồ tốt nhất.
* **Chức năng chia sẻ thiết bị:**
  + Cho phép người dùng chia sẻ quyền điều khiển và giám sát thiết bị với các thành viên khác trong gia đình.
* **Cải thiện giao diện:**
  + Tích hợp bản đồ hoặc sơ đồ trực quan hơn để hiển thị vị trí và trạng thái của nhiều thiết bị cùng lúc (nếu người dùng có nhiều giàn phơi).
  + Cung cấp các tùy chọn cá nhân hóa sâu hơn cho người dùng.

# TÀI LIỆU THAM KHẢO