HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

_____O_____

BÀI TẬP LỚN

Môn học: IoT và ứng dụng

Đề tài: Hệ thống nhà thông minh

Giảng viên: Nguyễn Quốc Uy

Nhóm môn học: 05

Sinh viên: Nguyễn Trần Đạt

Mã sinh viên: B21DCCN216

Hà Nội, 2024

Mục Lục

Chương 1: Tổng quan về hệ thống IoT

1.1 Giới thiệu chung về IoT

Ngày nay có hàng tỷ thiết bị điện tử được kết nối internet. Những thiết bị này thu thập và chia sẻ dữ liệu và tạo nên "Internet of Things" – một mạng lưới cho phép các công nghệ riêng lẻ làm được nhiều việc cùng nhau một cách đồng bộ, hiệu quả. Ý tưởng về Iot bắt nguồn từ 1926, Nikola Tesla đã hình dung ra một "thế giới được kết nối". Ông dự đoán về một "world converted into a huge brain" trong một bài phỏng vấn của tạp chí Colliers.

Người đầu tiên sử dụng thuật ngữ IoT là vào năm 1999, Kevin Ashton, trong lĩnh vực quản lý chuỗi cung ứng với nhận dạng sản phẩm qua tần số vô tuyến (RFID) – các mặt hàng được gắn thẻ hoặc mã vạch giúp mang lại hiệu quả cao hơn. Máy ATM có thể được coi là một trong những smart thing đầu tiên được đưa vào thực tế từ 1974, ngoài ra còn có máy bán hàng tự động vào năm những năm 1980.

IoT – Internet of Things – là mạng kết nối các "Things", các đồ vật, con người,… thông qua các thiết bị cảm biến, phần mềm và các công nghệ khác, cho phép các "Things" có thể thu thập và trao đổi dữ liệu với nhau. IoT giúp những thiết bị được kết nối trở lên thông minh hơn nhờ khả năng nhật và hoạt động dựa trên những dữ liệu đó.

1.2 Ứng dụng của IoT

Tiềm năng to lớn của IoT đã hiện thực hoá rất nhiều các ứng dụng trong nhiều lĩnh vực khác nhau để cải thiện chất lượng cuộc sống của con người. Tự động hoá, giao thông thông minh, nông nghiệp và canh tác thông minh, sản xuất thông minh, giáo dục thông minh,… là một vài lĩnh vực IoT đóng ột vai trò quan trọng. Trong thời gian gần đây, việc

trang bị các thiết bị thông minh với các khả năng khác nhau đã giúp cải thiện tiêu chuẩn cuộc sống của con người ở các năng lực khác nhau.

Miền ứng dụng IoT	Ví dụ
Tự động hóa trong nhà	Cuộc sống thoải mái, tự động hóa gia đình, giám sát và an ninh gia đình, thiết bị thông minh, bảo vệ trẻ em, giám sát video, v.v.
Chăm sóc sức khỏe và sống tốt	Dịch vụ bệnh viện thông minh, theo dõi bệnh nhân từ xa, hỗ trợ người cao tuổi, hỗ trợ người tàn tật, thiết bị y tế/dược phẩm, theo dõi và quản lý xe cứu thương, chẩn đoán và kiểm tra từ xa, quản lý hồ sơ y tế, hỗ trợ sinh hoạt, hỗ trợ người chăm sóc từ xa, hỗ trợ di động, v.v.
Giao thông thông minh	Ô tô được kết nối, tính di động thông minh, giám sát đường bộ, chia sẻ phương tiện, ô tô tự động, hệ thống thanh toán tự động, hệ thống đỗ xe, bảo trì ô tô theo lịch chủ động, ứng dụng an toàn, ứng dụng thông tin giải trí, ứng dụng quản lý giao thông, quản lý tín hiệu giao thông, tàu thông minh.

Miền ứng dụng IoT	Ví dụ
Nông nghiệp và canh tác thông minh	Giám sát và quản lý đồng ruộng/trang trại/đồng cỏ, giám sát và quản lý sản xuất nông nghiệp và thức ăn chăn nuôi, quản lý và theo dõi vật nuôi, tưới tiêu đồng ruộng/trang trại, giám sát quá trình chế biến thực phẩm, quản lý, hạn sử dụng và tự động hóa đặt hàng, giao hàng và thanh toán các sản phẩm nông nghiệp, v.v.
Tự động hóa công nghiệp, Sản xuất thông minh và Logistics	Sản xuất thông minh bao gồm xác định nguyên liệu và hư hỏng sản phẩm, quản lý kho, giám sát nhà máy công nghiệp, quản lý hành lý, theo dõi vận chuyển và hoạt động lên tàu, giám sát quy trình/thiết bị, giám sát cơ sở của nhân viên và nhà cung cấp, v.v.
Giáo dục thông minh	Giám sát và an ninh của các viện giáo dục, giáo dục từ xa, học tập điện tử, tự động hóa các dịch vụ nhận dạng sinh viên/giảng viên, v.v.
An toàn công cộng và quân sự	Nhận dạng và theo dõi vũ khí, phát hiện bắn tỉa, giám sát lãnh thổ thông qua máy quay video, radar và vệ tinh, kế hoạch khẩn cấp và cứu hộ, v.v.
Bán lẻ và Khách sạn	Quản lý hàng tồn kho, hoạt động mua sắm, thanh toán nhanh, chống trộm cắp và gian lận, hỗ trợ nhiều ngôn ngữ, v.v.
Bảo tồn năng lượng	Lưới điện thông minh để sản xuất, phân phối và quản lý điện, đo lường thông minh, quản lý nước thông minh, điều phối tải thông minh-cân bằng với pin và hệ thống lưu trữ, v.v.
Ứng dụng Chính phủ và khu vực Tư nhân	Thành phố thông minh, tòa nhà thông minh, cộng đồng thông minh, giám sát môi trường trong thời gian thực, nước đô thị và giám sát hệ thống thoát nước, cung cấp dịch vụ từ xa và giám sát tuân thủ, bảo trì và quản lý các địa điểm và công viên lịch sử, v.v.

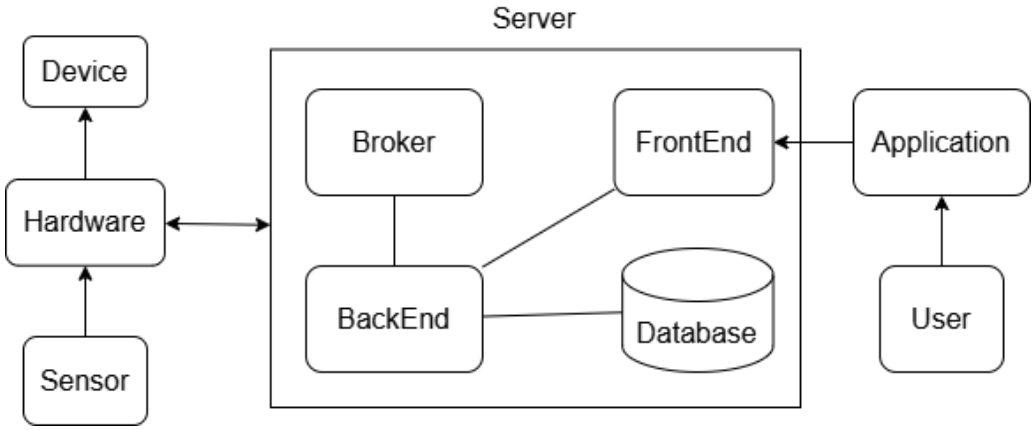
1.3 Giới thiệu đề tài

Ngày nay có hàng tỷ thiết bị điện tử được kết nối internet. Những thiết bị này thu thập và chia sẻ dữ liệu và tạo nên "Internet of Things" – một mạng lưới cho phép các

công nghệ riêng lẻ làm được nhiều việc cùng nhau một cách đồng bộ, hiệu quả. Ý tưởng về Iot bắt nguồn từ 1926, Nikola Tesla đã hình dung ra một "thế giới được kết nối". Ông dự đoán về một "world converted into a huge brain" trong một bài phỏng vấn của tạp chí Colliers.

1.4 Mô hình hệ thống IoT

Với bất kỳ một công nghệ mới nào cũng sẽ có các mô hình kiến trúc. Mô hình kiến trúc là một cái nhìn tổng quát về việc vận hành của công nghệ đó. Một kiên trúc được chia thành các lớp riêng biệt, đảm nhận một vai trò riêng trong quá trình vận hành của hệ thống. Trong hệ thống này, có 3 lớp chính: Hardware, Server, Application.

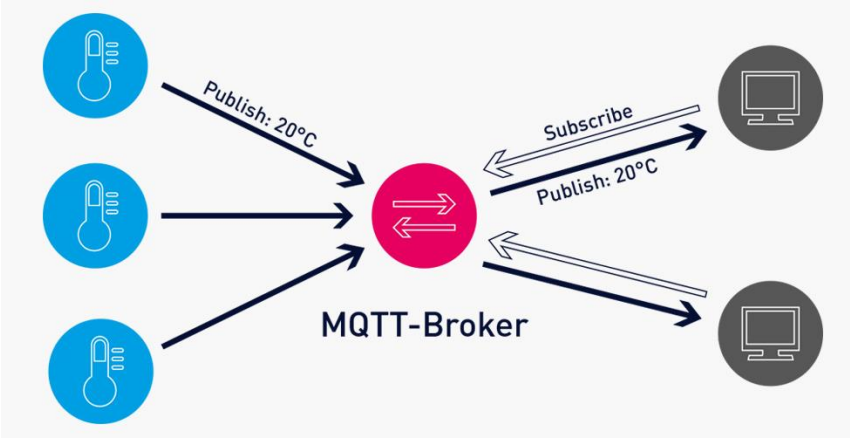


1.5 Các giao thức

Các giao thức là một phần quan trọng trong hệ thống IoT. Chúng là những quy tắc giúp các thiết bị có thể liên kết với nhau, giúp các thiết bị phần cứng liên kết với phần mềm thông qua mạng internet.

1.5.1 Giao thức MQTT

Giao thức MQTT là giao thức truyền thông tệp theo mô hình Publish/Subcribe, thích hợp với những thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định.



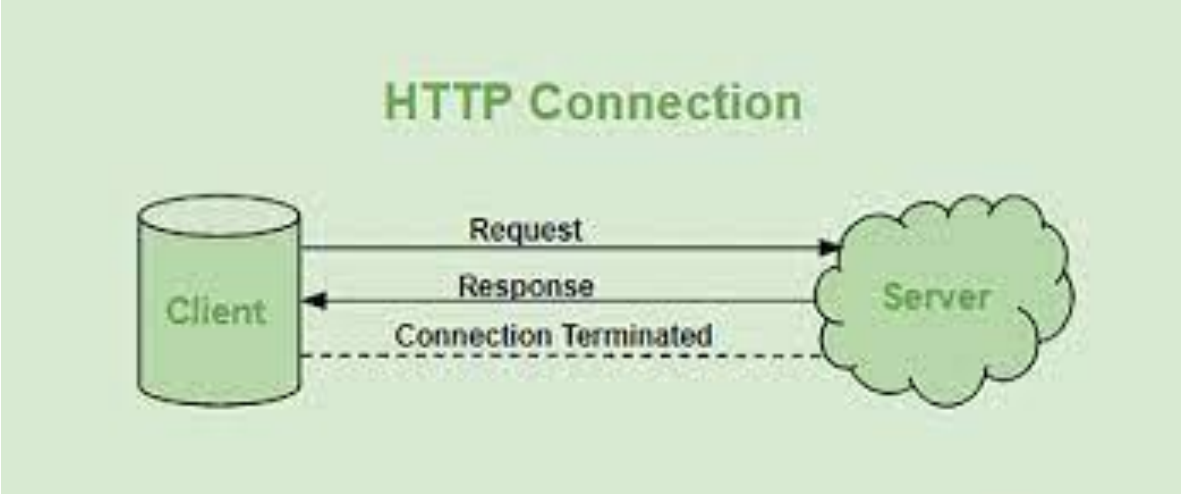
Giao thức MQTT có một server riêng gọi là MQTT broker, có thể kết nối với nhiều thiết bị (MQTT client). Các client có thể Publish thông tin thu thập từ các cảm biến, sau đó truyền lên Broker. Các Client khác có thể Subcribe những thông tin này về thiết bị để xem, hoặc Publish những tín hiệu điều khiển.

Giao thức này là sự lựa chọn lý tưởng cho những nơi có mạng băng thông thấp, cho những thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ và lý tưởng cho ứng

dụng M2M. MQTT được sử dụng nhiều trong ngành công nghiệp dầu khí, thám hiểm, các công ty lớn như Amazon, Facebook…bởi chi phí thấp, tiết kiệm thời gian và độ an toàn, bảo mật cao.

1.5.2 Giao thức HTTP

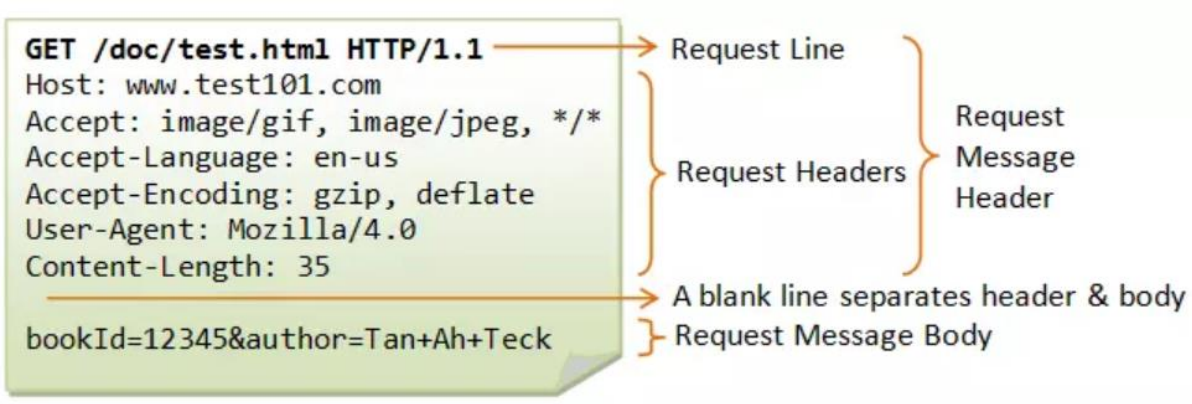
Giao thức HTTP – Giao thức truyền tải siêu văn bản – là một trong các giao thức chuẩn về mạng Internet, chuyên dùng để liên hệ thông tin giữa máy cung cấp dịch vụ (Web Server) và máy sử dụng (Web Client), là giao thức Client/Server phổ biến nhất hiện nay dùng cho World Wide Web và là một giao thức ứng dụng của bộ giao thức TCP/IP.



Trong mô hình Client/Server, máy tính người dùng đóng vai trò làm máy khách (Client). Máy khách khi muốn kết nối với Server cần gửi một yêu cầu đến máy chủ và chờ câu trả lời từ máy chủ. Khi máy chủ cho phép kết nối, giữa máy khách và máy chủ sẽ tạo một giao diện kết nối tạm thời cho phép truyền thông tin trên đó.

HTTP request method là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định. Một request sẽ bao gồm:

- Request-line: Xác định phương thức, URI-Request và phiên bản HTTP.
- Có thể hoặc không có các trường header: Các trường header cho phép client truyền thông tin bổ sung về yêu cầu và về chính client, đến server.
- Một dòng trống để tách header và body
- Body message

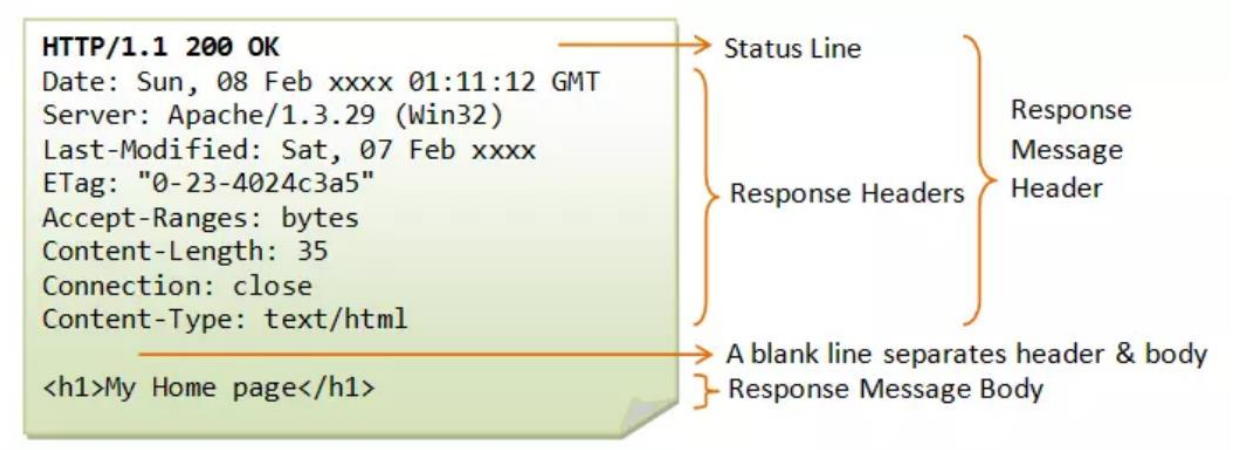


Một số HTTP request method thường được dùng:

Method	Hoạt dộng
GET	Được sử dụng để lấy lại thông tin từ Server một tài nguyên xác định
POST	Yêu cầu máy chủ chấp nhận thực thể được đính kèm trong request được xác định bởi URI.
PUT	Nếu URI đề cập đến một tài nguyên đã có, nó sẽ bị sửa đổi; nếu URI không trỏ đến một tài nguyên hiện có, thì máy chủ có thể tạo ra tài nguyên với URI đó.
DELETE	Xoá bỏ tất cả đại diện của tài nguyên được chỉ định bởi URI
PATCH	Áp dụng cho việc sử đổi một phần của tài nguyên được xác định
…	…

Sau khi nhận được request từ client và map yêu cầu đó với tập tin trong tài liệu của server hoặc một chương trình trên server. Kết quả của chương trình và tập tin đó sẽ được server phản hồi client thông qua HTTP Respone. Cấu trúc của một HTTP response:

- Status-line xác định phiên bản HTTP, mã trạng thái và trạng thái. Mã trạng thái thông báo về kết quả khi nhận được yêu cầu và xử lý bên server cho client. Ví dụ mã 2xx là mã thành công, 4xx là mã liên quan đến lỗi phía client,…
- Có thể có các trường header
- Một dòng trống để tách header và body
- Body message



Ngoài những giao thức trên còn nhiều giao thức khác nữa trong hệ thống IoT . Trong bài báo cáo tiểu luận này, chỉ trình bày một số giao thức cơ bản , được sử dụng trong hệ thống của em.

1.6 Công nghệ sử dụng

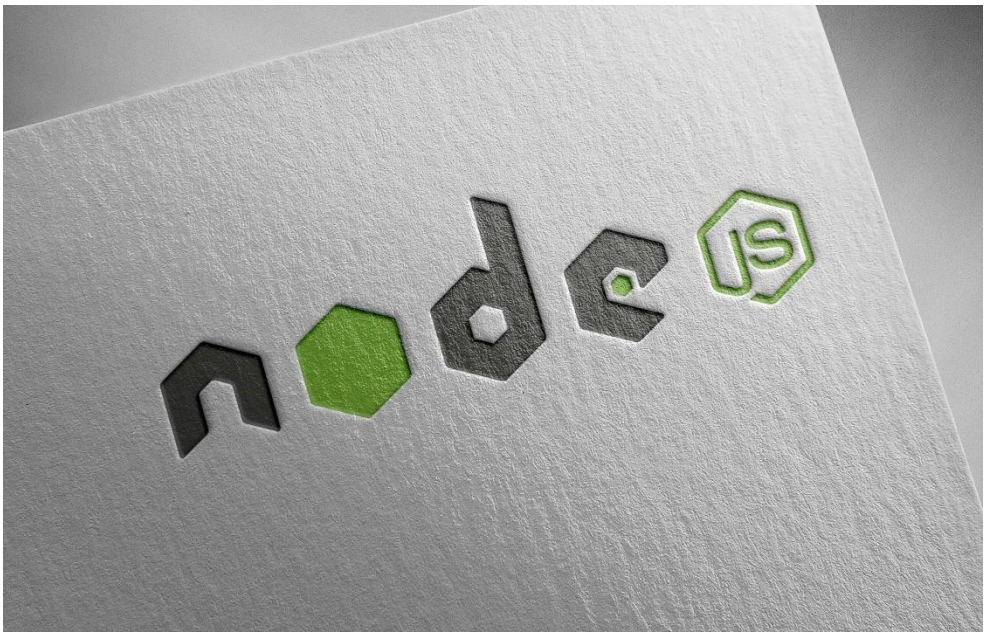
1.6.1 Cơ sở dữ liệu MySQL

MySQL là một hệ quản trị cơ sở dữ liệu tự do mã nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng. MySQL có nhiều ưu điểm nội trội như tốc độ cao, ổn định và dễ sử dụng, tương thích với nhiều hệ điều hành và cung cấp hệ thống lớn các hàm tiện ích rất mạnh. MySQL dược sử dụng cho việc bổ trợ Node.js, PHP, Perl và nhiều ngôn ngữ lập trình khác.

1.6.2 Node.js

Node.js là một nền tảng được xây dựng trên ‘V8 Javascript engine’ được viết bằng C++ và JasvaScript. Nền tảng này được phát triển vào năm 2009 bởi Ryan Lienhart Dahl.

Nền tảng này ra đười khi các developer đời đầu của JavaScript mở rộng nó từ một thứ chỉ chạy được trên trình duyệt thành m ột thứ có thể chạ trên máy tính dưới dạng ứng dụng độc lập.



Node.js là một lựa chọn linh hoạt cho các nhà phát triển muốn xây dựng một ứng dụng đa nền tảng bởi nó là mã nguồn mở và không phụ thuộc vào bất kỳ hệ điều hành nào cụ thể, có thể chạy trên Linux, macOS hay Windows.

1.6.3 Android studio với Jetpack compose

Android studio là môi trường phát triển tích hợp chính thức dành cho phát triển ứng dụng trên nền tảng Android. Android Studio được thiết kế dựa tren phần mềm IntelliJ IDEA của JetBrains là IDE chính thức của Google để phát triển ứng dụng Android Native thay thế cho Android Development Tool (ADT) dựa trên Eclipse.

Jetpack compose là một UI framework mã nguồn mở cho Android được phát triển dựa trên Kotlin bởi Google, lần đầu được giới thiệu vào tháng 5 năm 2019 và sẵn sàng cho production vào 2021.

Jetpack compose hỗ trợ từ Android 5.0, sử dụng ngôn ngữ lập trình Kotlin. Thư viện này cung cấp mô hình Reactive programing tương tự như các UI framework khác như Vue.js hay React Native. Compose được thiết kế để tích hợp liền mạch với ứng dụng Android và thư viện đã tồn tại. Từ khi ra mắt Jetpack compose đã phát triển rất nhanh, được các nhà phát triển lựa chọn thay thế cho XML view layout rườm rà. Tính đến tháng 10 năm 2022, trong top 1000 ứng dụng trên Play Store có 160 ứng dụng được tích hợp Compose.

Chương 2: Xây dựng hệ thống nhà thông minh

Với sự phát triển nhanh chóng của các thiết bị phần cứng, phần mềm và các dịch vụ viễn thông, Iot đang dần trở nên phổ biến trong đời sống thường ngày của con người. Chúng ta có thể dễ dàng tìm thấy một ứng dụng của hệ thống IoT trong một lĩnh vực bất kì.

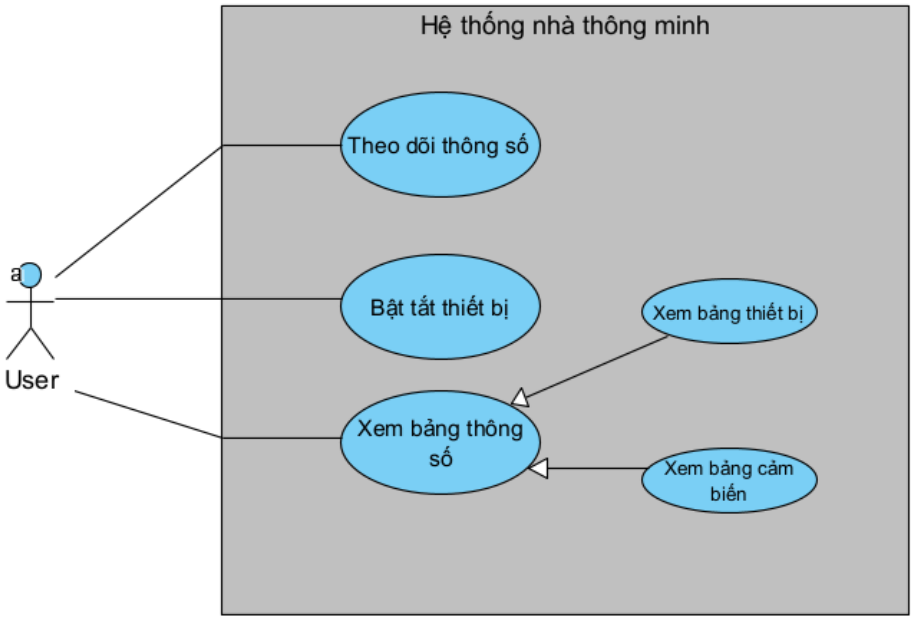
Hệ thống nhà thông minh là một ứng dụng của IoT trong lĩnh vực đời sống, thường nhật. Hệ thống này cho phép người dùng có thể theo dõi được các thông số về nhiệt độ, độ ẩm, ánh sáng của ngôi nhà của mình, đồng thời điều khiển một vài các thiết bị điện trong căn nhà.

2.1 Mô hình hệ thống

2.1.1 Sơ đồ usecase

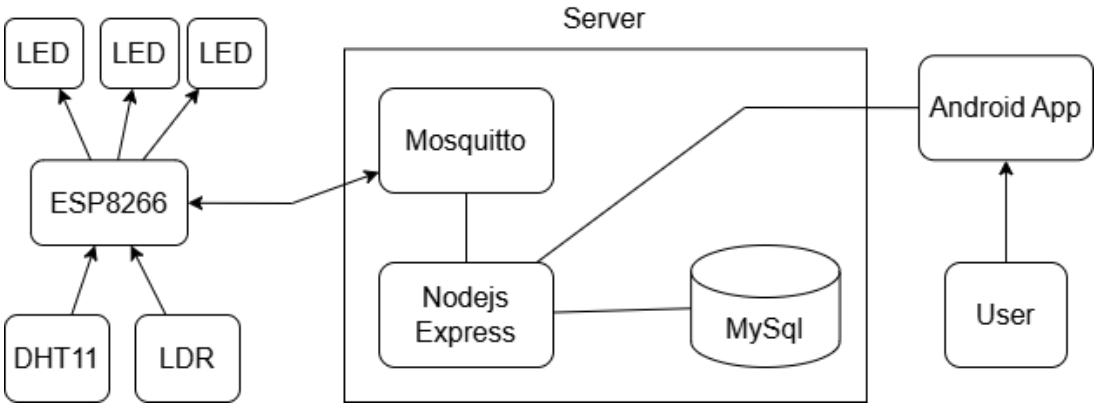
Các chức năng chính của hệ thống được mô tả như sau:

- Actor:
  - User: Người dùng hệ thống
- Usecase:
  - Theo dõi thông số
  - Bật tắt thiết bị
  - Xem bảng thông số cảm biến
  - Xem bảng lịch sử bật tắt



2.1.2 Mô hình hệ thống

Mô hình hệ thống được miêu tả bằng sơ đồ sau đây:



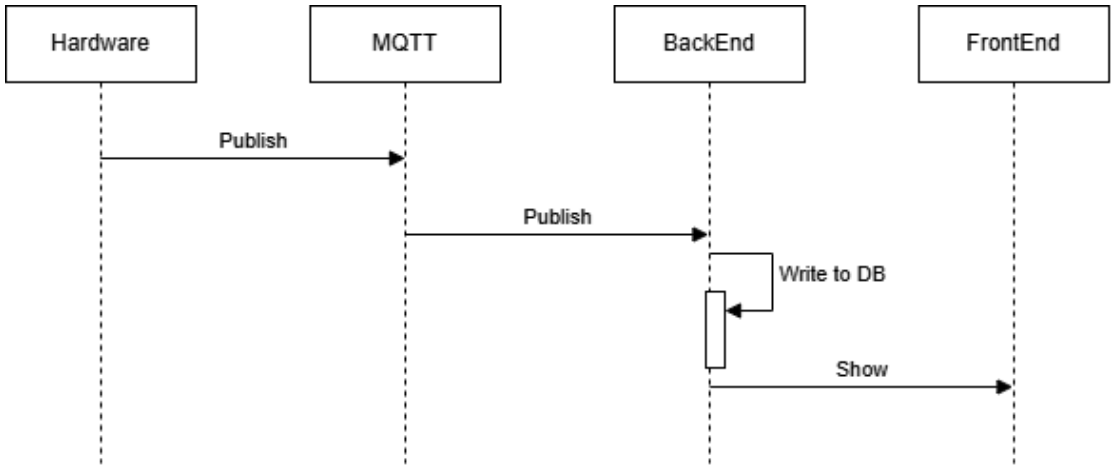
Khối hardware gồm có ESP8266, cảm biến nhiệt độ, độ ẩm DHT11, một cảm biến ánh sáng LDR.

Khối server gồm có Mosquitto broker đóng vai trò là cầu nối giữa ESP8266 và Nodejs. Nodejs Express là backend xử lý dữ liệu gửi về từ ESP8266, Request từ Frontend. Dữ liệu sẽ được lưu vào MySql.

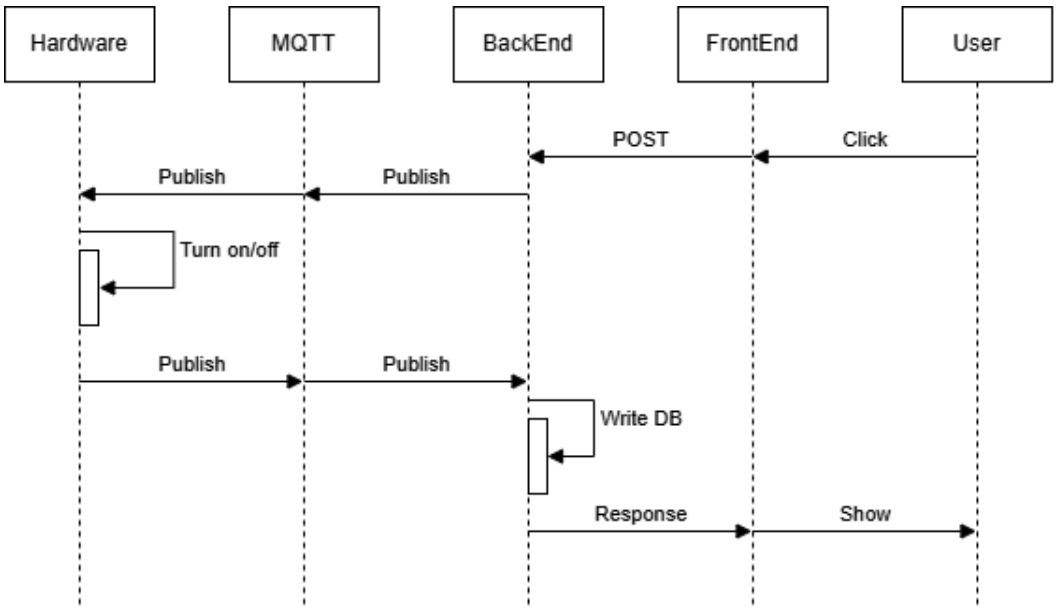
Người dùng sẽ tương tác với hệ thống thông qua một ứng dụng android.

2.1.3 Sequence Diagram

Lấy dữ liệu từ hardware:



Người dùng điều kiểu khiển thiết bị:



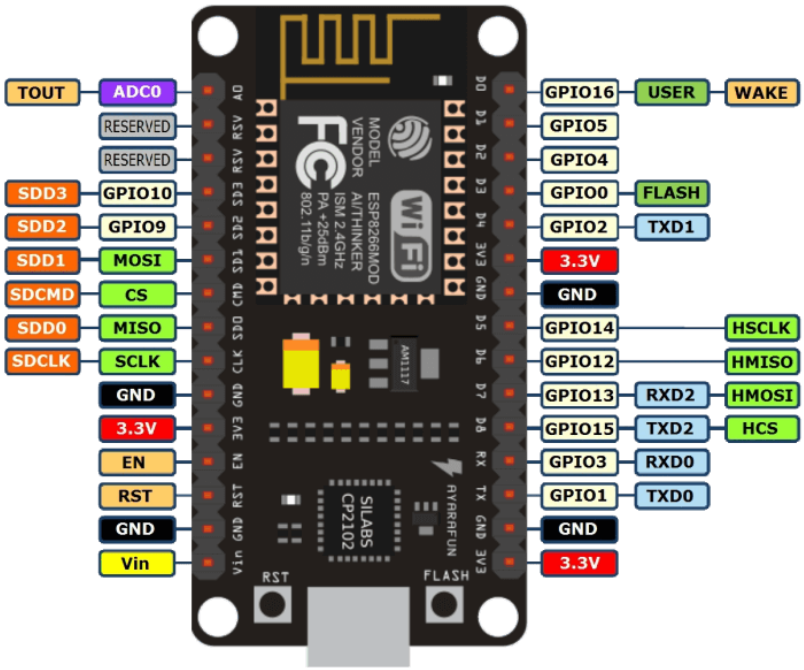
2.2 Thiết bị phần cứng

Các thiết bị phần cứng được sử dụng trong hệ thống này bao gồm:

- Cảm biến DHT11
- Cảm biến LDR
- Vi điều khiển ESP8266

2.2.1 ESP8266

ESP8266 là một hệ thống chip (SoC), do công ty Espressif của Trung Quốc sản xuất. Modul này bao gòm một vi điều khiển Ténilica L106 32-bit (MCU) và bộ thu phát WiFi. Vi điều khiển ESP8266 có tích hợp như Bluetooth, Wifi, UART, SPI, I2C, GPIO và nhiều nút chức nanwg khác, nhờ đó mà nó trở thành một lựa chọn phổ biến trong lĩnh vực IoT.



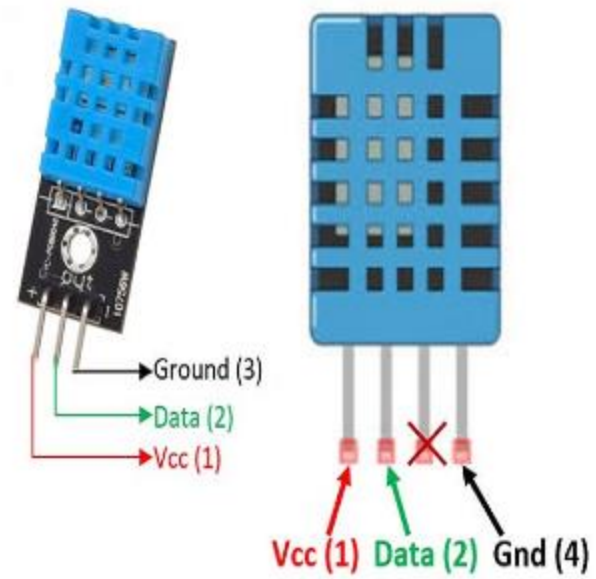
Trong Module ESP8266 có cấu hình chân bao gồm:

- EN, RST: Chốt và nút đặt lại
- A0: chân Analog
- GPIO Pin: từ GPIO1 đến GPIO16 hỗ trợ giao thức GPIO
- SPI Pins: SD1, CMD, SD0, SCL, hỗ trợ giao tiếp SPI
- UART Pins: TXD0, RXD0, TXD2, RXD2: hỗ trợ giao thức UART
- I2C Pins: Cấu hình chung không có chân nào hỗ trợ I2C, nhưng có thể tự thiết lập bằng lệnh.

Ưu điểm của việc sử dụng module ESP8266 đó là có thể tích hợp, điều khiển cùng lúc nhiều thiết bị (có thể lên tới hàng trăm thiết bị) mà vẫn có thể đảm bảo tính an toàn, chính xác. Nó còn tiết kiệm thời gian nhân công vận hành, tắt mở, điều chỉnh, thích hợp với những dây chuyền sản xuất hàng loạt. Nhờ việc có kết nối Internet, người sử dụng có thể điều khiển thiết bị mọi lúc, mọi nơi.

2.2.2 Cảm biến DHT11

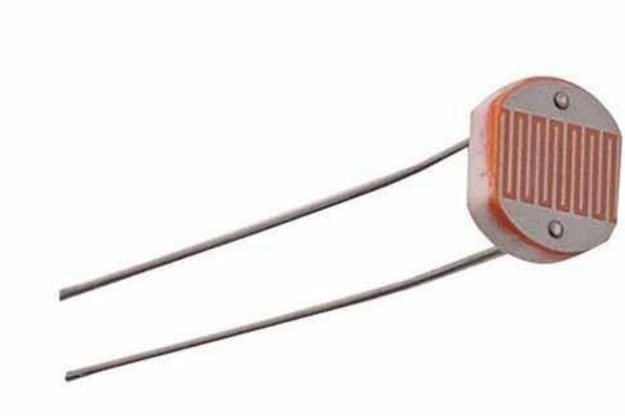
Cảm biến nhiệt độ và độ ẩm DHT11 ra đời sau và được sử dụng thay thế cho dòng SHT1x ở những nơi không cần độ chính xác cao về nhiệt độ và độ ẩm. Cảm biến này được sử dụng rộng rãi do chi phí rẻ, dễ lấy dữ liêu thông qua giao tiếp 1 wire, có thể dễ dàng giao tiếp với bất kỳ một vi diều khiển nào như Arduino, Raspberry Pi,…



Để đo nhiệt độ, cảm biến này sử dụng một nhiệt điện trở có hệ số nhiệt độ âm, làm giảm giá trị điện trở của nó khi nhiệt độ tăng. Để có được giá trị điện trở lớn hơn ngay cả đối với sự thay đổi nhỏ nhất của nhiệt độ, cảm biến này thường được làm bằng gốm bán dẫn hoặc polymer.

2.2.3 LDR

Điện trở quang (LDR) là linh kiện điện tử chế tạo bằng chất bán dẫn có điện trơ thay đổi theo mức độ ánh sáng chiếu vào. Đó là điện trở phi tuyến, phi ohmic. Quang trở làm bằng chất bán dẫn trở kháng cao và không có tiếp giáp nào, hoạt động dựa trên hiệu ứng quang điện trong khối vật chất



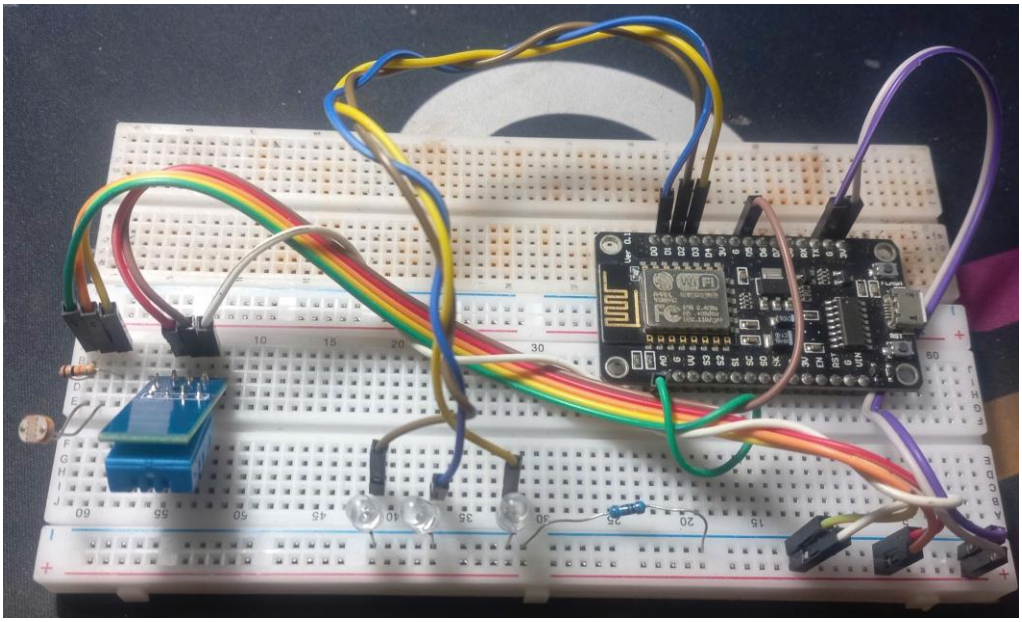
Quang trở được dùng làm cảm biến nhạy sáng trong các mạch dò, ví dụ trong máy đo ánh sáng, mạch xác định mức sáng tối của môi trường để tắt bật đèn tự động. Sơ đồ mạch

2.2.4 Kết nối phần cứng

Các phần cứng được đấu nối theo bảng sau:

Cảm biến	Chân cảm biến	ESP8266
DHT11	VCC	3.3V
	DATA	D5
	GND	G
LDR		A0
		3.3V
Điện trở 10k		A0
		G
LED 1		D1
		G
LED 2		D2
		G
LED 3		D3
		G

Hình ảnh thực thế sau khi đấu nối



2.3 Thiết lập phần mềm

2.3.1 Mosquitto

Đầu tiên tạo một file để mã hoá tên đăng nhập và mật khẩu cho mosquitto. Sau đó chạy mosquitto_passwd để mã hoá mật khẩu. Mosquitto_passwd là một công cụ quản lý các file mật khẩu cho mosquitto MQTT broker. Username và password lưu trong file passwrodfile cách nhau với dấu ':'.

Synopsis

```
mosquitto_passwd [ -H hash ] [ -c | -D ] passwordfile username
mosquitto_passwd [ -H hash ] -b passwordfile username password
mosquitto_passwd -U passwordfile
```

Tạo file broker.conf để config mosquitto. Mosquitto sử dụng port 1883, vậy nên ta cũng cần thêm một inbound rule của tường lửa cho phép mosquitto sử dụng tcp với port 1883.

broker.conf

```
listener 1883
password_file passFile
allow_anonymous true
```

Sau khi cấu hình xong, ta có thể chạy mosquitto bằng câu lệnh sau. Thay thế config file bằng tên file config ta tạo bên trên.

Synopsis

```
mosquitto  [-c  config file ] [ -d | --daemon ] [-p  port number ] [-v]
```

Các topic sẽ được sử dụng trong dự án này

Tên topic	Mô tả
Sensor	- ESP8266 publish dữ liệu cảm biến - Server subcribe để nhận dữ liệu cảm biến
Action	- ESP8266 subcribe để nhận dữ liệu điêu khiển, publish message đã điều khiển - Server publish dữ liệu điều khiển và subcribe để nhận message đã điều khiển của ESP8266
Initial	- ESP8266 publish tín hiệu sau khi được cấp nguồn và subcribe để nhận dữ liệu trạng thái bật tắt của các thiết bị - Server subcribe để nhậ tín hiệu ESP8266 đã khởi động và publish dữ liệu trạng thái bật tắt của các thiết bị được lưu trong database

2.3.2 Cài đặt ESP8266

a) Các thư viện cần sử dụng

Trong dự án này, em sử dụng một số thư viện sau cho ESP826:

- Thư viện **DHT.h** và **DHT_U.h** là thư viện hỗ trợ đọc dữ liệu nhiệt độ, độ ẩm của cảm biến DHT
- Thư viện **ArduinoJson.h** và **ArduinoJson.hpp** là thư viện hỗ trợ kiểu dữ liệu json. Ta có thể chuyển từ string nhận được thành một json object hoặc ngược lại.
- Thư viện **ESP8266WiFi.h** để cấu hình và kết nối vào mạng wifi
- Thư viện **PubSubClient.h** là thư viện mqtt client cho phép ESP8266 thực hiện subcirbe và public vào các topic.

```
#include <DHT.h>
#include <DHT_U.h>
#include <ArduinoJson.h>
#include <ArduinoJson.hpp>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

b) Hàm setup

Hàm setup() khởi tạo các thành phần phần cứng và kết nối ESP8266 với wifi và broker MQTT. Đầu tiên khởi động serial với baud 115200 bps và khởi động cảm biến dht. Sau đó cài đặt chế độ OUTPUT cho các chân điều khiển Led, quạt và relay.

```
void setup() {
    Serial.begin(115200);
    dht.begin();

    pinMode(led, OUTPUT);
    pinMode(fan, OUTPUT);
    pinMode(relay, OUTPUT);

    connectToWiFi();
    mqtt_client.setServer(mqtt_broker, mqtt_port);
    mqtt_client.setCallback(mqttCallback);
    connectToMQTTBroker();
}
```

c) Hàm loop

Trong hàm loop thực hiện kiểm tra và kết nối với MQTT broker, hàm mqtt_client.loop() duy trì kết nối MQTT. Trong hàm này, cứ mỗi 2 giây sẽ đọc và publish dữ liệu cảm biến một lần.

```
void loop() {
  if (!mqtt_client.connected()) {
      connectToMQTTBroker();
  }
  mqtt_client.loop();

  unsigned long currentMillis = millis();
  if (currentMillis - lastPublishTime >= publishInterval) {
    lastPublishTime = currentMillis;
    publishSensorData();
  }
}
```

2.3.3 Cài đặt Server

a) Cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống gồm có 2 bảng:

- Bảng dữ liệu cảm biến

id	temp	humid	light	time
51719	31	92	76	2024-09-02 00:17:48
51718	31	92	68	2024-09-02 00:17:46
51717	31	92	92	2024-09-02 00:17:44
51716	31	92	76	2024-09-02 00:17:42
51715	31	92	81	2024-09-02 00:17:40
51714	31	92	87	2024-09-02 00:17:38
51713	31	92	92	2024-09-02 00:17:36

- Bảng dữ liệu điều khiển

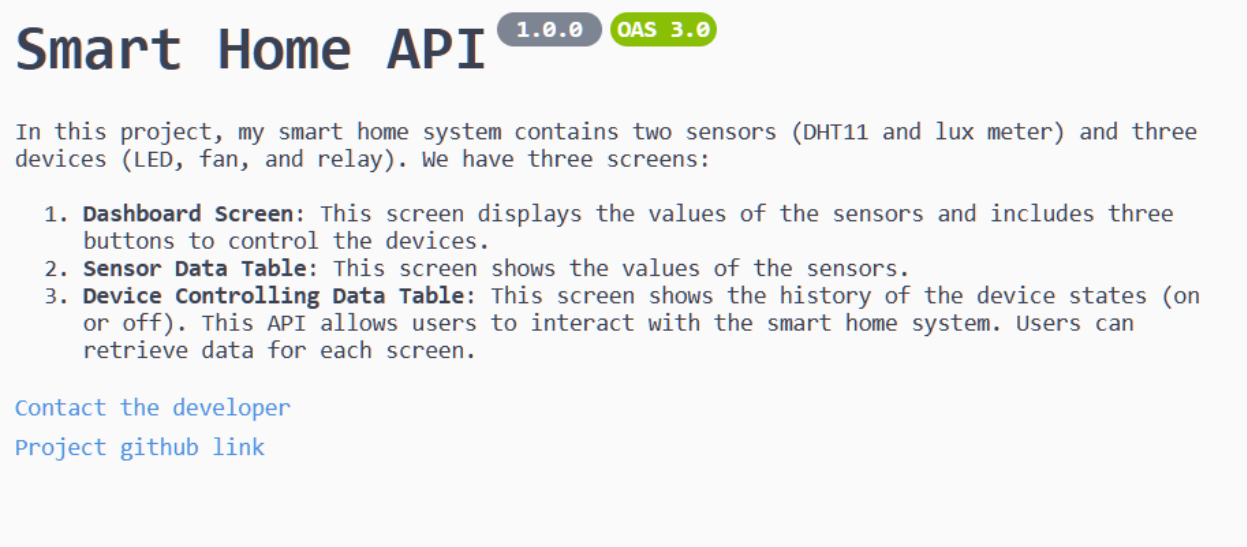
id	fan	led	relay	time
61262	0	1	1	2024-09-02 00:16:46
61261	1	1	1	2024-09-01 23:26:43
61260	1	1	0	2024-09-01 23:25:51
61259	1	1	1	2024-09-01 23:25:40
61258	1	1	1	2024-09-01 23:25:37
61257	0	1	1	2024-09-01 23:23:45
61256	0	0	0	2024-09-01 23:19:34

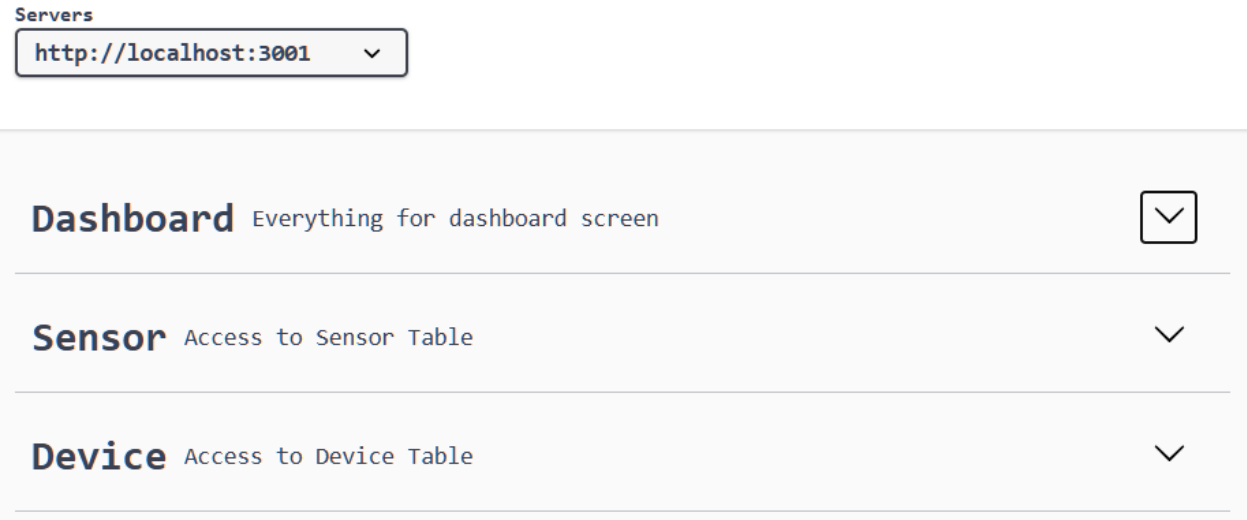
b) Nodejs Express

```
const express = require('express');
const AppRepository = require('./repository');
const MqttClient = require('./mqtt'); // Import the MQTT setup function
const app = express()
// Swagger for API doc
const swaggerUi = require('swagger-ui-express');
const YAML = require('yaml');
const fs = require('fs');
const file = fs.readFileSync('./apidoc.yaml', 'utf-8');
const swaggerDocument = YAML.parse(file);
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument));

// Repository va MQTT Client
const appRepository = new AppRepository();
const client = new MqttClient(appRepository);

// Methods
app.get('/dashboard', (req, res) => appRepository.getDashboardData(res));

app.post('/dashboard', (req, res) => client.publishAction(req, res));

app.get('/table/sensor', (req, res) => appRepository.getSensorTable(req, res));

app.get('/table/action', (req, res) => appRepository.getActionTable(req, res));

const server = app.listen(3001, () => {
    console.log(`Express running → PORT ${server.address().port}`);
});
```

2.3.4 API docs

Truy cập vào http://localhost:3001/api-docs/ để xem API docs sau khi chạy thành công server.

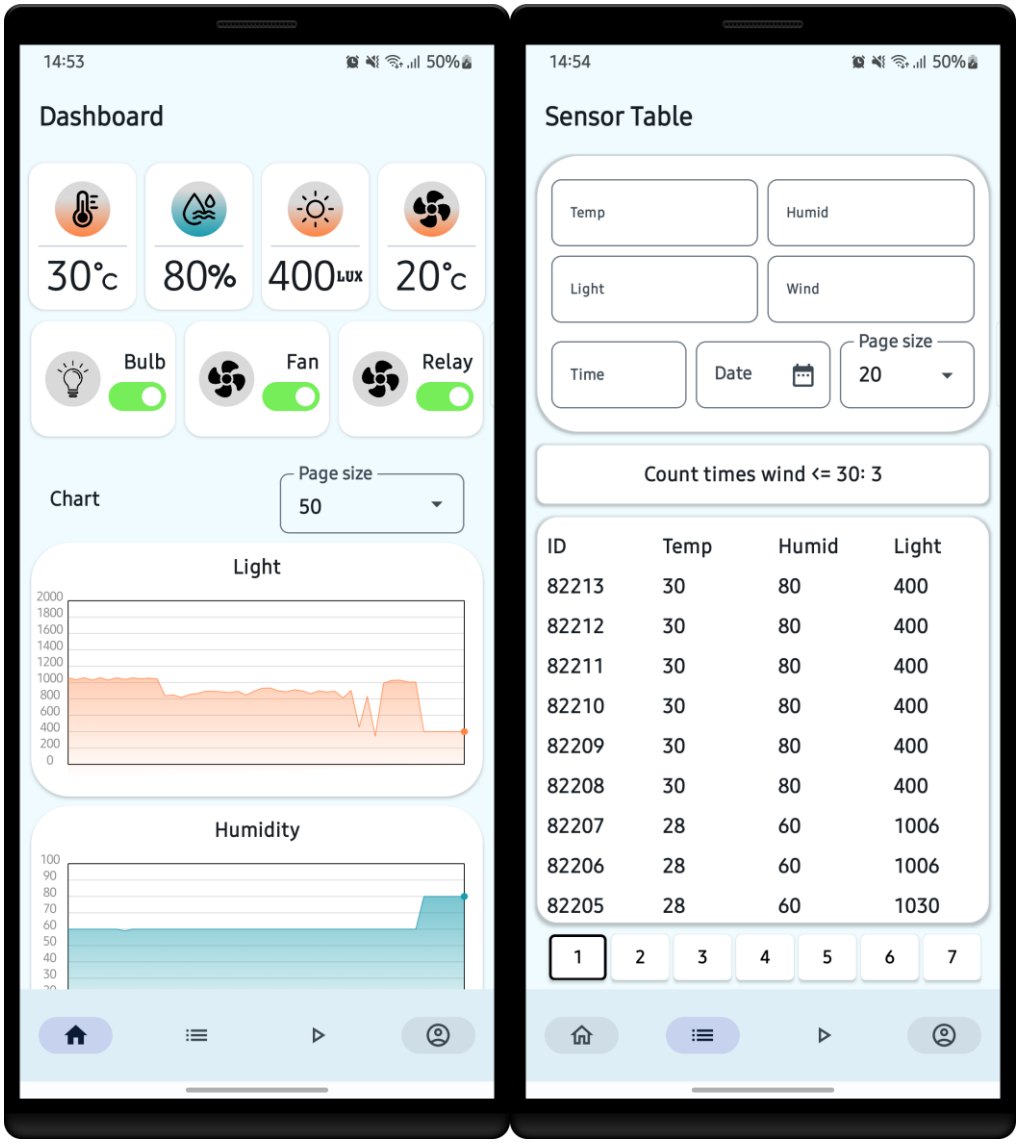


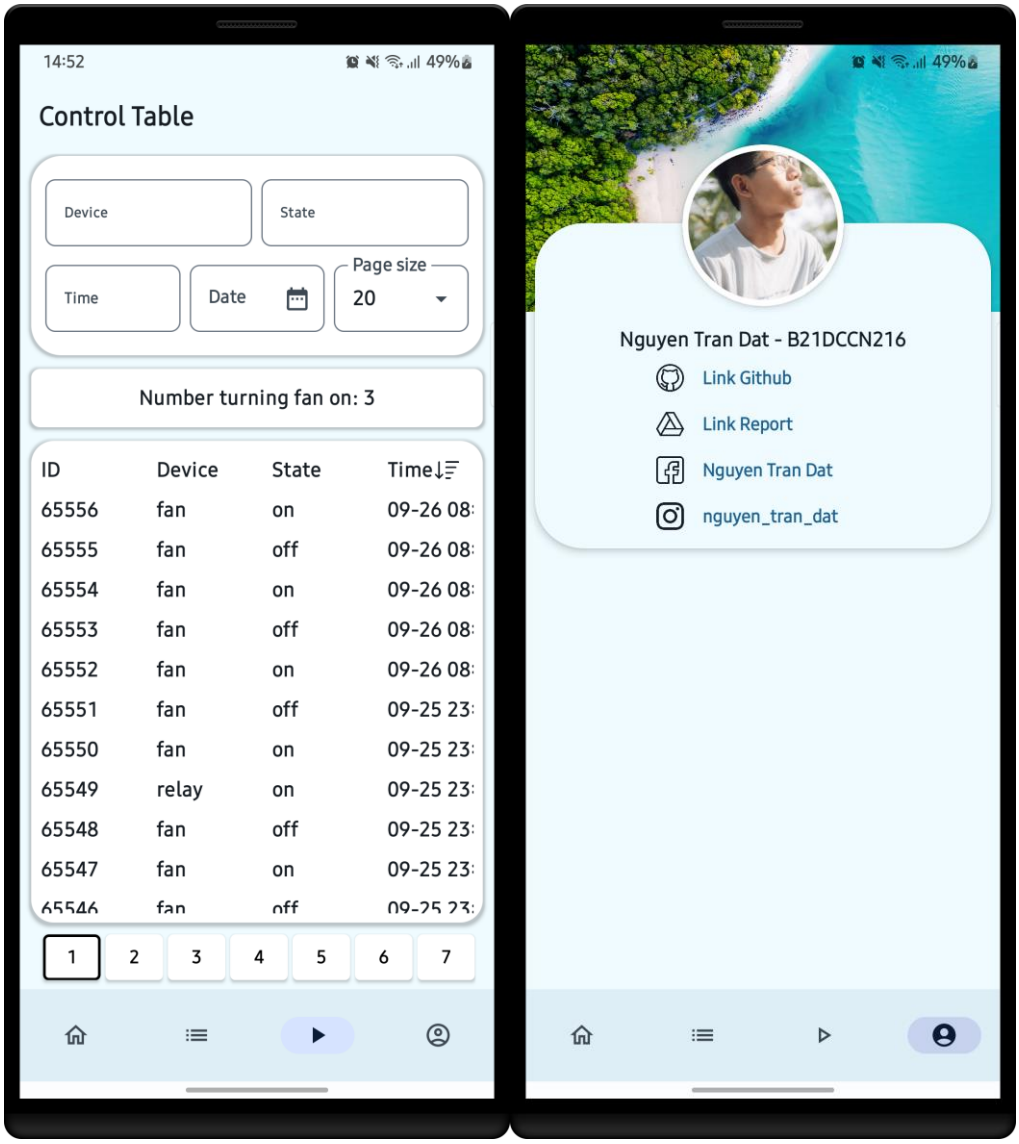


2.3.5 Android App

Người dùng sẽ tương tác với hệ thống này qua một ứng dụng Android được lập trình bằng kotlin. Ứng dụng gồm có 4 giao diện:

- Giao diện Dashboard
- Giao diện bảng dữ liệu cảm biến
- Giao diện bảng dữ liệu điều khiển
- Giao diện profile


14:53
50%
Dashboard
30°c
80%
400LUX
20°c
Bulb
Fan
Relay
Chart
Page size
50
Light
2000
1800
1600
1400
1200
1000
800
600
400
200
0
Humidity
100
90
80
70
60
50
40
30
14:54
50%
Sensor Table
Temp
Humid
Light
Wind
Time
Date
Page size
20
Count times wind <= 30: 3
ID Temp Humid Light
82213 30 80 400
82212 30 80 400
82211 30 80 400
82210 30 80 400
82209 30 80 400
82208 30 80 400
82207 28 60 1006
82206 28 60 1006
82205 28 60 1030
1 2 3 4 5 6 7


14:52
49%
Control Table
Device
State
Time
Date
Page size
20
Number turning fan on: 3
ID
Device
State
Time
65556 fan on 09-26 08
65555 fan off 09-26 08
65554 fan on 09-26 08
65553 fan off 09-26 08
65552 fan on 09-26 08
65551 fan off 09-25 23
65550 fan on 09-25 23
65549 relay on 09-25 23
65548 fan off 09-25 23
65547 fan on 09-25 23
1 2 3 4 5 6 7
49%
Nguyen Tran Dat - B21DCCN216
Link Github
Link Report
Nguyen Tran Dat
nguyen_tran_dat

2.4 Triển khai hệ thống

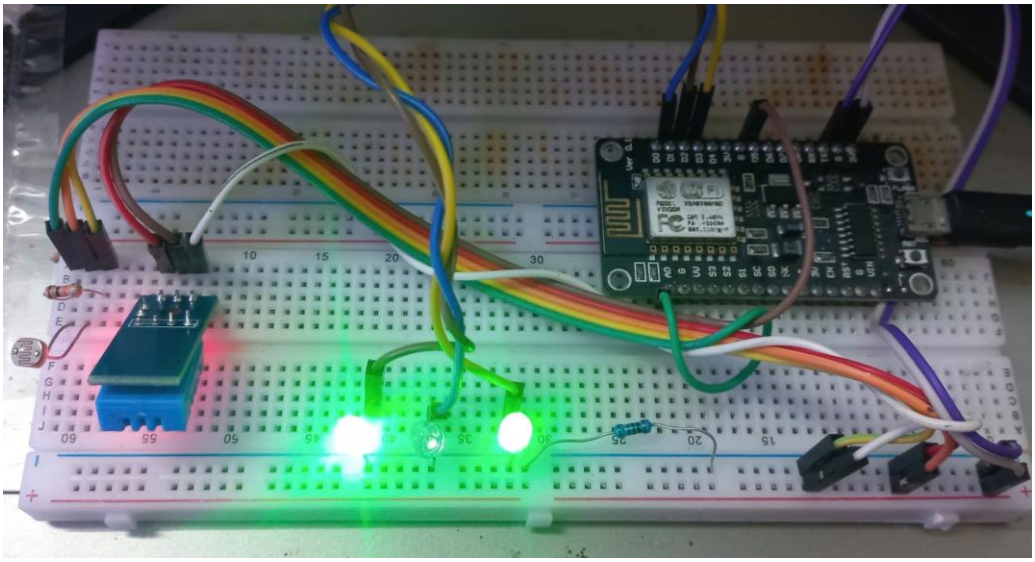
2.4.1 Khởi động MQTT mosquitto

```
C:\Users\ACER>cd c:\mosquitto

c:\mosquitto>mosquitto -v -c broker.conf
1725358641: mosquitto version 2.0.18 starting
1725358641: Config loaded from broker.conf.
1725358641: Opening ipv6 listen socket on port 1883.
1725358641: Opening ipv4 listen socket on port 1883.
1725358641: mosquitto version 2.0.18 running
```

2.4.2 Khởi động Server

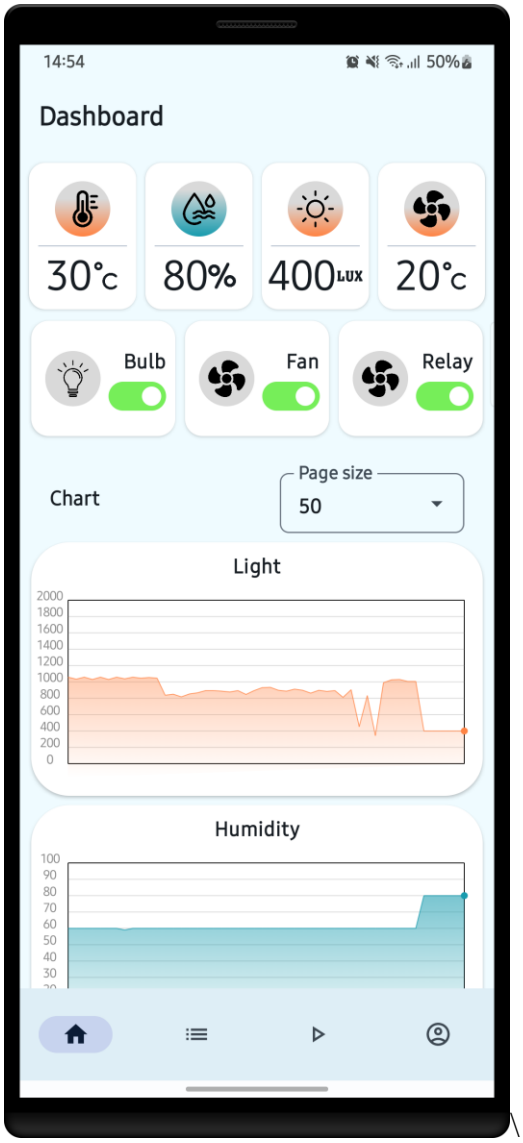
```
PS C:\Users\ACER\Downloads\ky1_nam4\iot\iot_server\node> node smarthome.js
Express running → PORT 3001
connected to mqtt
Connected to database!
```

2.4.3 Cấp nguồn cho ESP8266



2.4.4 Khởi động app

Sau khi khởi động app, màn hình dashboard sẽ hiện lên.



Chương 3: Kết luận

3.1 Kết luận

Trong bài tập lớn này, em dã tiến hành nghiên cứu và xây dựng được một mô hình nhà thông minh đơn giản ứng dụng các công nghệ IoT. Qua quá trình xây dựng và triển khai, hệ thống đã đạt được các mục tiêu mà thầy đặt ra bao gồm khả năng điều khiển thiết bị, theo dõi tình trạng môi trường trong nhà: nhiệt độ, độ ẩm từ ứng dụng web/app.

Thông qua bài tập lớn này, em đã hình dung được một hệ thống IoT đơn giản gồm những gì và ứng dụng thực tế của IoT. Em học được cách triển khai một hệ thống IoT đơn giản.

3.2 Hướng phát triển trong tương lai

Qua đề tài này em xin đề xuất một số hướng phát triển cho hệ thống:

- Nâng cấp cảm biến nhiệt độ, độ ẩm. Cảm biến DHT11 là cảm biến tiện lợi nhưng cho độ chính xác không cao. Chúng ta có thể thay thế bằng những cảm biến khác như SHT1, DHT22,…
- Tích hợp thêm nhiều loại cảm biến khác nhau: Cảm biến lửa, cảm biến khí gas, cảm biến gió,…
- Tích hợp Relay thay cho led để có thể điều khiển các thiết bị chạy điện 220V
- Có thể phát triển các chức năng khác như hẹn giờ bật/tắt thiết bị hay bật tắt thiết bị khi thông số của cảm biến thoả mãn một điều kiện nào đó.
- Có thể mua hosting để chạy server để có thể theo dõi và điều khiển thiết bị khi không có ở nhà.

Tài liệu tham khảo

1. Mosquitto man page | Eclipse Mosquitto
2. esp8266-technical_reference_en.pdf (espressif.com)
3. npm Docs (npmjs.com)
4. API Documentation Tools | Swagger