**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**------------------🙠🕮🙢------------------**

**BÁO CÁO**

**PROJECT II**

**Đề tài: Mô phỏng, điều khiển các thiết bị IOT**

**bằng app điện thoại (smart home)**

**Giáo viên hướng dẫn:** TS. Trần Hải Anh

**Họ tên sinh viên:** Nguyễn Phương Nam

**MSSV:** 20183805

**Lớp, Khóa :** Lớp Kỹ thuật máy tính (IT2 – 01) – K63

1. **Đặt vấn đề và nhiệm vụ của môn học Project**

## Đặt vấn đề

Việc bật tắt các thiết bị điện bằng công tắc với chúng ta là điều đã quá quen thuộc, tuy vậy cũng có khi việc này trở nên khó khăn và bất tiện vì các công tắc điện nằm ở xa nhau, khiến ta phải đi đến từng nơi để bật tắt mất thời gian và công sức. Đặc biệt việc bật tắt công tắc với những người ít khả năng đi lại và vận động hạn chế như người bệnh thấp khớp, người tàn tật, người thị giác kém,... Mặt khác an ninh của ngôi nhà và người sử dụng có thể bị đe dọa bởi những nguy hiểm tiềm tàng như: trộm, dò ga, cháy nổ, điện giật, nồng độ các khí độc hại tăng cao do quá trình đun nấu, sưởi ấm,... Hiện nay trên thế giới cũng như ở Việt Nam đã có các ngôi nhà thông minh (SmartHome). Đây là ngành có tiềm năng phát triển lớn trong tương lai.

## Lý do chọn đề tài

* Vì có thể tạo ra các hệ thống điều khiển theo dõi giám sát từ xa một cách thuận tiện.
* Bảo vệ an toàn cho chủ nhà
* Muốn tìm hiểu hơn về hệ nhúng, phần cứng, app điện thoại.
* Thực hành lại các kiến thức đã học.

## Yêu cầu của Hệ thống

* Xây dựng một mô hình mô phỏng nhà thông minh có thể điều khiển đèn (bật – tắt).
* Tự động bật tắt đèn khi trời sáng, tối.
* Người dùng có thể theo dõi một cách realtime ngôi nhà của mình chỉ bằng 1 app điện thoại đơn giản.

## Các yêu cầu về chức năng

* Hệ thống cần phải login trước khi sử dụng
* Hệ thống nhận dữ liệu từ các cảm biến (nhà) đến app
* Hệ thống gửi lệnh đến các cảm biến (nhà)
* Các gói tin gửi nhận dữ liệu đều có định dạng json

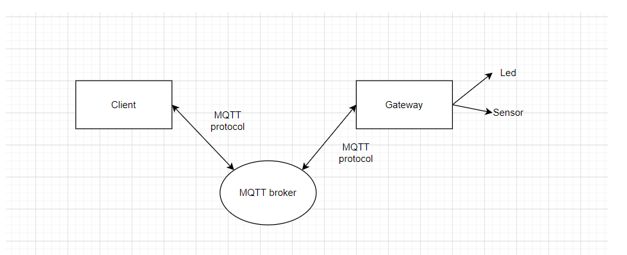
## Các yêu cầu phi chức năng của hệ thống

Hệ thống nhận dữ liệu phải đảm bảo 10s/1 gói tin nhận được để cập nhập trực tiếp đến màn hình user.

1. **Nội dung**

## ***Chương 1: Cơ sở lý thuyết và định hướng giải pháp***

1. **Các lý thuyết**
2. *Sơ đồ tổng quan của hệ thống như sau:*



* Trong các project IOT, luôn có thiết bị đầu (phần cứng) và thiết bị cuối (app user), bởi IOT thực chất là các thiết bị phần cứng đọc dữ liệu từ môi trường, xử lý, và tác động lại môi trường.
* Trong trường hợp này, việc đọc dữ liệu và xử lý dữ liệu từ Sensor gửi về sẽ ở Gateway. Sau khi đọc và xử lý số liệu xong, gateway có thể trực tiếp tác động lại môi trường bằng cách bật, tắt đèn. Ngoài ra Sensor sẽ gửi dữ liệu về cho Client thông qua giao thức mqtt
* Client có thể gửi các command để điều khiển phần cứng (Led) thông qua giao thức mqtt

1. *Giao thức MQTT -  Message Queuing Telemetry Transport*

- MQTT viết tắt của MQ Telemetry Transport (trước đó MQTT được định nghĩa là Message Queuing Telemetry Transport), là giao thức ở tầng ứng dụng chạy trên nền TCP thuộc tập giao thức TCP/IP.

- Để giao thức này hoạt động được, cần có một thành phần trung tâm gọi là **Broker** và các thiết bị/ứng dụng khách còn được gọi là các **Client**, các Client sẽ kết nối đến Broker. Broker và Client sẽ “nói chuyện” với nhau thông qua các gói tin MQTT được xây dựng dựa theo chuẩn OASIS. Tiêu chuẩn này còn định nghĩa các mức quality of service để đảm bảo độ tin cậy khi truyền nhận gói tin, các kịch bản always connected (luôn giữ kết nối) hoặc sometimes connected (thỉnh thoảng kết nối); khả năng mở rộng để hỗ trợ kết nối số lượng lớn thiết bị…

* Giao thức MQTT có kiến trúc theo mô hình Publish/Subscribe giúp truyền tải dữ liệu giữa các thiết bị, ứng dụng với nhau. Dữ liệu trong giao thức này là một chuỗi nhị phân (binary) chứ không phải chuỗi văn bản (text string), được định dạng theo gói tin command hoặc gói tin command acknowledgement.
* Các thành phần lõi trong kiến trúc của MQTT:

|  |  |
| --- | --- |
| **MQTT Broker** | Được cung cấp dưới dạng mã nguồn mở hoặc các phiên bản thương mại, có thể đi kèm với các dịch vụ điện toán đám mây. Công việc của Broker là lọc các tin nhắn dựa trên topic, sau đó phân phối các tin nhắn đến các thiết bị/ứng dụng đã đăng ký topic đó. Các bạn có thể tham khảo một số MQTT Broker như: HiveMQ, Mosquitto, MQTTRoute, Jmqtt, … |
| **MQTT Client** | Là các thiết bị/ứng dụng Client kết nối đến Broker để thực hiện truyền nhận dữ liệu. Hiện nay có rất nhiều mã nguồn mở MQTT Client được viết dưới nhiều ngôn ngữ khác nhau như HiveMQ MQTT Client được phát triển dựa trên ngôn ngữ Java, Eclipse Paho dựa trên C/C++, Python |
| **Topic** | Mỗi MQTT Client thực hiện truyền/nhận dữ liệu với nhau thông qua các Topic được quản lý bởi Broker. Một Client đăng ký nhận dữ liệu được gọi là một Subcriber còn một Client gửi dữ liệu đi được gọi là một Publisher. Để nhận dữ liệu từ Publisher, đầu tiên Subcriber phải subscribe (đăng ký theo dõi) đến một Topic, sau đó bất cứ Client nào publish dữ liệu đến đúng Topic, thì Broker sẽ lọc và chuyển tiếp gói tin đến đúng Subscriber đó. Một Client có thể subscribe hoặc publish đến nhiều Topic khác nhau. |

1. *Thiết bị phần cứng – ESP32*

* Là thiết bị cho phép tương tác trực tiếp với phần cứng, có thể kết nối đến internet
* Cấu hình thiết bị:
* CPU
* CPU: Xtensa Dual-Core LX6 microprocessor.
* Chạy hệ 32 bit
* Tốc độ xử lý 160MHZ up to 240 MHz
* Tốc độ xung nhịp đọc flash chip 40mhz --> 80mhz (tùy chỉnh khi lập trình)
* RAM: 520 KByte SRAM
* Hỗ trợ 2 giao tiếp không dây
* Wi-Fi: 802.11 b/g/n/e/i
* Bluetooth: v4.2 BR/EDR and BLE
* Ngoài ra còn rất nhiều cổng giao tiếp mà esp32 hỗ trợ

### **Công nghệ**

Sử dụng thiết bị vào mạng là ESP32

Sử dụng 1 cảm biến ánh sáng



Sử dụng 1 vài đèn Led, điện trở, dây dẫn

Phần gateway

(phần cứng)

Sử dụng thư viện [pubsubclient](https://github.com/knolleary/pubsubclient) để dùng các api thiết lập giao tiếp mqtt

Sử dụng thư viện [ArduinoJson](https://arduinojson.org/) để parse/build gói tin json

App trên android viết bằng java

Phần client

Sử dụng thư viện paho mqtt để thiết lập giao tiếp mqtt

Sử dụng mô hình MVVM (Model – View - ViewModel) kết hợp

với databinding làm mô hình kiến trú

Sử dụng Broker online

Phần

MQT Broker

Địa chỉ kết nối đến broker: **broker.hivemq.com**

TCP port: 1883

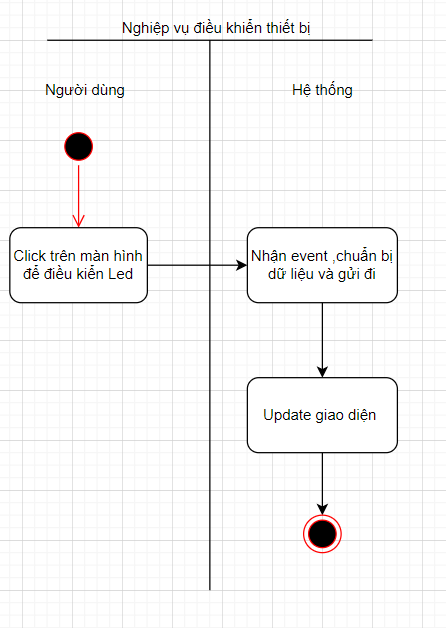
### **Công cụ sử dụng**

|  |  |
| --- | --- |
| Phần gateway | Sử dụng **arduino ide** để lập trình, upload cho ESP32 |
| Phần client | Sử dụng Android Studio để lập trình cho client (app điện thoại) |
| Các công cụ debug gói tin | MQTT Box: Là công cụ để theo dõi các gói tin, giả lập gửi gói tin, dùng để theo dõi xem các thành phần đã giao tiếp đúng hay sai, các gói tin có đúng chuẩn định dạng hay không. |

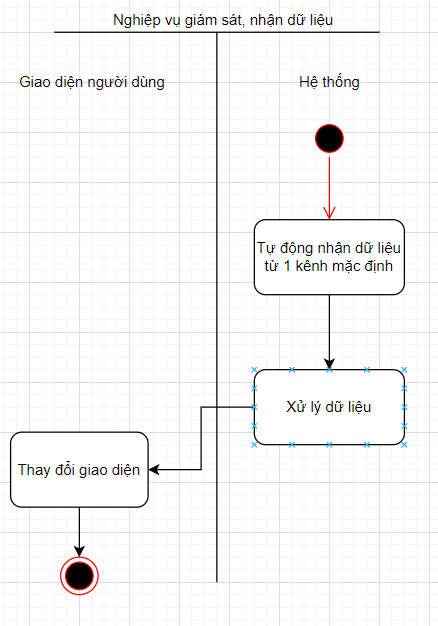
## **Chương 2: Phân tích thiết kế hệ thống**

1. **Phân tích quy trình**

* Hệ thống có 2 quy trình chính:
* Gửi command điều khiển

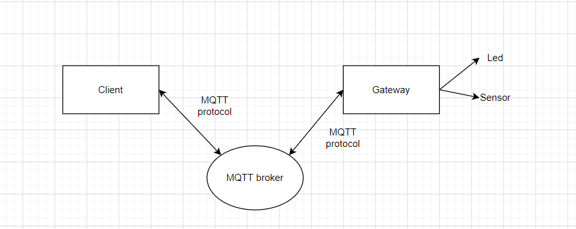


* Nhận dữ liệu về và hiện lên sự thay đổi trên giao diện



1. **Phân tích và thiết kế hệ thống**
2. Kiến trúc tổng quan của hệ thống

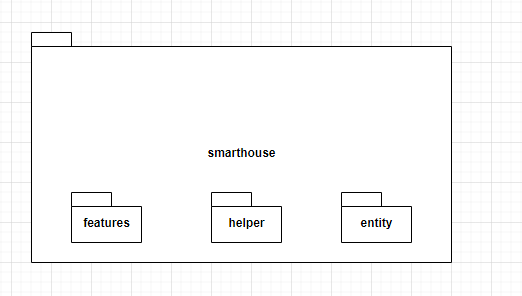
* Sơ đồ hệ thống



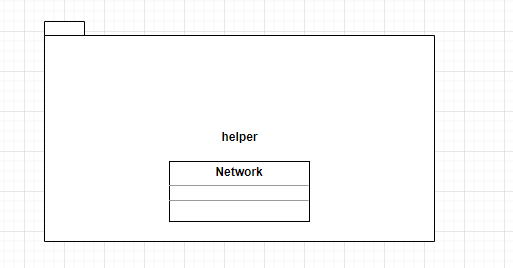
* Mô tả chức năng của hệ thống phía người dùng
* Người dùng sẽ đăng nhập vào hệ thống trước khi quản lý các thiết bị cứng.
* Sau khi đăng nhập có thể theo dõi, giám sát, điều khiển thiết bị phần cứng.
* Các thành phần
* Client: Module biểu diễn cho phần client ở đây là App điện thoại, chịu trách nhiệm theo dõi, điều khiển trực tiếp phần cứng qua giao thức MQTT.
* Gateway: Module biểu diễn cho phần gateway là ESP32, có trách nhiệm kết nối trực tiếp đến đèn, cảm biến, lấy dữ liệu từ cảm biến, điều khiển phần cứng, có thể kết nối internet, giao tiếp với client thông qua giao thức MQTT.

1. App điện thoại

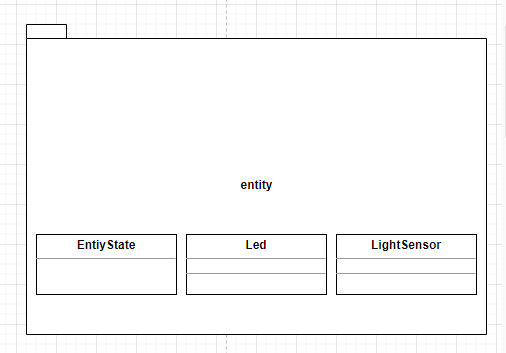
*Mô hình packet*



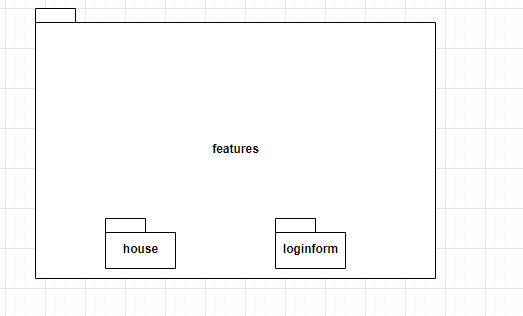
* Packet smarthouse: là packet chứa cả project.
* Packet features: chứa các nghiệp vụ liên quan đến giao diện, xử lý giao diện.
* Packet helper: chứa các nghiệm vụ cho các lớp trong features có thể sử dụng như kết nối đến internet, api giao tiếp với mqtt
* Packet entity: chứa các thực thể phần cứng như Led, Sensor



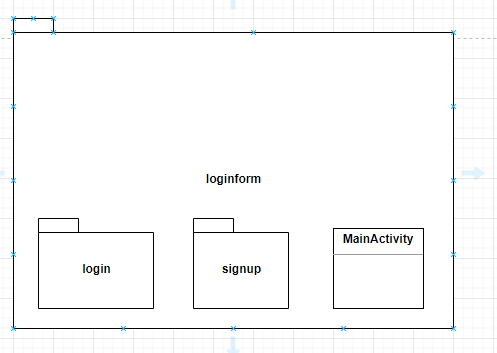
* Lớp Network: chứa các api giao tiếp mqtt



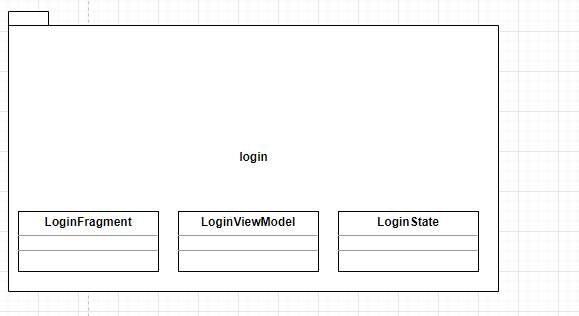
* Enum EntityState: chứa các trạng thái của các entity khác như LED\_ON, LED\_OFF, LIGHT\_SENSOR\_HIGH, LIGHT\_SENSOR\_LOW
* Lớp Led: là lớp thực thể map thẳng vào thiết bị phần cứng Led, chứa trạng thái phần cứng. Khi trạng thái này thay đổi sẽ tự động cập nhập lên giao diện
* Lớp LightSensor: là lớp thực thể map thẳng vào thiết bị phần cứng Cảm biến ánh sáng, chứa trạng thái phần cứng. Khi trạng thái này thay đổi sẽ tự động cập nhập lên giao diện



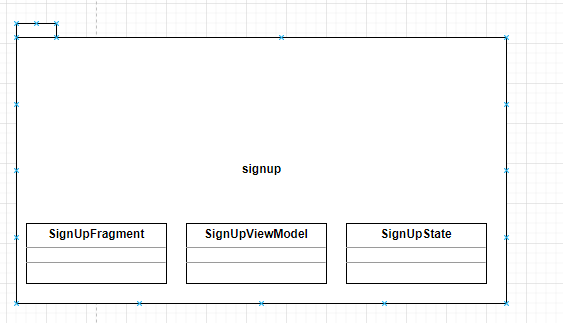
* Packet house: ở trong packet features, đại diện cho 1 activity. Chứa các giao diện, quản lý nhà
* Packet loginform: ở trong packet features, đại diện cho 1 activity. Chứa các giao diện, xử lý giao diện cho phần login, signup



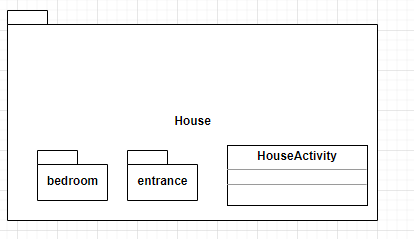
* Packet login: chứa giao diện, quản lý giao diện của màn hình login
* Packet signup: chứa giao diện, quản lý giao diện của màn hình signup
* Lớp MainActivity: là nơi bắt đầu chạy của hệ thống, chịu tránh nhiệm quản lý 2 màn hình giao diện login và signup



* Lớp LoginFragment: có nhiệm vụ quản lý phần User Interface (ui)
* Lớp LoginViewModel: có nhiệm vụ xử lý các nghiệm vụ do tầng ui đưa xuống như verify user, password.
* Lớp LoginState: chứa các state của màn hình login như LOGIN, LOGIN\_FAIL



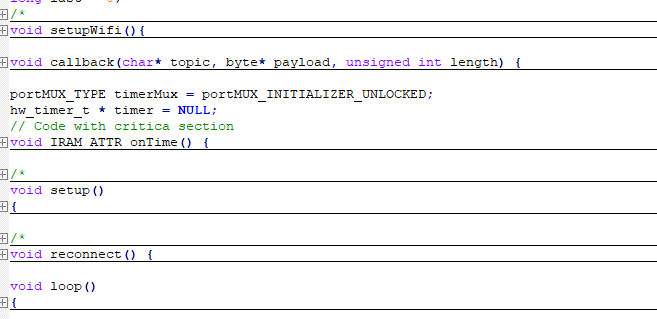
* Lớp SignUpFragment: có nhiệm vụ quản lý ui
* Lớp SignUpViewModel: có nhiệm vụ xử lý event do ui đưa xuống.
* Lớp SignUpState: chứa các trạng thái của màn hình signup



* Packet bedroom: nằm trong packet house, chứa ui, quản lý ui của màn hình bedroom
* Packet entrance: nằm trong packet house, chứa ui, quản lý ui của màn hình entrance
* Lớp HouseActivity: có nhiệm vụ quản lý 2 màn hình bedroom và entrance

1. Gateway

* Các hàm dùng để tạo nên gateway

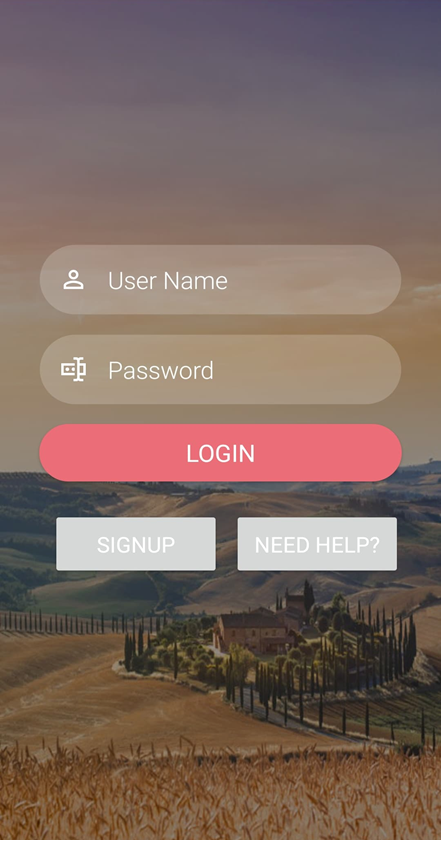


* Hàm setup: là hàm chạy đầu tiên của hệ thống trên esp32, có nhiệm vụ khởi tạo internet, kết nối wifi, khởi tạo mqtt, gắn callback cho mqtt. Hàm này sẽ gọi các hàm con để thực hiện các nhiệm vụ trên.
* Hàm setupWifi: có nhiệm vụ kết nối đến wifi, chỉ trả về luồng chính khi đã kết nối thành công.
* Hàm callback: là hàm được gán callback của mqtt. Dùng để bắt các gói tin mqtt và xử lý
* Hàm reconnect: là hàm được gán callback của mqtt. Dùng để kết nối lại đến mqtt broker khi kết nối bị gián đoạn giữa chừng.
* Hàm IRAM\_ATTR\_ontime: là hàm được gán với 1 timer. Cứ 10s sẽ chạy 1 lần. Có tránh nhiệm đọc dữ liệu từ sensor và xử lý trực tiếp (trong trường hợp này là đọc dữ liệu từ cảm biến ánh sáng, xử lý số liệu đó và sẽ bật tắt đèn tùy theo độ mạnh yếu của ánh dữ liệu thu được)
* Hàm loop: là hàm chạy chính của ESP32, sau 1 khoảng thời gian nhất định (10s) sẽ đọc dữ liệu từ cảm biến, đèn và gửi qua mqtt.

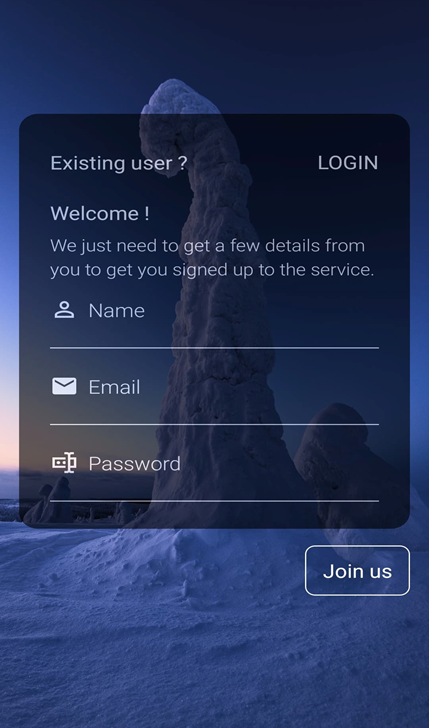
### **Thiết kế giao diện**

1. Giao diện app

* Màn hình login



* Màn hình signup



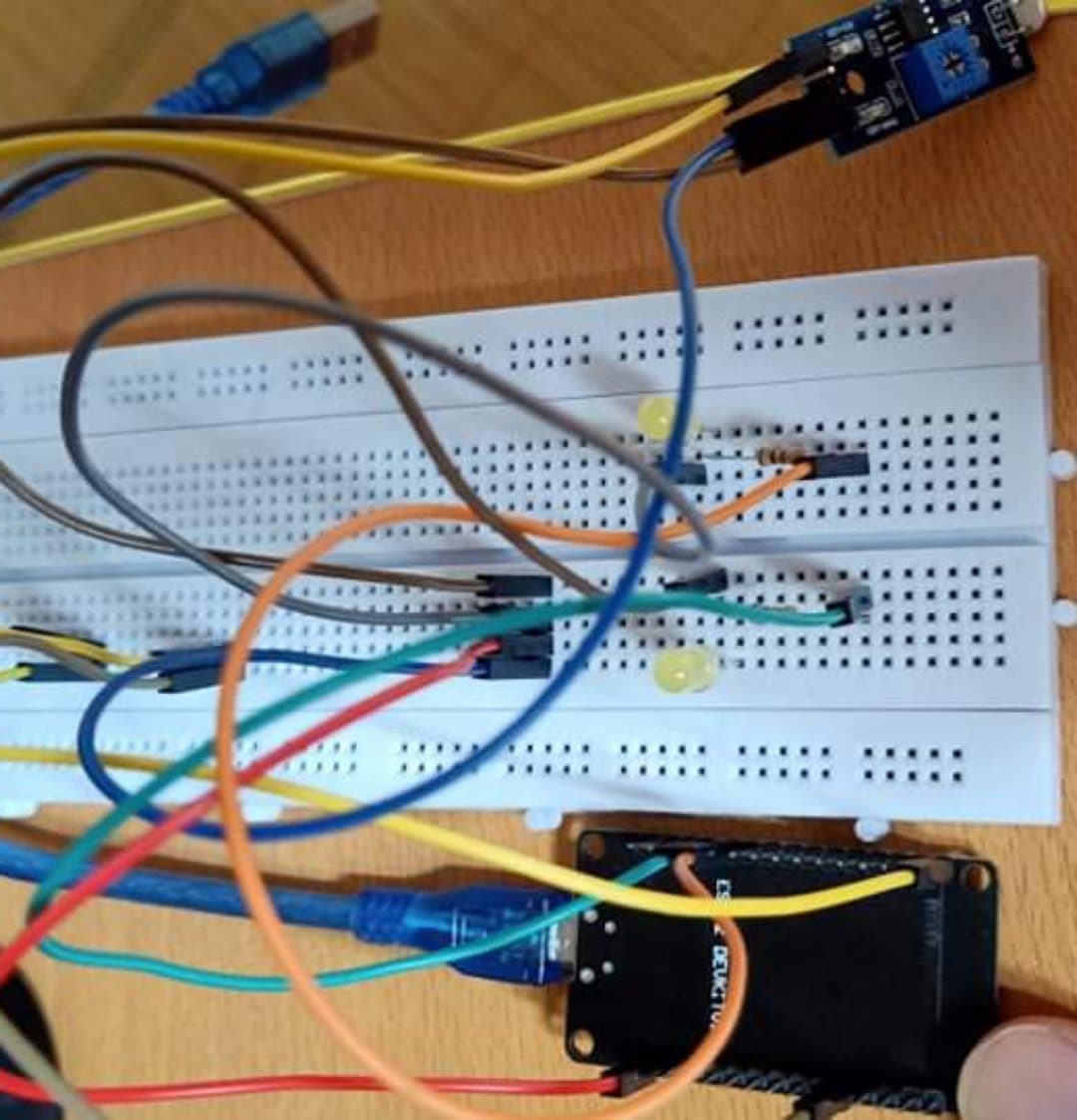
* Màn hình bedroom



* Màn hình entrance



1. Hình ảnh mạch điện



* Hình ảnh mạch trên có chứa 2 đèn Led, 1 cảm biến ánh sáng
* 1 đèn Led tượng trưng cho đèn led trong phòng ngủ, user có thể bật tắt qua app điện thoại
* 1 đèn Led tượng trưng cho đèn led ngoài cửa ra vào, user có thể xem trạng thái của đèn này trên màn hình entrance. Đèn này sẽ bật khi cảm biến ánh sáng nhận giá trị quá tối, sẽ tắt khi cảm biến nhận lại giá trị sáng.

## **3. Chương 3: Thử nghiệm**

- Video demo

# **III. Kết luận**

## *Kết luận về hệ thống*

* Hệ thống trên đã có thể mô phỏng được quá trình điều khiển từ thiết bị đầu cuối đến thiết bị phần cứng.
* Đáp ứng đủ yêu cầu về mặt truyền tải dữ liệu.
* Có thể mở rộng thêm về mặt bảo mật cho ngôi nhà như các kết nối cần được mã hóa.

## *2. Các ý tưởng nâng cấp hệ thống*

* Cần có 1 server để lưu dữ liệu từ gateway.
* Cần thêm kết nối http đến server để verify user quản lý ngôi nhà.
* Cần có thêm 1 màn hình trước khi vào quản lý ngôi nhà: màn hình chọn nhà để quản lý. Vì 1 user có thể có nhiều nhà. Ứng dụng sẽ mở rộng cho nhiều user sử dụng.
* Cần thiết kế hệ thống database để lưu trữ dữ liệu, các bảng về user,data, home.