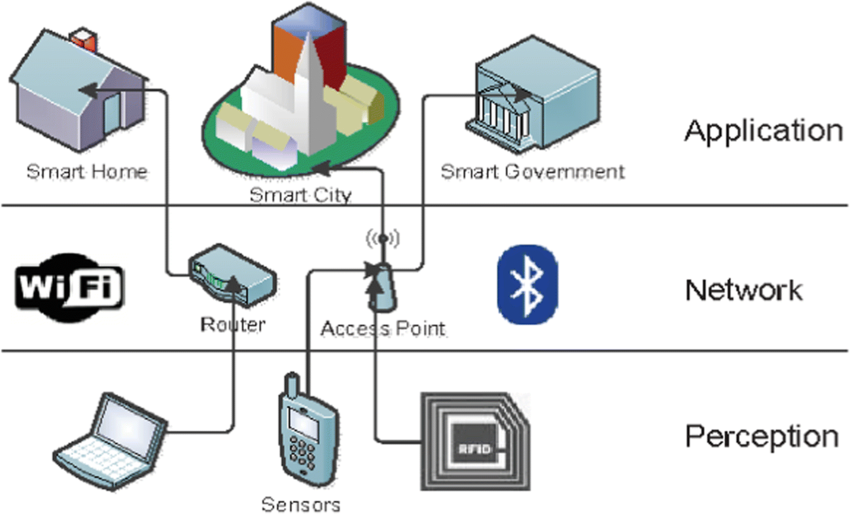
1. Tổng quan
   1. Đặt vấn đề:

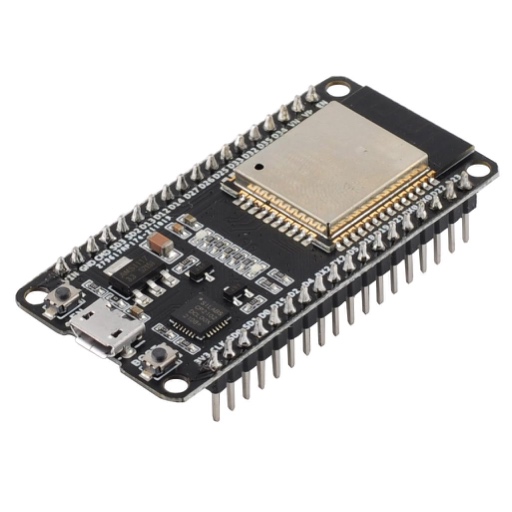
* Trong bối cảnh phát triển mạnh mẽ của công nghệ IoT (Internet of Things), việc tự động hóa và điều khiển từ xa các thiết bị trong đời sống hàng ngày đang trở thành nhu cầu phổ biến. An ninh và sự tiện lợi là những yêu cầu quan trọng đối với các hệ thống cửa ra vào trong các hộ gia đình, văn phòng, và các khu vực cần bảo vệ. Việc sử dụng các công nghệ mới như cảm biến và điều khiển không dây không chỉ giúp gia tăng bảo mật mà còn mang lại sự tiện lợi cho người dùng.
  1. Mục tiêu:
* **Xây dựng hệ thống khóa cửa thông minh** sử dụng ESP32 làm vi điều khiển trung tâm để điều khiển khóa điện và các cảm biến liên quan, đáp ứng nhu cầu bảo mật và quản lý ra vào một cách an toàn, tiện lợi.
* **Phát triển khả năng điều khiển từ xa qua mạng Wi-Fi:** Cho phép người dùng mở khóa hoặc khóa cửa từ xa qua ứng dụng di động hoặc giao diện web, đồng thời nhận thông báo về trạng thái của cửa và hệ thống.
* **Tích hợp cảm biến an ninh** như cảm biến vân tay, cảm biến RFID, và cảm biến chuyển động SR505, để tăng cường mức độ bảo mật và cho phép xác thực đa yếu tố.
* **Thiết lập cơ chế cảnh báo an ninh:** Gửi thông báo đến người dùng khi có bất kỳ hành vi xâm nhập trái phép nào, như việc sử dụng thẻ không hợp lệ hoặc phát hiện chuyển động bất thường gần khu vực cửa.

1. Cơ sở lý thuyết
   1. Tổng quan về IoT:



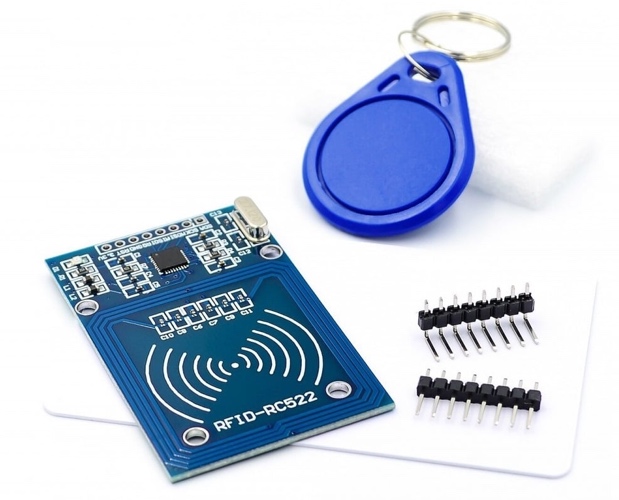
*Hình 2.1: Tổng quan về IoT*

* Internet of Things (IoT) là một hệ thống kết nối các thiết bị, máy móc, và cảm biến qua mạng internet để thu thập và chia sẻ dữ liệu mà không cần sự can thiệp trực tiếp của con người. Những thiết bị này có thể là các thiết bị gia dụng, cảm biến môi trường, phương tiện giao thông, hoặc bất kỳ thiết bị nào có khả năng thu thập thông tin và giao tiếp qua mạng. Mục tiêu của IoT là tạo ra một thế giới kết nối thông minh, nơi các thiết bị có thể phối hợp tự động, tối ưu hóa quy trình và mang lại nhiều lợi ích tiện nghi cho cuộc sống. Trong các lĩnh vực như nhà thông minh, thành phố thông minh, y tế, sản xuất, và nông nghiệp, IoT đã và đang mang lại những cải tiến đáng kể, giúp giám sát và quản lý từ xa một cách hiệu quả và tiết kiệm tài nguyên. Dù vậy, IoT cũng gặp phải các thách thức lớn về bảo mật và quyền riêng tư, khi một lượng lớn dữ liệu cá nhân và hoạt động cần được bảo vệ khỏi các mối đe dọa an ninh mạng. Bên cạnh đó, việc tương thích giữa các thiết bị từ nhiều nhà sản xuất khác nhau và quản lý khối lượng dữ liệu khổng lồ cũng đòi hỏi những tiêu chuẩn và công nghệ đồng bộ. IoT là một xu hướng công nghệ mang tính đột phá, mở ra nhiều cơ hội ứng dụng mới nhưng cũng đòi hỏi hạ tầng và giải pháp bảo mật mạnh mẽ để phát triển bền vững.
  1. Tổng quan về web server:
* Web, viết tắt của World Wide Web (WWW), là một hệ thống thông tin toàn cầu kết nối hàng tỷ trang web và tài nguyên trên Internet, cho phép người dùng truy cập và chia sẻ thông tin một cách nhanh chóng và dễ dàng. Ra đời từ những năm 1990, web đã phát triển vượt bậc, trở thành nền tảng phổ biến nhất cho việc truyền tải thông tin, giao tiếp, và tương tác trên toàn thế giới. Các trang web được tạo thành từ các tài liệu HTML, CSS và JavaScript, hiển thị thông tin qua trình duyệt web, giúp người dùng dễ dàng truy cập nội dung từ bất kỳ đâu chỉ với một thiết bị có kết nối Internet. Bên cạnh đó, các công nghệ web như HTTP và HTTPS giúp dữ liệu được truyền tải một cách an toàn và nhanh chóng. Web đã phát triển từ những trang tĩnh ban đầu thành các ứng dụng web động và phức tạp, có khả năng tương tác thời gian thực, hỗ trợ mua sắm trực tuyến, mạng xã hội, công việc từ xa và học tập trực tuyến. Với sự đổi mới không ngừng, web đã và đang trở thành nền tảng trọng yếu của kỷ nguyên số, phục vụ đa dạng nhu cầu cá nhân và doanh nghiệp, góp phần thay đổi cách con người kết nối và tiếp cận tri thức.
  1. Tổng quan về ESP32:
* ESP32 là một vi điều khiển tích hợp Wi-Fi và Bluetooth phổ biến, do Espressif Systems phát triển, mang lại giải pháp mạnh mẽ và linh hoạt cho các ứng dụng IoT và các hệ thống nhúng. Với bộ vi xử lý lõi kép Xtensa LX6 có tốc độ lên đến 240MHz, ESP32 cung cấp hiệu suất xử lý cao và khả năng đa nhiệm mạnh mẽ, đáp ứng tốt yêu cầu của nhiều tác vụ phức tạp trong các ứng dụng IoT như điều khiển thiết bị từ xa, thu thập và xử lý dữ liệu từ các cảm biến. Ngoài kết nối Wi-Fi và Bluetooth BLE, ESP32 còn hỗ trợ nhiều giao thức giao tiếp như SPI, I2C, UART, và PWM, giúp dễ dàng tương thích với các thiết bị ngoại vi và cảm biến khác nhau. Bên cạnh đó, ESP32 tích hợp các chế độ tiết kiệm năng lượng hiệu quả, tối ưu hóa cho các hệ thống hoạt động bằng pin với khả năng hoạt động ổn định trong thời gian dài. Nhờ khả năng linh hoạt và chi phí thấp, ESP32 là lựa chọn lý tưởng trong các dự án IoT, nhà thông minh, tự động hóa công nghiệp và nhiều ứng dụng khác, đồng thời cũng dễ lập trình thông qua các nền tảng như Arduino IDE và ESP-IDF.



*Hình 2.1 ESP*

* 1. Giới thiệu cảm biến:
     1. Cảm biến RFID-RC522:
* Cảm biến RFID-RC522 là một module đọc thẻ RFID phổ biến, được sử dụng rộng rãi trong các ứng dụng nhận dạng và kiểm soát truy cập. Module này hoạt động dựa trên công nghệ RFID (Radio Frequency Identification), cho phép nhận diện các thẻ RFID không tiếp xúc thông qua sóng radio. RC522 có khả năng hoạt động ở tần số 13.56 MHz và hỗ trợ các thẻ RFID theo tiêu chuẩn ISO/IEC 14443, mang lại hiệu suất đọc nhanh và chính xác. Với kích thước nhỏ gọn và dễ dàng tích hợp, cảm biến này thường được sử dụng trong các dự án IoT, như hệ thống kiểm soát ra vào, thanh toán điện tử, và quản lý kho. RC522 có giao tiếp SPI, I2C và UART, giúp dễ dàng kết nối với các vi điều khiển như Arduino hoặc ESP32. Đặc biệt, với các thư viện lập trình phong phú, việc sử dụng cảm biến RFID-RC522 trở nên đơn giản và thuận tiện, mở ra nhiều cơ hội cho các ứng dụng tự động hóa và cải thiện hiệu quả quản lý trong nhiều lĩnh vực khác nhau.



*Hình 2.2 RFID RC522*

* + 1. Cảm biến AS 608:
* Cảm biến AS608 là một module nhận diện vân tay tiên tiến, được thiết kế để cung cấp khả năng xác thực sinh trắc học an toàn và đáng tin cậy cho các ứng dụng bảo mật. Với công nghệ quét vân tay quang học, AS608 có khả năng lưu trữ đến 1.500 mẫu vân tay và thực hiện việc nhận diện nhanh chóng, thường chỉ trong vòng chưa đầy một giây. Module này hỗ trợ giao tiếp qua UART, giúp dễ dàng tích hợp vào các vi điều khiển như Arduino hoặc ESP32, cho phép lập trình viên xây dựng các hệ thống kiểm soát truy cập thông minh. Ngoài khả năng xác thực vân tay, AS608 còn đi kèm với các tính năng như lưu trữ dữ liệu và quản lý người dùng, giúp đơn giản hóa việc quản lý quyền truy cập. Thiết kế nhỏ gọn và dễ sử dụng của cảm biến này làm cho nó trở thành một lựa chọn lý tưởng cho các ứng dụng trong lĩnh vực an ninh, nhà thông minh và hệ thống kiểm soát ra vào, đồng thời nâng cao tính bảo mật và tiện lợi cho người dùng.



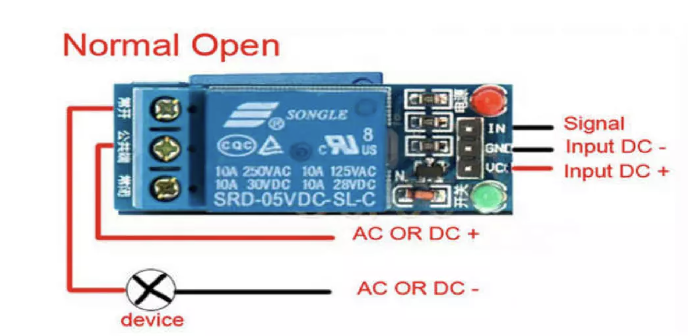
*Hình 2.3 AS 608*

* + 1. Cảm biến SR 505:
* Cảm biến AS608 là một module nhận diện vân tay tiên tiến, được thiết kế để cung cấp khả năng xác thực sinh trắc học an toàn và đáng tin cậy cho các ứng dụng bảo mật. Với công nghệ quét vân tay quang học, AS608 có khả năng lưu trữ đến 1.500 mẫu vân tay và thực hiện việc nhận diện nhanh chóng, thường chỉ trong vòng chưa đầy một giây. Module này hỗ trợ giao tiếp qua UART, giúp dễ dàng tích hợp vào các vi điều khiển như Arduino hoặc ESP32, cho phép lập trình viên xây dựng các hệ thống kiểm soát truy cập thông minh. Ngoài khả năng xác thực vân tay, AS608 còn đi kèm với các tính năng như lưu trữ dữ liệu và quản lý người dùng, giúp đơn giản hóa việc quản lý quyền truy cập. Thiết kế nhỏ gọn và dễ sử dụng của cảm biến này làm cho nó trở thành một lựa chọn lý tưởng cho các ứng dụng trong lĩnh vực an ninh, nhà thông minh và hệ thống kiểm soát ra vào, đồng thời nâng cao tính bảo mật và tiện lợi cho người dùng.



*Hình 2.4 SR 505*

* 1. Giới thiệu các thiết bị ngoại vi:
     1. Relay:
* Relay là một loại thiết bị điện tự động mà tín hiệu đầu ra thay đổi nhảy cấp khi tín hiệu đầu vào đạt những giá trị xác định. Relay là thiết bị điện dùng để đóng cắt mạch điện điều khiển, bảo vệ và điêu khiên sự làm việc của mạch điện động lực.



*Hình 2.5 Relay*

* + 1. Đèn:
* Đèn LED (Light Emitting Diode) là một loại đèn sử dụng đi-ốt phát quang để tạo ra ánh sáng. Đèn LED nổi bật nhờ hiệu suất chiếu sáng cao, tuổi thọ dài và tiêu thụ ít điện năng hơn so với các loại đèn truyền thống như đèn sợi đốt và đèn huỳnh quang. Đèn LED được ứng dụng rộng rãi trong các lĩnh vực như chiếu sáng dân dụng, biển quảng cáo, điện tử và ô tô.

A close-up of a light emitting diode

Description automatically generated

*Hình 2.6 Đèn led*

* + 1. Khoá điện:
* Khóa điện solenoid là một loại khóa sử dụng nam châm điện (solenoid) để điều khiển cơ cấu đóng mở của khóa. Khi cấp nguồn điện, nam châm điện bên trong khóa tạo ra từ trường, kéo hoặc đẩy một chốt để mở khóa. Khi ngắt điện, chốt khóa trở lại vị trí ban đầu, khóa lại cơ cấu. Khóa solenoid thường được ứng dụng trong các hệ thống an ninh, cửa tự động, tủ điện tử, và các thiết bị tự động hóa.



*Hình 2.7 Khoá điện solenoid*

* + 1. Loa buzzer:
* Loa Buzzer (còn gọi là còi báo hoặc buzzer) là một thiết bị phát ra âm thanh đơn giản, thường được sử dụng trong các hệ thống điện tử để tạo âm báo hoặc tín hiệu cảnh báo. Buzzer hoạt động bằng cách tạo ra rung động trong không khí, từ đó phát ra âm thanh. Loa buzzer có thể phát ra âm thanh với tần số và âm lượng khác nhau, tùy thuộc vào loại và cấu tạo của nó.



*Hình 2.8 Loa buzzer*

* + 1. Nguồn Adapter 5v:
* Adapter (hay bộ chuyển đổi) là một thiết bị điện tử được sử dụng để chuyển đổi nguồn điện hoặc tín hiệu để phù hợp với yêu cầu của các thiết bị khác. Adapter phổ biến nhất là loại dùng để chuyển đổi điện áp, ví dụ như từ 220V AC (dòng điện xoay chiều) thành 5V hoặc 12V DC (dòng điện một chiều) để cung cấp năng lượng cho các thiết bị điện tử như laptop, điện thoại di động, đèn LED và các thiết bị ngoại vi khác.

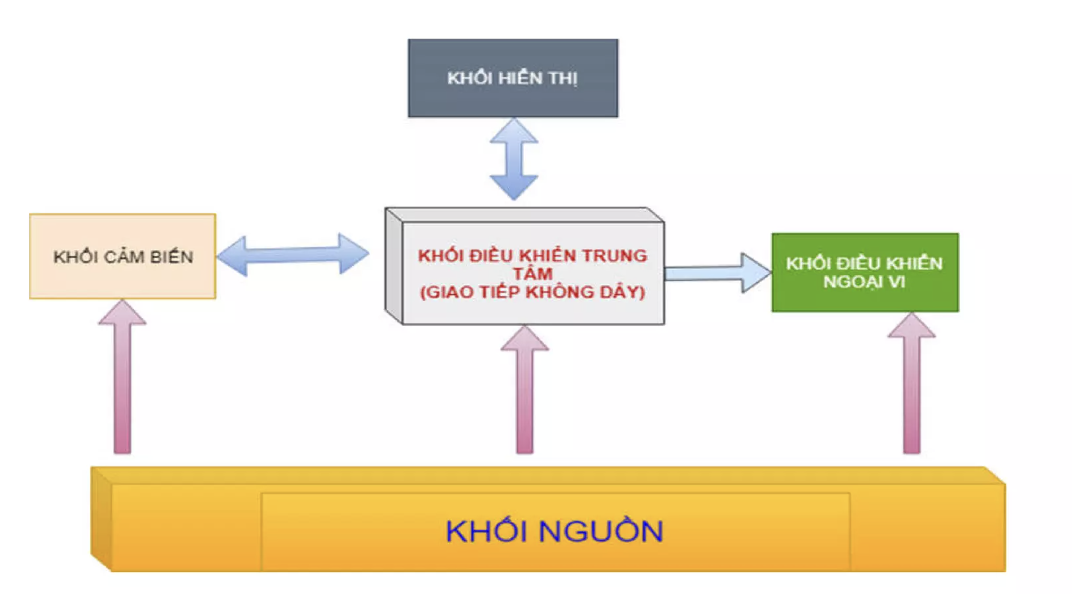


*Hình 2.9 Nguồn adapter*

* 1. Giới thiệu Firebase:
* Firebase là một nền tảng phát triển ứng dụng di động và web của Google, cung cấp các công cụ và dịch vụ để hỗ trợ quá trình phát triển, vận hành và tối ưu hóa ứng dụng. Firebase đặc biệt phổ biến với các lập trình viên nhờ vào khả năng cung cấp các dịch vụ mạnh mẽ như cơ sở dữ liệu thời gian thực, xác thực người dùng, lưu trữ đám mây, và nhiều công cụ hỗ trợ khác mà không cần phải xây dựng backend từ đầu.

1. Tính toán và thuyết kế:
   1. Giới thiệu:

* Đề tài thuyết kế hệ thống khoá thông minh hiển thị dữ liệu trên web server và firebase bao gồm:
* Các cảm biến thu nhập dữ liệu gửi qua esp32.
* Esp32 sẽ gửi dữ liệu trực tiếp lên firebase và hiển thị dữ liệu lên webserver.
  1. Tính toán và thuyết kế hệ thống:
     1. Thuyết kế sơ đồ khối hệ thống:
* Hệ thống bao gồm 5 khối ghép lại với nhau và tạo nên một hệ thống, được trình bày trong sơ đồ khối hình 3.1 như sau

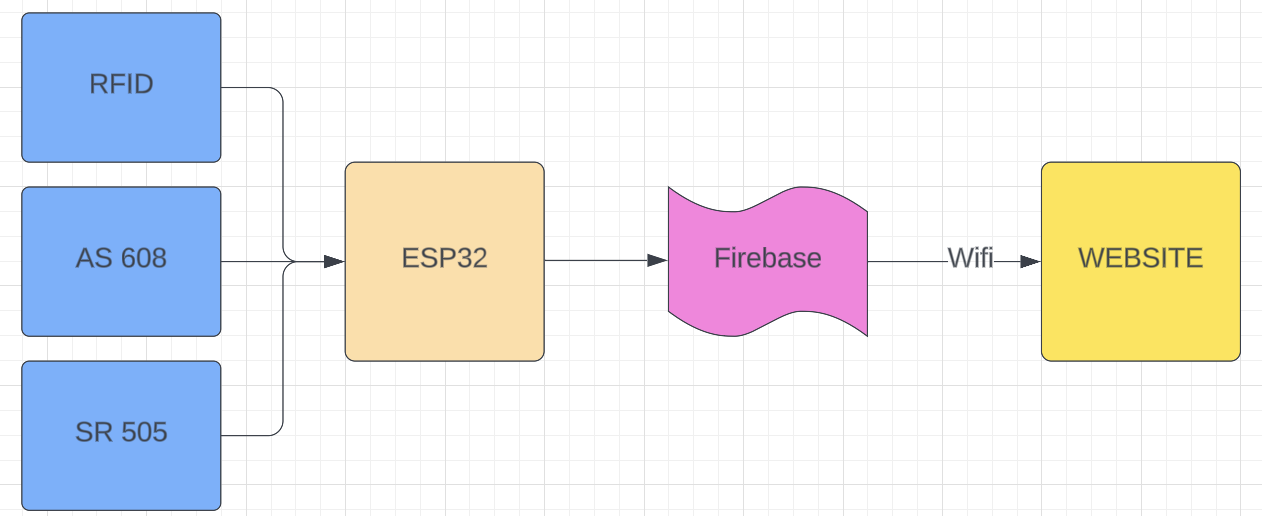


*Hình 3.1 Sơ đồ khối hệ thống*

**Chức năng từng khối:**

* Khối xử lý trung tâm: Là khối điều khiển chính của hệ thống, thu thập dữ liệu từ các thiết bị sau đó xử lý và điều khiển khối ngoại vi và khối hiển thị.
* Khối cảm biến: bao gồm nhiều cảm biến thu thập các thông số.
* Khối điều khiển thiết bị ngoại vi: bao gồm các thiết bị điều khiển như nút điều khiển các chế độ hoạt động trên website, cùng các thiết bị thực thi hành động là các Rơ-le dưới sự điều khiển của khối xử lý giao tiếp không dây.
* Khối hiển thị: các thông số đo được từ cảm biến cũng như trạng thái hoạt động của các thiết bị ngoại vi được hiển thị trực tiếp trên website.
* Khối nguồn: Cung cấp nguồn cho toàn bộ hoạt động của hệ thống bao gồm: khối điều khiển trung tâm, các cảm biến.

Hệ thống hoạt động dựa trên mô hình được trình bày như hình 3.2 dưới đây:



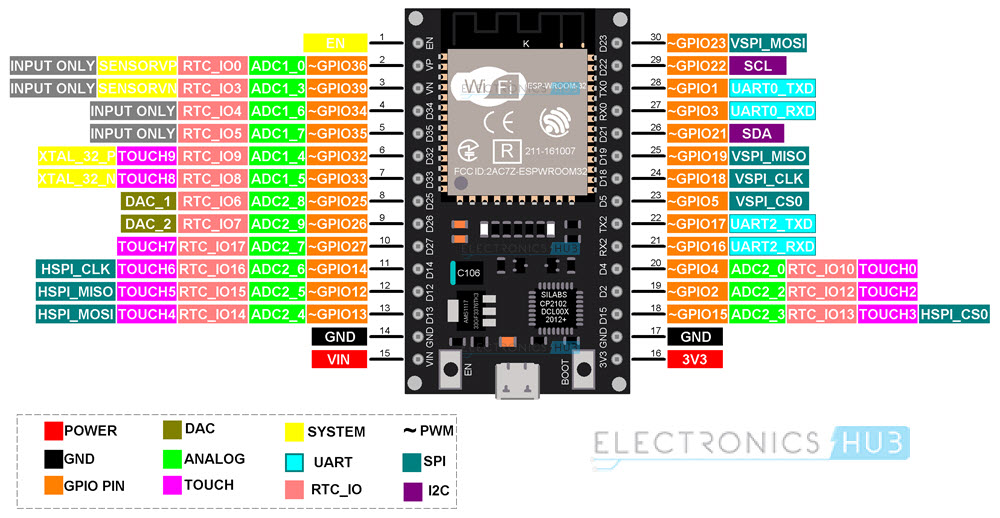
*Hình 3.2 Mô hình hệ thống*

* + 1. Tính toán và thuyết kế mạch:
       1. Khối xử lý trung tâm:
* Khối xử lý trung tâm trong hệ thống IoT cần đáp ứng các yêu cầu về khả năng kết nối, hiệu suất, tiết kiệm năng lượng, và bảo mật để vận hành hiệu quả. Trước hết, vi điều khiển cần hỗ trợ các kết nối không dây như Wi-Fi và Bluetooth nhằm trao đổi dữ liệu nhanh chóng, ổn định, đáp ứng khả năng mở rộng và kết nối với nhiều thiết bị ngoại vi. Đồng thời, khả năng tính toán mạnh mẽ, đa tác vụ và quản lý bộ nhớ tốt là cần thiết để xử lý dữ liệu từ các cảm biến một cách liên tục và chính xác. Đối với các hệ thống yêu cầu hoạt động dài hạn trên nguồn pin, khối xử lý trung tâm cần có chế độ tiết kiệm năng lượng hiệu quả để tối ưu hóa thời gian hoạt động. Ngoài ra, các tính năng bảo mật như mã hóa dữ liệu và xác thực an toàn là yếu tố bắt buộc, nhằm đảm bảo rằng hệ thống tránh được những rủi ro an ninh mạng.

*Thông số kỹ thuật của ESP32:*

* Bộ xử lý:
  + CPU: Bộ vi xử lý Xtensa lõi kép (hoặc lõi đơn) 32-bit LX6, hoạt động ở tần số 240 MHz (160 MHz cho ESP32-S0WD và ESP32-U4WDH) và hoạt động ở tối đa 600 MIPS (200 MIPS với ESP32-S0WD/ESP32-U4WDH)
  + Bộ đồng xử lý (co-processor) công suất cực thấp (Ultra low power, viết tắt: ULP)
* Hệ thống xung nhịp: CPU Clock, RTC Clock và Audio PLL Clock
* Bộ nhớ nội: 448 KB bộ nhớ ROM và 520 KB bộ nhớ SRAM
* Kết nối không dây: Wi-Fi: 802.11 b/g/n và Bluetooth: v4.2 BR/EDR và BLE
* 34 GPIO pad vật lý
* Bảo mật:
  + Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI.
  + Secure boot (tạm dịch: khởi động an toàn)
  + Mã hóa flash
  + 1024-bit OTP, lên đến 768-bit cho khách hàng
  + Tăng tốc mã hóa phần cứng: AES, SHA-2, RSA, elliptic curve cryptography
* Quản lý năng lượng:
  + Bộ ổn áp nội với điện áp rơi thấp (internal low-dropout regulator)
  + Miền nguồn riêng (individual power domain) cho RTC
  + Dòng 5 μA cho chế độ deep sleep
  + Trở lại hoạt động từ ngắt GPIO, timer, đo ADC, ngắt với cảm ứng điện dung

*Sơ đồ chân:*



*Hình 3.3 Sơ đồ chân*

* + - 1. Khối cảm biến:
* Khối cảm biến trong hệ thống IoT cần đáp ứng các yêu cầu về độ chính xác, khả năng tương thích, tiết kiệm năng lượng và hoạt động ổn định trong môi trường thực tế. Để đảm bảo hệ thống thu thập dữ liệu đáng tin cậy, các cảm biến phải có độ nhạy cao và cung cấp dữ liệu chính xác, hỗ trợ các giao thức giao tiếp như I2C, SPI, hoặc UART để dễ dàng kết nối với vi điều khiển ESP32. Tiêu thụ năng lượng thấp cũng là một yếu tố quan trọng nhằm duy trì thời gian hoạt động dài trong các ứng dụng dùng nguồn pin. Ngoài ra, các cảm biến phải hoạt động ổn định trong điều kiện môi trường khác nhau, bao gồm nhiệt độ và độ ẩm biến đổi, nhằm duy trì độ bền và tính nhất quán của hệ thống. Khả năng cập nhật dữ liệu theo thời gian thực giúp hệ thống phản ứng nhanh chóng khi có thay đổi, đặc biệt quan trọng trong các ứng dụng yêu cầu tính tức thì.
* Cảm biến RFID:



*Hình 3.4 RFID RC522*

* + Lí do lựa chọn linh kiện:
* Cảm biến RFID RC522 là một lựa chọn phổ biến cho các dự án điện tử cần sử dụng công nghệ nhận dạng qua sóng vô tuyến, đặc biệt là trong các ứng dụng như kiểm soát truy cập, theo dõi tài sản và hệ thống tự động hóa. Lý do nên chọn RFID RC522 là vì nó có giá thành hợp lý, dễ tìm kiếm trên thị trường, phù hợp cho cả các dự án học tập và ứng dụng thực tế. RC522 hoạt động ở tần số 13.56 MHz, tương thích với nhiều loại thẻ và chip RFID hiện nay, giúp linh hoạt trong việc triển khai. Thiết bị này cũng có kích thước nhỏ gọn, tiêu thụ điện năng thấp và có khả năng kết nối dễ dàng với các vi điều khiển như Arduino hay ESP32 thông qua giao tiếp SPI, giảm thiểu công sức lập trình và tối ưu hóa thời gian phát triển. Ngoài ra, các thư viện mã nguồn mở có sẵn trên nhiều nền tảng cho phép người dùng dễ dàng tích hợp RC522 vào dự án, tạo ra hệ thống nhận dạng thông minh và hiệu quả.
  + Thông số kỹ thuật:
* Nguồn sử dụng: 3.3VDC
* Dòng điện:13~26mA
* Tần số hoạt động: 13.56Mhz
* Khoảng cách hoạt động: 0~60mm（mifare1 card）
* Chuẩn giao tiếp: SPI
* Tốc độ truyền dữ liệu: tối đa 10Mbit/s
* Các loại card RFID hỗ trợ: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire
* Kích thước: 40mm × 60mm
  + Cách nối chân:
* SDA🡪 D21
* SCK🡪 D18
* MOSI🡪 D23
* MISO🡪 D19
* IRQ
* GND🡪 GND của ESP32
* RST🡪 D22
* 3.3V🡪 3V3
  + Hình nối chân

A circuit board with wires connected to it

Description automatically generated

*Hình 3.5*

* Cảm biến AS 608:

A black device with colorful wires

Description automatically generated

*Hình 3.6 AS 608*

* + Lí do lựa chọn linh kiện:
* Cảm biến AS608 là một giải pháp lý tưởng cho các ứng dụng nhận diện vân tay nhờ vào những ưu điểm nổi bật của nó. Đầu tiên, AS608 sở hữu khả năng quét và nhận diện vân tay nhanh chóng và chính xác, giúp nâng cao hiệu suất cho các hệ thống kiểm soát truy cập. Thứ hai, cảm biến này có kích thước nhỏ gọn và dễ dàng tích hợp vào nhiều loại thiết bị, từ hệ thống bảo mật đến các sản phẩm tiêu dùng, làm cho việc triển khai trở nên thuận tiện. Bên cạnh đó, AS608 được trang bị công nghệ học máy giúp cải thiện khả năng nhận diện và giảm thiểu sai sót, ngay cả khi điều kiện ánh sáng không lý tưởng. Hơn nữa, cảm biến này hỗ trợ lưu trữ nhiều mẫu vân tay, cho phép quản lý nhiều người dùng cùng một lúc, điều này rất hữu ích trong các ứng dụng doanh nghiệp. Cuối cùng, với giá thành phải chăng và tính khả thi cao, cảm biến AS608 là lựa chọn hoàn hảo cho những ai cần một giải pháp nhận diện vân tay hiệu quả và tin cậy.
  + Thông số kỹ thuật:
* Điện áp sử dụng: 3.0~3.6VDC (**thường cấp 3.3VDC**, lưu ý quan trọng nếu cấp lớn hơn 3.3VDC cảm biến sẽ cháy ngay lập tức).
* Dòng tiêu thụ: 30~60mA, trung bình 40mA
* Communication Interface: USB /UART
* Tốc độ Baudrate UART: 9600 x N (N từ 1~12), mặc định N=6 baudrate = 57600,8,1.
* USB communication: 2.0 full speed
* Sensor image size (pixel): 256 x 288 pixels
* Image processing time (s): <0.4s
* Power-on delay (s): <0.1s (the module needs about 0.1S to initialize after power on)
* Job search time (s): <0.3s
* FRR (rejection rate) <1%
* FAR (recognition rate) <0.001%
* Fingerprint storage capacity 300 (ID: 0 ~ 299)
  + Cách nối chân
* RXD🡪 TX2
* TXD🡪 RX2
* GND🡪 GND của ESP32
* 3V3🡪 3V3
  + Hình nối chân

A circuit board with different colors and wires

Description automatically generated

*Hình 3.7*

* Cảm biến SR 505:



*Hình 3.8 SR 505*

* + Lí do lựa chọn linh kiện
* Cảm biến SR505 là lựa chọn tuyệt vời cho các ứng dụng phát hiện chuyển động nhờ vào nhiều ưu điểm nổi bật. Đầu tiên, cảm biến này có khả năng phát hiện chuyển động trong khoảng cách lên đến 5 mét với góc quét rộng, giúp theo dõi một khu vực lớn một cách hiệu quả. Thứ hai, SR505 hoạt động với điện áp thấp và tiêu thụ năng lượng rất ít, giúp tiết kiệm điện năng, làm cho nó trở thành lựa chọn lý tưởng cho các dự án yêu cầu nguồn điện hạn chế. Bên cạnh đó, cảm biến này dễ dàng tích hợp với các vi điều khiển như Arduino và ESP32, giúp lập trình viên dễ dàng triển khai trong các hệ thống tự động hóa và bảo mật. Hơn nữa, SR505 có thiết kế nhỏ gọn và bền bỉ, phù hợp với nhiều điều kiện môi trường khác nhau. Cuối cùng, với giá thành hợp lý và tính linh hoạt cao, cảm biến SR505 là giải pháp đáng tin cậy cho các ứng dụng như hệ thống chiếu sáng tự động, cảnh báo an ninh, và quản lý giao thông.
  + Thông số kỹ thuật
* Điện áp hoạt động: 4.5~20VDC
* Dòng tiêu thụ: <60uA
* Mức tín hiệu: High 3.3V / Low 0V
* Trigger: repeatable trigger
* Thời gian trễ T sau khi kích hoạt: 8s + -30%
* Góc quét: Max 100 độ (hình nón có tâm là cảm biến)
* Khoảng cách bắt: 3 meters
* Đường kính thấu kính: 10mm
* Kích thước: 10 x 23mm
  + Cách nối chân
* OUT🡪 D15
* GND🡪 GND của ESP32
* VCC🡪 3V3
  + Hình nối chân

A circuit board with wires

Description automatically generated

*Hình 3.9*

* + - 1. Khối điều khiển ngoại vi:
* Đèn 1
  + Lí do lựa chọn linh kiện
* LED (diode phát sáng) là một lựa chọn lý tưởng để mô phỏng trong các dự án điện tử nhờ vào nhiều ưu điểm vượt trội. Đầu tiên, LED có kích thước nhỏ gọn và nhẹ, dễ dàng tích hợp vào nhiều loại mạch và thiết kế, từ những dự án đơn giản đến phức tạp. Thứ hai, chúng có khả năng phát sáng ngay lập tức khi được cấp điện, cho phép người dùng quan sát kết quả một cách nhanh chóng và rõ ràng. Hơn nữa, LED rất tiết kiệm năng lượng, tiêu thụ ít điện năng so với các nguồn sáng truyền thống, giúp giảm chi phí hoạt động và bảo vệ môi trường. LED cũng có độ bền cao, khả năng chống va đập và ít bị ảnh hưởng bởi điều kiện môi trường, làm cho chúng trở thành giải pháp lý tưởng cho cả các ứng dụng trong nhà và ngoài trời.
  + Thông số kỹ thuật
* Loại led: led trong 7 màu
* Kích thước: 3mm / 5mm (tự chọn trong phân loại)
* Điện áp: 3.2 – 3.6V
* Dòng: 15 – 20 mA
* Chế độ nháy tự động: FADE > SMOOTH > FLASH
* Độ sáng: 6000 – 8000 MCD
* Trọng lượng: 2 ~ 5g (10 led )
  + Cách nối chân
* GND🡪 GND
* Chân còn lại🡪 D25
  + Hình nối chân

A computer chip with wires connected to it

Description automatically generated

*Hình 3.10 Cách nối chân đèn*

* Đèn 2
  + Cách nối chân
* GND🡪 GND
* Chân còn lại🡪 D2
  + Hình nối chân

A computer chip with a green wire

Description automatically generated

*Hình 3.11 Cách nối chân đèn*

* Loa buzzer

A black round object with a hole

Description automatically generated

*Hình 3.12 Loa buzzer*

* + Lí do lựa chọn linh kiện
* Buzzer là một thiết bị âm thanh rất phổ biến trong các dự án mô phỏng và điện tử, nhờ vào nhiều ưu điểm vượt trội. Đầu tiên, buzzer có kích thước nhỏ gọn và dễ dàng tích hợp vào nhiều loại mạch điện, giúp tiết kiệm không gian và phù hợp cho cả những dự án phức tạp lẫn đơn giản. Thứ hai, buzzer có khả năng phát ra âm thanh ngay lập tức khi được cấp nguồn, cho phép người dùng nhận biết thông tin một cách nhanh chóng và rõ ràng, rất hữu ích trong các ứng dụng báo động hoặc thông báo. Hơn nữa, buzzer tiêu thụ năng lượng thấp, giúp tiết kiệm điện cho các dự án sử dụng pin hoặc nguồn điện hạn chế. Buzzer cũng có độ bền cao và hoạt động ổn định trong nhiều điều kiện môi trường khác nhau, làm cho nó trở thành lựa chọn đáng tin cậy cho cả ứng dụng trong nhà và ngoài trời. Cuối cùng, với sự đa dạng về kiểu dáng và âm thanh, buzzer cho phép người dùng dễ dàng tạo ra các hiệu ứng âm thanh khác nhau, phục vụ cho nhiều mục đích mô phỏng và giải trí. Tất cả những lý do này khiến buzzer trở thành một công cụ lý tưởng trong việc mô phỏng và thử nghiệm trong lĩnh vực điện tử.
  + Thông số kỹ thuật
* Nguồn : 3.5V - 5.5V
* Dòng điện tiêu thụ: <25mA
* Tần số cộng hưởng: 2300Hz ± 500Hz
* Biên độ âm thanh: >80 dB
* Nhiệt độ hoạt động:-20 °C đến +70 °C
* Kích thước : Đường kính 12mm, cao 9,7mm
  + Cách nối chân
* Đầu âm🡪 GND
* Đầu dương🡪 D26
  + Hình nối chân

A circuit board with wires

Description automatically generated

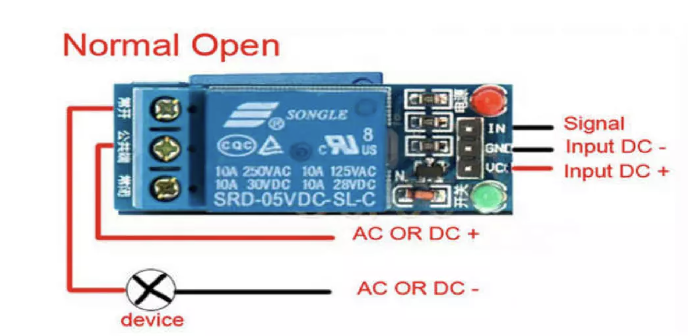
*Hình 3.13 Cách nối chân loa buzzer*

* Khoá điện



*Hình 3.14 Khoá điện solenoid*

* + Lí do lựa chọn linh kiện
* Khóa solenoid là một giải pháp lý tưởng cho các hệ thống kiểm soát truy cập và an ninh nhờ vào những ưu điểm nổi bật của nó. Đầu tiên, khóa solenoid hoạt động dựa trên nguyên lý điện từ, cho phép mở và đóng khóa một cách nhanh chóng và chính xác chỉ bằng cách cấp điện, giúp tiết kiệm thời gian và tăng cường sự tiện lợi cho người sử dụng. Thứ hai, thiết kế đơn giản và nhỏ gọn của khóa solenoid giúp dễ dàng tích hợp vào nhiều loại cửa và hệ thống khác nhau, từ cửa gỗ đến cửa kính. Hơn nữa, khóa solenoid có độ bền cao và khả năng chống lại sự cố cơ học, đảm bảo an toàn và bảo mật cho tài sản của người dùng. Bên cạnh đó, với tính năng kiểm soát từ xa, khóa solenoid có thể dễ dàng kết nối với các hệ thống tự động hóa và các thiết bị như thẻ RFID, vân tay hoặc smartphone, tạo ra sự linh hoạt trong việc quản lý và truy cập. Cuối cùng, giá thành phải chăng và khả năng hoạt động ổn định trong nhiều điều kiện khác nhau khiến khóa solenoid trở thành lựa chọn tuyệt vời cho các ứng dụng an ninh hiện đại. Tất cả những lý do này làm cho khóa solenoid trở thành một công cụ đáng tin cậy và hiệu quả trong việc bảo vệ tài sản và kiểm soát truy cập.
  + Thông số kỹ thuật
* Điện áp sử dụng: 12 / 24VDC (tùy chọn).
* Dòng điện tiêu thụ: 0.8A.
* Công suất tiêu thụ: 9.6W
* Sử dụng Solenoid từ.
* Tốc độ phản ứng: < 1s.
* Thời gian kích liên tục: < 10s
* Đi kèm gá chốt khóa.
  + Cách nối chân
* Đầu âm 🡪 Adapter5v
* Đầu dương🡪 Chia làm hai: một đầu nối với COM(relay), đầu còn lại nối với NO(relay)
* Relay



*Hình 3.15 Relay*

* + Lí do lựa chọn linh kiện
  + Thông số kỹ thuật
* Điện áp hoạt động: 5V
* Dòng kích Relay: 5mA
* Kích thước: 43mm x 17.3mm x 17mm (dài x rộng x cao)
* Trọng lượng: 15g
  + Cách nối chân
* DC+ 🡪 3.3V
* DC- 🡪 Nối vào GND
* IN 🡪 Chân 4
  + Hình nối chân

A circuit board with wires and a square

Description automatically generated with medium confidence

*Hình 3.16*

* + - 1. Khối nguồn:
* Nguồn adapter



*Hình 3.17 Nguồn adapter*

* + Lí do lựa chọn linh kiện
* Nguồn adapter 12V là lựa chọn phổ biến và lý tưởng cho nhiều ứng dụng điện tử, đặc biệt là trong các hệ thống điều khiển như khóa solenoid. Đầu tiên, adapter 12V cung cấp điện áp ổn định và đủ mạnh để đáp ứng yêu cầu hoạt động của nhiều thiết bị, giúp đảm bảo hiệu suất tối ưu và độ bền cho các linh kiện. Thứ hai, điện áp 12V là mức điện áp tiêu chuẩn được sử dụng rộng rãi trong ngành công nghiệp, khiến việc tìm kiếm và thay thế adapter trở nên dễ dàng và tiết kiệm chi phí. Bên cạnh đó, adapter 12V thường có khả năng cung cấp dòng điện cao (thường từ 1A đến 5A), điều này rất hữu ích cho các thiết bị tiêu thụ nhiều năng lượng như khóa solenoid. Hơn nữa, với tính năng bảo vệ quá tải và ngắn mạch, adapter 12V giúp bảo vệ an toàn cho các thiết bị kết nối, giảm thiểu nguy cơ hư hại do sự cố điện. Cuối cùng, sự phổ biến của adapter 12V cũng đồng nghĩa với việc có nhiều tùy chọn về chất lượng và giá cả, cho phép người dùng dễ dàng tìm kiếm sản phẩm phù hợp với nhu cầu và ngân sách của mình
  + Thông số kỹ thuật
* Model: OEM ADS0248-W120200A
* Điện áp ngõ vào: 100~240VAC, 50/60Hz.
* Điện áp ngõ ra: 12VDC (nếu sử dụng liên tục nên cung cấp ở mức 70% công suất).
* Dòng điện ngõ ra tối đa: 2A
* Công suất tối đa: 24W (nếu sử dụng liên tục thì tải tiêu thụ nên ở khoảng 70% công suất của nguồn để đạt độ bền cao nhất).
* Kiểu nguồn: Nguồn xung.
* Kiểu giắc ngõ ra: Chuẩn Jack DC tròn 5.5\*2.1~2.5mm
* Chiều dài dây dẫn: 1.5m
* Kích thước: 72 x 34 x 50mm
  + Cách nối chân
* Đầu âm🡪 kết nối với khoá điện
* Đầu dương🡪 kết nối với khoá điện
  + 1. Sơ đồ nguyên lý toàn mạch

A diagram of a circuit board

Description automatically generated

*Hình 3.18 Sơ đồ toàn mạch*

1. Thi công hệ thống:
   1. Giới thiệu:

* Sau quá trình tìm hiểu và nguyên cứu thì chúng ta sẽ bắt đầu lập trình về thi công hệ thống
  1. Lập trình hệ thống:
     1. Lưu đồ giải thuật:
     2. Phần mềm lập trình cho vi điều khiển:
* Phần mềm lập trình Arduino, hay còn gọi là Arduino IDE (Integrated Development Environment), là một công cụ mạnh mẽ và dễ sử dụng dành cho những ai muốn phát triển và lập trình các dự án điện tử dựa trên nền tảng Arduino. Với giao diện thân thiện, Arduino IDE cho phép người dùng dễ dàng viết mã, biên dịch và tải lên chương trình cho các board Arduino, như Arduino Uno, Mega, Nano và nhiều loại khác. Một trong những ưu điểm nổi bật của phần mềm này là nó hỗ trợ nhiều thư viện phong phú, giúp người dùng dễ dàng tích hợp các cảm biến, motor, và các thiết bị ngoại vi khác vào dự án của mình. Ngoài ra, Arduino IDE còn có tính năng kiểm tra lỗi và gợi ý cú pháp, giúp người dùng mới có thể học hỏi và cải thiện kỹ năng lập trình của mình một cách hiệu quả. Với khả năng tương thích trên nhiều hệ điều hành như Windows, macOS và Linux, Arduino IDE đã trở thành lựa chọn hàng đầu cho những ai đam mê công nghệ, từ những người mới bắt đầu đến các chuyên gia trong lĩnh vực lập trình và điện tử.
  + 1. Lập trình web, kết nối firebase:

# Tổng quan

* Xây dựng một dashboard đơn giản cho dự án Smart Lock với giao diện web đơn giản để hiển thị và điều khiển các trạng thái của hệ thống.
* Kết nối với firebase với mục đích hiển thị các trạng thái theo thời gian thực như trạng thái mở/khóa cửa, mở đèn, …
* Các yêu cầu của dashboard:
  + Hiển thị trạng thái khóa cửa, trạng thái đèn điện trong nhà, cảnh báo
  + Trạng thái cảm biến vân tay, thẻ từ, cảm biến chuyển động.
  + Cập nhật trạng thái real-time từ Firebase
  + Sử dụng biểu đồ để trực quan hóa dữ liệu trạng thái cửa theo thời gian
* Các bước cần chuẩn bị
  + Firebase Realtime Database để lưu trữ và cập nhật trạng thái thiết bị và cảm biến
  + HTML, CSS, Javascript cho giao diện người dùng

# Các bước cụ thể:

## HTML, CSS, JavaScript

### Cấu trúc cơ bản cho HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8"> <!-- định dạng kí tự -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- hiển thị toàn phù hợp với màn hình máy tính, điện thoại, máy tính bảng, ... -->

    <title>Smart Lock</title>

    <!-- Gowun Batang Font -->

    <link href="https://fonts.googleapis.com/css2?family=Gowun+Batang:wght@400;700&display=swap" rel="stylesheet">

    <!-- Material Icons -->

    <link href="https://fonts.googleapis.com/icon?family=Material+Icons+Outlined" rel="stylesheet">

    <!-- đường link thêm + Outlined -->

    <!-- Custom CSS -->

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <!-- Custom JS -->

    <script src="js/scripts.js"></script>

</body>

</html>

### Bố cục trang sử dụng CSS Grid

Mô-đun The CSS Grid Layout cung cấp hệ thống bố cục dựa trên lưới, với các hàng và cột, giúp thiết kế trang web dễ dàng hơn mà không cần phải sử dụng float và positioning.

Trước hết tạo một thẻ div với class=“grid-container” chứa 3 thẻ:

* <header class="header"></header> : thành phần đầu trang
* <aside id="sidebar"></aside> : thanh bên
* <main class="main-container"></main> : phần nội dung chính

HTML Code:

<div class="grid-container">

    <!-- Header -->

    <header class="header"></header>

    <!-- End Header -->

    <!-- Sidebar -->

    <aside id="sidebar"></aside>

    <!-- End Sidebar -->

    <!-- Main -->

    <main class="main-container"></main>

    <!-- End Main -->

</div>

CSS Code:

/\* ------------------- BODY ------------------- \*/

body {

    margin: 0; /\* Loại bỏ khoảng cách mặc định của trang \*/

    padding: 0; /\* Loại bỏ đệm mặc định của trang \*/

    background-color: #1d2634; /\* Đặt màu nền tối cho toàn bộ trang \*/

}

/\* ------------------- GRID CONTAINER ------------------- \*/

.grid-container {

    display: grid; /\* Sử dụng CSS Grid để tạo bố cục lưới \*/

    grid-template-columns: 260px 1fr 1fr 1fr; /\* Tạo 4 cột với sidebar cố định và ba cột còn lại bằng nhau \*/

    grid-template-rows: 0.2fr 3fr; /\* Tạo 2 hàng: header chiếm 0.2 phần và main chiếm 3 phần \*/

    /\* Xác định khu vực lưới cho từng thành phần \*/

    grid-template-areas:

        "sidebar header header header" /\* Hàng đầu tiên với sidebar và header \*/

        "sidebar main main main"; /\* Hàng thứ hai với sidebar và main \*/

    height: 100vh; /\* Chiều cao của lưới bằng 100% chiều cao màn hình \*/

}

/\* ------------------- HEADER ------------------- \*/

.header {

    grid-area: header; /\* Xác định khu vực lưới là header \*/

    height: 50px; /\* Chiều cao của header là 50px \*/

    display: flex; /\* Sử dụng Flexbox để căn chỉnh nội dung bên trong \*/

    align-items: center; /\* Căn giữa nội dung theo chiều dọc \*/

    justify-content: space-between; /\* Phân bố nội dung đều theo chiều ngang \*/

    background-color: #9900db; /\* Đặt màu nền cho header \*/

    padding: 0 30px; /\* Thêm đệm trái và phải là 30px \*/

    box-shadow: 0 14px 7px -3px #000000d9; /\* Thêm bóng đổ cho header \*/

}

/\* ------------------- SIDEBAR ------------------- \*/

#sidebar {

    grid-area: sidebar; /\* Xác định khu vực lưới là sidebar \*/

    height: 100%; /\* Sidebar chiếm 100% chiều cao của container \*/

    background-color: #55007ad9; /\* Đặt màu nền cho sidebar \*/

    overflow-y: auto; /\* Thêm thanh cuộn dọc nếu nội dung vượt quá chiều cao \*/

    transition: all 0.5s; /\* Hiệu ứng chuyển tiếp cho tất cả các thuộc tính trong 0.5 giây \*/

    -webkit-transition: all 0.5s; /\* Hỗ trợ hiệu ứng chuyển tiếp cho trình duyệt Webkit \*/

}

/\* ------------------- MAIN CONTAINER ------------------- \*/

.main-container {

    grid-area: main; /\* Xác định khu vực lưới là main \*/

    background-color: #9900db99; /\* Đặt màu nền với độ trong suốt \*/

    overflow-y: auto; /\* Thêm thanh cuộn dọc nếu nội dung vượt quá chiều cao \*/

    padding: 20px; /\* Thêm đệm trong để tạo khoảng cách cho nội dung \*/

}

Trong phần ***header* (đầu trang)** có ba phần:

* Menu-icon: tạo biểu tượng ***menu*** và sử dụng JavaScript để thực hiện hành động mở thanh sidebar khi biểu tượng được nhấp vào

HTML Code:

<div class="menu-icon" onclick="openSidebar()">

    <span class="material-icons-outlined">menu</span>

</div>

CSS Code:

.menu-icon {

    display: none; /\* Ẩn biểu tượng menu (không hiển thị trên trang) \*/

}

.sidebar-responsive {

    display: inline !important; /\* Hiển thị dưới dạng inline, ưu tiên hiển thị này \*/

    position: absolute; /\* Vị trí tuyệt đối cho phần sidebar-responsive \*/

    z-index: 12 !important; /\* Đặt độ ưu tiên hiển thị cho phần sidebar-responsive \*/

}

JavaScript Code:

// SIDEBAR TOGGLE

let sidebarOpen = false; /\* Khởi tạo biến báo trạng thái mở/đóng sidebar; mặc định là đóng \*/

let sidebar = document.getElementById("sidebar"); /\* Lấy phần tử sidebar từ DOM bằng ID "sidebar" \*/

function openSidebar() {

    if (!sidebarOpen) { /\* Kiểm tra nếu sidebar hiện đang đóng \*/

        sidebar.classList.add("sidebar-responsive"); /\* Thêm lớp "sidebar-responsive" để mở sidebar \*/

        sidebarOpen = true; /\* Cập nhật trạng thái sidebar là mở \*/

    }

}

function closeSidebar() {

    if (sidebarOpen) { /\* Kiểm tra nếu sidebar hiện đang mở \*/

        sidebar.classList.remove("sidebar-responsive"); /\* Xóa lớp "sidebar-responsive" để đóng sidebar \*/

        sidebarOpen = false; /\* Cập nhật trạng thái sidebar là đóng \*/

    }

}

* Header-left và header-right: tạo ra một cấu trúc cho phần đầu trang (header) với hai bên: bên trái và bên phải với bên trái đặt biểu tượng search và bên phải là các biểu tượng thông báo, email, tài khoản người dùng.

HTML Code:

<div class="header-left">

    <span class="material-icons-outlined">search</span>

</div>

<div class="header-right">

    <span class="material-icons-outlined">notifications</span>

    <span class="material-icons-outlined">email</span>

    <span class="material-icons-outlined">account\_circle</span>

</div>

CSS Code:

.header-right {

    text-justify: end; /\* Căn chỉnh văn bản về cuối dòng \*/

}

Trong phần ***sidebar*** có hai phần: tiêu đề thanh sidebar và danh sách các mục trong sidebar. Đối với tiêu đề sẽ chứa tên ứng dụng và biểu tượng, còn phần danh sách các mục điều hướng đến các mục của trang.

HTML Code:

<aside id="sidebar">

    <!-- Tiêu đề của sidebar -->

    <div class="sidebar-title">

        <div class="sidebar-brand">

            <!-- Liên kết đến trang dashboard -->

            <a href="dashboard.html">

                <span class="material-icons-outlined">lock</span> <!-- Biểu tượng khóa -->

                <p>SMART LOCK</p> <!-- Tên ứng dụng -->

            </a>

        </div>

        <!-- Biểu tượng đóng sidebar, gọi hàm closeSidebar khi nhấp vào -->

        <span class="material-icons-outlined" onclick="closeSidebar()">close</span>

    </div>

    <!-- Danh sách các mục trong sidebar -->

    <ul class="sidebar-list">

        <li class="sidebar-list-item">

            <!-- Liên kết đến trang dashboard -->

            <a href="dashboard.html">

                <span class="material-icons-outlined">dashboard</span> Dashboard <!-- Biểu tượng và tên mục -->

            </a>

        </li>

        <li class="sidebar-list-item">

            <!-- Liên kết đến trang thông tin -->

            <a href="information.html"> <!-- Tạo thêm file information.html -->

                <span class="material-icons-outlined">info</span> Informations <!-- Biểu tượng và tên mục -->

            </a>

        </li>

    </ul>

</aside>

CSS Code:

.sidebar-title {

    display: flex; /\* Sử dụng flexbox để sắp xếp các phần tử con \*/

    justify-content: space-between; /\* Căn giữa các phần tử, tạo khoảng cách giữa chúng \*/

    align-items: center; /\* Căn giữa theo chiều dọc \*/

    padding: 20px; /\* Khoảng cách bên trong cho sidebar-title \*/

    margin-bottom: 30px; /\* Khoảng cách bên dưới cho sidebar-title \*/

}

.sidebar-title > span {

    display: none; /\* Ẩn các phần tử <span> trực tiếp bên trong sidebar-title \*/

} /\* Dấu > ở trên để tác dụng các span nằm ngay phía trong còn trong nữa thì không tác dụng. \*/

.sidebar-brand {

    margin-top: 15px; /\* Khoảng cách bên trên cho sidebar-brand \*/

    font-size: 25px; /\* Kích thước chữ cho tên thương hiệu \*/

    font-weight: 800; /\* Độ đậm của chữ \*/

    color: black; /\* Màu chữ cho tên thương hiệu \*/

}

.sidebar-brand .material-icons-outlined {

    font-size: 60px; /\* Kích thước biểu tượng trong sidebar-brand \*/

    display: flex; /\* Sử dụng flexbox để căn giữa biểu tượng \*/

    justify-content: center; /\* Căn giữa biểu tượng theo chiều ngang \*/

}

a {

    text-decoration: none; /\* Xóa gạch chân của liên kết \*/

    color: #000000; /\* Màu chữ cho liên kết \*/

}

a:hover {

    color: #ffffffb1; /\* Màu chữ khi di chuột qua liên kết \*/

}

a:active {

    opacity: 0.5; /\* Độ mờ cho liên kết khi được nhấp \*/

}

.sidebar-list {

    padding: 0; /\* Xóa khoảng cách bên trong cho danh sách sidebar \*/

    margin-top: 15px; /\* Khoảng cách bên trên cho danh sách sidebar \*/

    list-style-type: none; /\* Làm mất dấu đầu dòng của danh sách \*/

}

.sidebar-list-item {

    padding: 20px; /\* Khoảng cách bên trong cho từng mục danh sách \*/

    font-size: 20px; /\* Kích thước chữ cho từng mục danh sách \*/

}

.sidebar-list-item:hover {

    background-color: #55007ad9; /\* Màu nền khi di chuột qua từng mục danh sách \*/

    border-radius: 40px; /\* Bo tròn góc cho nền mục danh sách \*/

    font-weight: 550; /\* Độ đậm của chữ khi di chuột qua \*/

    color: #ffffffb1; /\* Màu chữ khi di chuột qua \*/

    cursor: pointer; /\* Hiển thị con trỏ chuột như một con trỏ nhấp \*/

}

.sidebar-list-item:active {

    opacity: 0.5; /\* Độ mờ cho mục danh sách khi được nhấp \*/

}

.sidebar-responsive {

    display: inline !important; /\* Hiển thị dưới dạng inline, ưu tiên hiển thị này \*/

    position: absolute; /\* Vị trí tuyệt đối cho phần sidebar-responsive \*/

    z-index: 12 !important; /\* Đặt độ ưu tiên hiển thị cho phần sidebar-responsive \*/

}

Trong phần ***main*** có các thẻ như sau:

* <div class="date-time-container">…</div> hiển thị thời gian thực

HTML Code:

<div class="date-time-container">

    <div id="real-time">

    </div>

    <h2>SMART LOCK</h2>

    <div id="date"></div>

    <script src="js/date\_time.js"></script>

</div>

CSS Code:

.date-time-container {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 0;

    margin: 0;

    background-color: #ffffff52;

    border-radius: 15px;

    cursor: pointer;

}

JavaScript Code:

function updateTime() {

    // Declare values

    const clockElement = document.getElementById("real-time");

    const dateElement = document.getElementById("date");

    const now = new Date();

    // Take the present time

    const hours = now.getHours().toString().padStart(2, '0');

    const minutes = now.getMinutes().toString().padStart(2, '0');

    const seconds = now.getSeconds().toString().padStart(2, '0');

    const timeString = `${hours}:${minutes}:${seconds}`;

    // Take the present date

    const day = now.getDate().toString().padStart(2, '0');

    const month = (now.getMonth() + 1).toString().padStart(2, '0');

    const year = now.getFullYear();

    const dateString = `${day}/${month}/${year}`;

    // Display time and date

    clockElement.textContent = timeString;

    dateElement.textContent = dateString;

}

// Update date and time per second

setInterval(updateTime, 1000);

// call Function

updateTime();

* <div class="main-title"></div> Phần tiêu đề chính tên dashboard

HTML Code:

<div class="main-title">

    <h2>DASHBOARD</h2>

</div>

CSS Code:

.main-title {

    display: flex;

    justify-content: space-between;

}

* <div class="main-cards">…</div> Chứa các thẻ devices và sensors ở bên trong
* <div class="devices-container" style="display: none;"></div> Chế độ hiển thị các thiết bị và trạng thái ở bên trong devices card với code như sau:

HTML Code:

<div class="devices-card" onclick="status(this, 'lock')">

    <h2 class="devices-title">LOCK DEVICE</h2>

    <span class="material-icons-outlined">lock</span>

    <h1 class="status" id="lock">OFF</h1>

</div>

CSS Code:

.devices-container {

    display: grid;

    grid-template-columns: 1fr 1fr 1fr;

    gap: 20px;

    margin-top: 60px;

    cursor: pointer;

    transition: background-color 2s;

}

.devices-card {

    background-color: #1d2634;

    color: #f5f5f5;

    margin-bottom: 20px;

    padding: 25px;

    box-sizing: border-box;

    -webkit-column-break-inside: avoid;

    border-radius: 5px;

    box-shadow: 0 6px 7px -4px #00000033;

}

.devices-card span {

    display: flex;

    justify-content: space-around;

}

.devices-card .material-icons-outlined {

    font-size: 50px;

}

.devices-card h1 {

    text-align: center;

}

.devices-title {

    display: flex;

    align-items: center;

    justify-content: center;

}

JavaScript Code:

// Display Device Status

let islocked = false;

// Display devices

function displayDevice() {

    // Get container of devices-card

    const deviceContainer = document.querySelector('.devices-container');

    // Make sure that the container is display or hidden

    const displayStyle = window.getComputedStyle(deviceContainer).display;

    if (displayStyle === "none") {

        deviceContainer.style.display = "grid"; // Display container

    } else {

        deviceContainer.style.display = "none"; // Hidden container

    }

}

* <div class="sensors-container" style="display: none;"></div> Chế độ hiển thị các sensors và trạng thái của chúng với code tương tự phần devices bên trên
* <div class="charts"></div> Phần này hiển thị biểu đồ trạng thái khóa theo thời gian thực

Lưu ý phải liên kết với phần chart như sau thì mới hiển thị biểu đồ:

<!-- ApexCharts -->

<script src="https://cdn.jsdelivr.net/npm/apexcharts"></script>

## Liên kết với Firebase

### Các bước liên kết với Firebase

**Bước 1: Tạo dự án Firebase**

* Truy cập Firebase Console và đăng nhập bằng tài khoản Google
* Chọn “Add Project” (Thêm dự án) và đặt tên cho dự án

**Bước 2: Thêm ứng dụng web vào Firebase**

* Sau khi dự án được tạo, chọn biểu tượng Web (</>) để thêm ứng dụng web
* Đặt tên cho ứng dụng của bạn và nhấn **Register App**.
* Firebase sẽ cung cấp một đoạn mã firebaseConfig, chứa thông tin cấu hình cần thiết để kết nối
* Lưu ý khi bấm vào Realtime Database: Vào phần rules và chỉnh thành true hết như hình dưới đây:

A computer screen with colorful text

Description automatically generated

**Bước 3: Thêm SDK Firebase vào Dự Án Web**

* Thêm các thư viện Firebase vào file HTML: đặt vào <body> tag:

<!-- Firebase -->

<script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js"></script>

<script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-database.js"></script>

**Bước 4: Khởi Tạo Firebase với Đối Tượng firebaseConfig**

Sao chép cấu hình firebaseConfig và dán vào file JavaScript của bạn:

// ------------------- FIREBASE -------------------

// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration

// For Firebase JS SDK v7.20.0 and later, measurementId is optional

const firebaseConfig = {

    apiKey: "AIzaSyDtbyt7uGp5k-62BKZ\_efJ5zhYMoCweZSY",

    authDomain: "smart-lock-project-9b50b.firebaseapp.com",

    databaseURL: "https://smart-lock-project-9b50b-default-rtdb.firebaseio.com",

    projectId: "smart-lock-project-9b50b",

    storageBucket: "smart-lock-project-9b50b.appspot.com",

    messagingSenderId: "830294056108",

    appId: "1:830294056108:web:edf2cb0dd7931488b9d309",

    measurementId: "G-V89KMGWYG5"

};

// Initialize Firebase

firebase.initializeApp(firebaseConfig);

const database = firebase.database();

Như vậy là đã liên kết với Firebase Realtime Database thành công.

Lưu ý trong phần này bạn thay đổi để liên kết với Firebase của các bạn còn code trên là ghi về database của mình.

// Cấu hình Firebase

const firebaseConfig = {

    apiKey: "API\_KEY",

    authDomain: "PROJECT\_ID.firebaseapp.com",

    databaseURL: "https://PROJECT\_ID.firebaseio.com",

    projectId: "PROJECT\_ID",

    storageBucket: "PROJECT\_ID.appspot.com",

    messagingSenderId: "SENDER\_ID",

    appId: "APP\_ID"

};

// Khởi tạo Firebase

firebase.initializeApp(firebaseConfig);

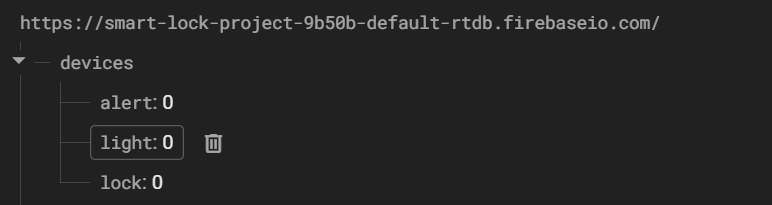
// Khởi tạo Realtime Database

const database = firebase.database();

### Đọc và ghi dữ liệu trong Realtime Database.

Sau khi đã liên kết thành công với Firebase, ta tiến hành đọc và ghi dữ liệu trong Realtime Database với các bước như sau:

Trước hết khởi tạo data như hình dưới đây:



A screenshot of a phone

Description automatically generated

### Ghi dữ liệu tới Database

Tiến hành ghi dữ liệu vào file JavaScript như sau:

Các thiết bị (devices):

// Lấy các phần tử DOM đại diện cho các thiết bị và cảm biến từ giao diện người dùng

const lock = document.getElementById('lock');

const light = document.getElementById('light');

const alert = document.getElementById('alert');

// Biến chứa trạng thái hiện tại của các thiết bị

let lockVal = lock.innerText; // Trạng thái của khóa

lock.onclick = function() { // Xử lý sự kiện khi người dùng nhấp vào "lock"

    lockVal = lockVal === 'OFF' ? 'ON' : 'OFF'; // Đổi trạng thái khóa giữa ON và OFF

    lock.innerHTML = lockVal; // Cập nhật giao diện hiển thị

    // Ghi trạng thái mới của khóa vào Realtime Database

    database.ref("/devices").update({

        "lock" : lockVal === 'OFF' ? 0 : 1 // 0 là OFF, 1 là ON

    })

}

let lightVal = light.innerText; // Trạng thái của đèn

light.onclick = function() { // Xử lý sự kiện khi người dùng nhấp vào "light"

    lightVal = lightVal === 'OFF' ? 'ON' : 'OFF'; // Đổi trạng thái đèn giữa ON và OFF

    light.innerHTML = lightVal; // Cập nhật giao diện hiển thị

    // Ghi trạng thái mới của đèn vào Realtime Database

    database.ref("/devices").update({

        "light" : lightVal === 'OFF' ? 0 : 1 // 0 là OFF, 1 là ON

    })

}

let alertVal = alert.innerText; // Trạng thái của cảnh báo

alert.onclick = function() { // Xử lý sự kiện khi người dùng nhấp vào "alert"

    alertVal = alertVal === 'OFF' ? 'ON' : 'OFF'; // Đổi trạng thái cảnh báo giữa ON và OFF

    alert.innerText = alertVal; // Cập nhật giao diện hiển thị

    // Ghi trạng thái mới của cảnh báo vào Realtime Database

    database.ref("/devices").update({

        "alert" : alertVal === 'OFF' ? 0 : 1 // 0 là OFF, 1 là ON

    })

}

Các cảm biến (sensors):

// Lấy các phần tử DOM đại diện cho các cảm biến từ giao diện người dùng

const sensor1 = document.getElementById('sensor1');

const sensor2 = document.getElementById('sensor2');

const sensor3 = document.getElementById('sensor3');

// Biến chứa trạng thái hiện tại của các cảm biến

let sensor1Val = sensor1.innerText; // Trạng thái của cảm biến dấu vân tay

sensor1.onclick = function() { // Xử lý sự kiện khi người dùng nhấp vào "sensor1"

    sensor1Val = sensor1Val === 'OFF' ? 'ON' : 'OFF'; // Đổi trạng thái cảm biến giữa ON và OFF

    sensor1.innerText = sensor1Val; // Cập nhật giao diện hiển thị

    // Ghi trạng thái mới của cảm biến vào Realtime Database

    database.ref("/sensors").update({

        "fingerprint" : sensor1Val === 'OFF' ? 0 : 1 // 0 là OFF, 1 là ON

    })

}

let sensor2Val = sensor2.innerText; // Trạng thái của cảm biến thẻ từ

sensor2.onclick = function() { // Xử lý sự kiện khi người dùng nhấp vào "sensor2"

    sensor2Val = sensor2Val === 'OFF' ? 'ON' : 'OFF'; // Đổi trạng thái cảm biến giữa ON và OFF

    sensor2.innerText = sensor2Val; // Cập nhật giao diện hiển thị

    // Ghi trạng thái mới của cảm biến vào Realtime Database

## database.ref("/sensors").update({

        "magnetic card" : sensor2Val === 'OFF' ? 0 : 1 // 0 là OFF, 1 là ON

    })

}

let sensor3Val = sensor3.innerText; // Trạng thái của cảm biến chuyển động

sensor3.onclick = function() { // Xử lý sự kiện khi người dùng nhấp vào "sensor3"

    sensor3Val = sensor3Val === 'OFF' ? 'ON' : 'OFF'; // Đổi trạng thái cảm biến giữa ON và OFF

    sensor3.innerText = sensor3Val; // Cập nhật giao diện hiển thị

    // Ghi trạng thái mới của cảm biến vào Realtime Database

    database.ref("/sensors").update({

        "movement" : sensor3Val === 'OFF' ? 0 : 1 // 0 là OFF, 1 là ON

    })

}

**Biến trạng thái** cho các thiết bị và cảm biến (như lockVal, lightVal, ...) dùng để lưu trữ và điều khiển trạng thái ON/OFF

**Sự kiện click** gán cho từng phần tử DOM để thực hiện chuyển đổi trạng thái và cập nhật lên giao diện người dùng và Firebase

**Ghi vào Firebase Realtime Database** được thực hiện bằng cách gọi

***database.ref("/devices").update()*** hoặc ***database.ref("/sensors").update()*** để đồng bộ trạng thái thiết bị/cảm biến với cơ sở dữ liệu Firebase.

### Đọc dữ liệu từ Firebase

Đọc các dữ liệu thời gian thực từ Firebase bằng cách lắng nghe sự kiện ***“value”*** trên các nút ***/devices*** và ***/sensors***

Cập nhật giao diện người dùng: Dữ liệu từ Firebase được kiểm tra để xác định trạng thái **ON/OFF** cho từng thiết bị và cảm biến, sau đó cập nhật trực tiếp vào giao diện (***innerText***) để đồng bộ với trạng thái trên Firebase.

JavaScript Code:

// Đọc dữ liệu từ Firebase Realtime Database cho các cảm biến

database.ref("/sensors").on("value", function(snapshot) {

    let sensors = snapshot.val(); // Lấy giá trị hiện tại của các cảm biến từ Firebase

    // Cập nhật trạng thái của từng cảm biến dựa trên dữ liệu từ Firebase

    sensors["fingerprint"] === 1 ? sensor1.innerText = 'ON' : sensor1.innerText = 'OFF'; // Kiểm tra và gán trạng thái của cảm biến dấu vân tay

    sensors["magnetic card"] === 1 ? sensor2.innerText = 'ON' : sensor2.innerText = 'OFF'; // Kiểm tra và gán trạng thái của cảm biến thẻ từ

    sensors["movement"] === 1 ? sensor3.innerText = 'ON' : sensor3.innerText = 'OFF'; // Kiểm tra và gán trạng thái của cảm biến chuyển động

})

# Kết luận:

Như vậy, trong phần này nhóm mình đã trình bày xong sơ lược cách mà dashboard được xây dựng đối với dự án Smart Lock đơn giản, từ bố cục bên ngoài đến các chức năng bên trong của dashboard này, cùng với cách liên kết Firebase, đọc và ghi dữ liệu lên Firebase và cập nhật vào giao diện để đồng bộ với trạng thái trên Firebase.

Source Code nhóm mình sẽ để ở trên Github:

<https://github.com/HTKFOOLISH/TT_IoT_Architecture_and_Protocol/tree/master/week5_6_7_8/dashboard>

Các bạn có thể xem source code phần dashboard để tham khảo thêm.

1. Kết quả - Nhận xét
   1. Kết quả
   2. Nhận xét

* Dự án SmartLock của tôi là một giải pháp tích hợp hoàn chỉnh, sử dụng ESP32 để điều khiển các thành phần như RFID-RC522, cảm biến vân tay AS608, và hệ thống cảm biến chuyển động SR505 nhằm đảm bảo an ninh và tiện ích cho người dùng. Với cơ chế mở khóa bằng thẻ RFID và vân tay, hệ thống của tôi cung cấp nhiều lớp bảo mật khác nhau, giúp nâng cao độ tin cậy và bảo vệ quyền riêng tư. Bằng cách kết hợp Firebase để lưu trữ và truy xuất dữ liệu, tôi đã tạo ra một hệ thống có khả năng kết nối từ xa, giúp dễ dàng quản lý và theo dõi hoạt động ra vào cửa. Bên cạnh đó, việc tích hợp relay điều khiển khóa điện và đèn cùng với cảnh báo bằng còi khi phát hiện thẻ không hợp lệ cho thấy mức độ tự động hóa và tiện lợi cao trong thiết kế. Nhìn chung, SmartLock không chỉ đảm bảo an toàn mà còn đáp ứng nhu cầu hiện đại của người dùng về một hệ thống nhà thông minh và tiện ích