Dashboard Smart Lock System

# 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

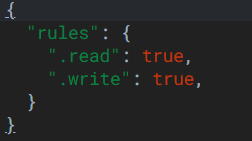
### Các bước liên kết với Firebase

**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:



**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);

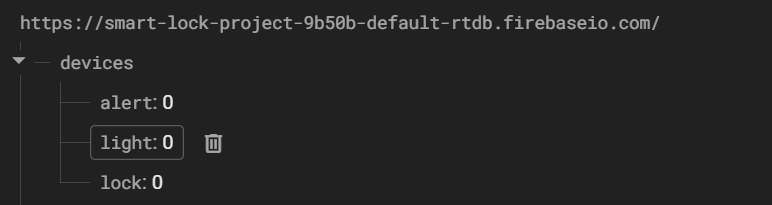
// Khởi tạo Realtime Database

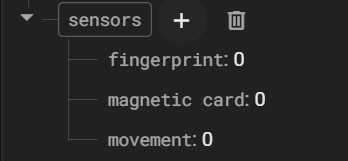
const database = firebase.database();

### 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:





### 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.