**Báo cáo chi tiết tích hợp Firebase trong hệ thống quản lý công việc dựa trên RFID**

**Giới thiệu**

Báo cáo này trình bày chi tiết việc tích hợp Google Firebase vào một hệ thống quản lý công việc sử dụng thẻ RFID, bàn phím 4x4 và màn hình TFT ILI9341, được phát triển trên nền tảng ESP32. Hệ thống cho phép công nhân đăng nhập bằng thẻ RFID, chọn và thực hiện các công việc (lắp ráp, sửa chữa, đóng gói), ghi nhận kết quả công việc (hoàn thành hoặc lỗi), và lưu trữ dữ liệu vào cơ sở dữ liệu thời gian thực của Firebase. Báo cáo bao gồm mục tiêu, cách thức tích hợp Firebase, cấu trúc dữ liệu, cơ chế hoạt động, các bước triển khai, ưu điểm, hạn chế và đề xuất cải tiến.

**Mục tiêu**

Hệ thống được thiết kế nhằm đạt được các mục tiêu sau:

* Xác thực công nhân thông qua thẻ RFID và truy xuất thông tin từ Firebase.
* Theo dõi và ghi nhận kết quả công việc (hoàn thành hoặc lỗi) theo thời gian thực.
* Cung cấp giao diện người dùng thân thiện trên màn hình TFT và bàn phím.
* Đồng bộ dữ liệu công việc với Firebase để hỗ trợ quản lý và phân tích từ xa.
* Đảm bảo tính chính xác của dữ liệu thời gian bằng cách sử dụng dịch vụ NTP.

**Tích hợp Firebase**

Firebase Realtime Database được sử dụng để lưu trữ và truy xuất thông tin công nhân, trạng thái làm việc và lịch sử công việc. Các thư viện HTTPClient và ArduinoJson được tích hợp để xử lý giao tiếp HTTPS với Firebase.

**Cấu hình Firebase**

* **URL cơ sở dữ liệu**: https://esp32-166c4-default-rtdb.firebaseio.com
* **Khóa API**: Được sử dụng để xác thực các yêu cầu HTTP, đảm bảo an toàn truy cập.
* **Giao thức**: Sử dụng HTTPS với các phương thức GET để truy xuất và POST để cập nhật dữ liệu.
* **Thư viện hỗ trợ**:
  + HTTPClient: Xử lý các yêu cầu HTTP đến Firebase.
  + ArduinoJson: Phân tích và tạo dữ liệu JSON cho các yêu cầu và phản hồi.

**Các hàm liên quan đến Firebase**

Các hàm chính được triển khai để tương tác với Firebase bao gồm:

* **fetchWorkerData(String uidStr)**:
  + **Chức năng**: Truy xuất thông tin công nhân từ Firebase dựa trên UID của thẻ RFID.
  + **Yêu cầu**: Gửi GET đến /workers/{uidStr}.json?auth={API\_KEY}.
  + **Kết quả**: Lấy tên công nhân và trạng thái công việc khả dụng (lapRap, suaChua, dongGoi).
  + **Xử lý lỗi**: Kiểm tra mã trạng thái HTTP và kết nối WiFi, hiển thị thông báo lỗi nếu thất bại.
* **sendWorkerStatus(String uidStr, String state)**:
  + **Chức năng**: Gửi trạng thái công nhân (check-in hoặc check-out) cùng thời gian hiện tại.
  + **Yêu cầu**: POST đến /workers/{uidStr}/status.json?auth={API\_KEY} với dữ liệu JSON chứa trạng thái và dấu thời gian.
  + **Xử lý lỗi**: Báo lỗi nếu mất kết nối WiFi hoặc yêu cầu thất bại.
* **sendTaskData(String uidStr, String taskName, int completed, int failed)**:
  + **Chức năng**: Ghi nhận số lượng công việc hoàn thành và lỗi cho từng loại công việc.
  + **Yêu cầu**: POST đến /history/{uidStr}.json?auth={API\_KEY} với dữ liệu JSON chứa thời gian, tên công việc, số lượng hoàn thành và lỗi.
  + **Xử lý lỗi**: Kiểm tra kết nối WiFi và mã HTTP.
* **sendCheckOutData(String uidStr)**:
  + **Chức năng**: Gửi tóm tắt công việc (số lượng hoàn thành và lỗi cho từng công việc) khi check-out.
  + **Yêu cầu**: POST đến /history/{uidStr}.json?auth={API\_KEY} với dữ liệu JSON tổng hợp.
  + **Xử lý lỗi**: Thông báo lỗi nếu yêu cầu không thành công.

**Cấu trúc dữ liệu trên Firebase**

Dữ liệu được tổ chức trong Firebase Realtime Database với cấu trúc sau:

* **/workers/{uidStr}**:
  + name: Tên công nhân (chuỗi, tối đa 50 ký tự).
  + tasks:
    - lapRap: {available: boolean}
    - suaChua: {available: boolean}
    - dongGoi: {available: boolean}
  + status:
    - state: "checkIn" hoặc "checkOut"
    - timestamp: Dấu thời gian (định dạng ISO 8601).
* **/history/{uidStr}**:
  + Lịch sử công việc:
    - timestamp: Thời gian thực hiện (ISO 8601).
    - task: Tên công việc (lapRap, suaChua, dongGoi).
    - completed: Số lượng công việc hoàn thành.
    - failed: Số lượng công việc lỗi.
  + Dữ liệu check-out:
    - checkOut:
      * lapRap: {completed: int, failed: int}
      * suaChua: {completed: int, failed: int}
      * dongGoi: {completed: int, failed: int}

**Các bước triển khai**

Việc tích hợp Firebase vào hệ thống được thực hiện qua các bước sau:

1. **Thiết lập dự án Firebase**:
   * Tạo dự án trên Firebase Console.
   * Kích hoạt Realtime Database và thiết lập quy tắc bảo mật để chỉ cho phép truy cập với khóa API.
   * Lấy URL cơ sở dữ liệu và khóa API từ cài đặt dự án.
2. **Cấu hình phần cứng**:
   * Kết nối ESP32 với các thành phần:
     + **MFRC522 (RFID)**: Sử dụng chân GPIO 5 (SDA), 22 (RST), và giao tiếp SPI (MOSI: 23, MISO: 19, SCK: 18).
     + **ILI9341 TFT**: Chân GPIO 15 (CS), 2 (DC), 4 (RST).
     + **Bàn phím 4x4**: Sử dụng các chân GPIO 13, 14, 12, 27 (hàng) và 26, 25, 33, 32 (cột).
   * Khởi tạo các thư viện: MFRC522, Adafruit\_ILI9341, Keypad, WiFi, HTTPClient, ArduinoJson, WiFiUDP, NTPClient.
3. **Kết nối WiFi**:
   * Sử dụng thông tin WiFi (WIFI\_SSID, WIFI\_PASSWORD) để kết nối ESP32 với mạng.
   * Kiểm tra trạng thái kết nối trong hàm setup() và thông báo qua Serial Monitor.
4. **Đồng bộ thời gian**:
   * Khởi tạo NTPClient với máy chủ pool.ntp.org và múi giờ UTC+7.
   * Hàm getTime() trả về dấu thời gian định dạng ISO 8601.
5. **Xây dựng giao diện người dùng**:
   * Sử dụng màn hình TFT để hiển thị các trạng thái (check-in, menu chính, chọn công việc, làm việc, check-out).
   * Hàm printWrapped xử lý hiển thị văn bản với tự động xuống dòng để tránh tràn màn hình.
   * Bàn phím 4x4 hỗ trợ nhập liệu với các phím số và ký tự đặc biệt (\* để quay lại, 1-3 để chọn tùy chọn).
6. **Triển khai xác thực RFID**:
   * Hàm readRFID đọc UID từ thẻ RFID với thời gian chờ 30 giây.
   * Hàm byteArrayToString chuyển đổi UID thành chuỗi HEX để gửi lên Firebase.
7. **Tích hợp Firebase**:
   * Triển khai các hàm fetchWorkerData, sendWorkerStatus, sendTaskData, và sendCheckOutData để giao tiếp với Firebase.
   * Xử lý dữ liệu JSON bằng ArduinoJson để truy xuất và gửi thông tin.
   * Kiểm tra kết nối WiFi trước mỗi yêu cầu HTTP.
8. **Quản lý trạng thái hệ thống**:
   * Sử dụng enum State để quản lý các trạng thái: CHECK\_IN, MAIN\_MENU, WORK\_MENU, TASK\_MENU, WORKING, CHECK\_OUT.
   * Hàm loop() xử lý đầu vào từ bàn phím và chuyển đổi trạng thái dựa trên phím nhấn.
9. **Thử nghiệm và gỡ lỗi**:
   * Kiểm tra giao tiếp RFID, hiển thị TFT, và phản hồi bàn phím.
   * Xác minh dữ liệu được gửi và nhận chính xác trên Firebase Console.
   * Sử dụng Serial Monitor để theo dõi trạng thái và lỗi.

**Cơ chế hoạt động**

Hệ thống hoạt động theo quy trình sau:

1. **Check-in**: Công nhân quét thẻ RFID, hệ thống gửi UID đến Firebase để lấy thông tin công nhân và hiển thị menu chính.
2. **Chọn công việc**: Công nhân chọn công việc (lắp ráp, sửa chữa, đóng gói) từ menu, chỉ hiển thị các công việc khả dụng.
3. **Thực hiện công việc**: Công nhân ghi nhận công việc hoàn thành hoặc lỗi qua bàn phím, dữ liệu được gửi lên Firebase.
4. **Check-out**: Công nhân quét thẻ RFID để xác nhận, hệ thống gửi tóm tắt công việc lên Firebase và hiển thị tổng quan công việc.
5. **Đồng bộ thời gian**: Dấu thời gian được lấy từ NTP để đảm bảo dữ liệu chính xác.

**Ưu điểm**

* **Thời gian thực**: Dữ liệu được đồng bộ ngay lập tức với Firebase, cho phép quản lý từ xa.
* **Giao diện thân thiện**: Màn hình TFT hiển thị thông tin bằng tiếng Việt, dễ sử dụng.
* **Bảo mật**: Xác thực RFID và khóa API Firebase đảm bảo an toàn dữ liệu.
* **Tính linh hoạt**: Hỗ trợ nhiều loại công việc và dễ dàng mở rộng.

**Hạn chế**

* **Phụ thuộc vào WiFi**: Mất kết nối mạng sẽ làm gián đoạn giao tiếp với Firebase.
* **Thiếu lưu trữ cục bộ**: Không có cơ chế lưu dữ liệu khi mất kết nối.
* **Xử lý lỗi hạn chế**: Chỉ kiểm tra cơ bản trạng thái HTTP, chưa xử lý các lỗi phức tạp như timeout hoặc JSON không hợp lệ.
* **Hiệu suất màn hình TFT**: Văn bản dài có thể bị cắt nếu không được xử lý cẩn thận.

**Đề xuất cải tiến**

* Thêm bộ nhớ đệm cục bộ để lưu dữ liệu khi mất kết nối WiFi, đồng bộ khi kết nối được khôi phục.
* Cải thiện xử lý lỗi HTTP với các cơ chế thử lại (retry) và thông báo chi tiết hơn.
* Tối ưu hóa hiển thị trên TFT bằng cách sử dụng font nhỏ hơn hoặc hỗ trợ cuộn văn bản.
* Tích hợp thông báo thời gian thực qua Firebase để cảnh báo quản lý khi có lỗi công việc.
* Thêm phân tích dữ liệu trên Firebase để tạo báo cáo hiệu suất công nhân.

**Kết luận**

Tích hợp Firebase vào hệ thống quản lý công việc dựa trên RFID cung cấp một giải pháp hiệu quả để theo dõi và quản lý công việc trong thời gian thực. Các bước triển khai bao gồm thiết lập Firebase, cấu hình phần cứng, xây dựng giao diện, và tích hợp giao tiếp HTTP đã được thực hiện thành công. Hệ thống đáp ứng tốt các yêu cầu cơ bản nhưng cần cải tiến để tăng độ tin cậy và tính năng. Trong tương lai, việc bổ sung lưu trữ cục bộ và phân tích dữ liệu sẽ nâng cao giá trị của hệ thống.