Universidade Federal de Lavras

Instituto de Ciências Exatas e Tecnológicas.

Departamento de Ciência da Computação

DISCIPLINA: INTERNET DAS COISAS - GCC271

Trabalho Prático 2

Relatório de Desenvolvimento Campus Inteligente:

Salas e Laboratórios

Discentes: Samuel Luiz Freitas Ferreira - 201820288

Thallys Henrique Martins - 201910647

Willian Brandao de Souza - 202220233

Data de entrega: 06 de julho de 2025

SUMÁRIO

INTRODUÇÃO

O presente relatório tem como objetivo descrever sobre o desenvolvimento do projeto de controle de acesso de Salas e laboratórios, ao longo do projeto serão discutidos os problemas e dificuldades enfrentados, além dos materiais utilizados e como se deu o desenvolvimento.

MATERIAIS E FERRAMENTAS UTILIZADAS

Para simular o controle de acesso às salas e avaliar as condições do ambiente, foram empregados os seguintes equipamentos e tecnologias:

- Sensor DHT11: Utilizado para monitorar a umidade e a temperatura do ambiente.
- Sensor RFID: Responsável pela leitura de cartões e tags RFID, essencial para o controle de acesso.
- Tags RFID e Cartões RFID: Dispositivos usados para a identificação e autenticação no sistema de controle de acesso.
- ESP32: Placa de desenvolvimento que serviu como o microcontrolador principal do projeto.
- Protoboard: Ambiente para a montagem e testes do circuito durante o desenvolvimento.
- Cabos: Utilizados para conectar os sensores à placa ESP32.

No que diz respeito ao desenvolvimento de software, o VS Code foi a ferramenta escolhida para a criação do Front-end. Para o armazenamento dos dados provenientes das leituras, utilizou-se o Firebase.

PROBLEMAS ENCONTRADOS

A principal dificuldade durante o desenvolvimento reside na identificação de pinos funcionais, um problema evidenciado pela inconsistência de funcionamento de um mesmo código em diferentes placas. A tentativa inicial foi remontar o circuito, mas mesmo assim, o sistema não solucionou.

Para contornar essa questão, desenvolveu-se um código de teste utilizando LEDs e resistores, o que permitiu identificar os pinos em pleno funcionamento. Contudo, mesmo após essa etapa, a integração com o módulo RFID não obteve êxito, consumindo um tempo significativo do desenvolvimento, uma vez que pensávamos que os problemas estavam nos pinos do ESP32, mas o que aparentemente não era o problema principal.

Posteriormente, a equipe optou por conectar diretamente o módulo RFID aos pinos funcionais da placa utilizando jumpers macho-fêmea. Essa abordagem revelou que o cerne do problema estava na comunicação do circuito na protoboard. Assim, tornou-se necessário realizar uma verificação detalhada, trilha a trilha, para garantir o correto funcionamento do circuito.

DESENVOLVIMENTO

SISTEMA DE CONTROLE DE ACESSO

Descrição

Nosso sistema visa o controle de acesso e o monitoramento ambiental de salas, utilizando sensores RFID e de umidade/temperatura. O principal objetivo é registrar a presença de indivíduos por meio de seus cartões RFID, além de acompanhar as condições climáticas do ambiente.

Adicionalmente, quando um usuário entra em uma sala, o sistema o registra e marca o ambiente como "ocupado". É fundamental destacar que, após o registro de entrada em uma sala, o usuário fica impedido de acessar qualquer outra sala até que sua saída da primeira sala seja devidamente registrada. Os dados de temperatura e umidade são transmitidos a cada 10 segundos para o Firebase, permitindo um registro contínuo e o controle ambiental.

Processo de Desenvolvimento

O desenvolvimento do controle de acesso e do monitoramento foi pautado na reutilização de códigos e conceitos aprendidos em aulas práticas da disciplina, complementados por pesquisas em materiais disponíveis na internet.

Inicialmente, focamos na conexão e interpretação dos dados dos sensores com o ESP32, por considerarmos a integração com o Firebase e as conexões de internet módulos relativamente mais simples, devido à experiência prévia em aulas. Dessa forma, pesquisamos e adaptamos códigos básicos de comunicação para:

- RFID: Implementamos a verificação inicial do RFID para identificar e exibir cartões autorizados e não autorizados.
- DHT11: Posteriormente, avançamos para a leitura e exibição dos dados de umidade e temperatura, tanto no *display* do ESP32 quanto no Monitor Serial da IDE Arduino.

Com a coleta e o funcionamento correto dos dados dos sensores, procedemos à implementação da comunicação com o Firebase. Para isso, adaptamos materiais de aula e exemplos existentes, ajustando-os às necessidades específicas do nosso projeto.

Estrutura de Dados no Firebase

A estrutura definida no Firebase foi projetada para ser escalável e intuitiva, permitindo a fácil adição de novas salas. A hierarquia principal é a seguinte:

```
None
salas
├──sala101
│   ├──numPessoas
│   ├──ocupada
│   ├──presentes
│   │   ├──CódigoRFID1: Horário
│   │   └──CódigoRFID1: Horário
│   ├──temperatura
│   │   ├──Horário1: temperaturaRegistrada1
│   │   └──Horário2: temperaturaRegistrada2
│   └──umidade
│       ├──Horário1: umidadeRegistrada1
│       └──Horário2: umidadeRegistrada2
└──sala102
    ├──numPessoas
    ├──ocupada
    ├──presentes
    │   ├──CódigoRFID1: Horário
    │   └──CódigoRFID1: Horário
    ├──temperatura
    │   ├──Horário1: temperaturaRegistrada1
    │   └──Horário2: temperaturaRegistrada2
    └──umidade
        ├──Horário1: umidadeRegistrada1
        └──Horário2: umidadeRegistrada2
```

Assim, a descrição dos atributos adotados no firebase são.

- `numPessoas`: Um valor inteiro que representa a quantidade de pessoas presentes na sala em tempo real.
- `ocupada`: Um indicador booleano (`true`/`false`). Seu valor é `false` quando `numPessoas` é igual a zero, servindo como um controle para verificar a presença de alguém na sala.
- presentes: registra os cartões que estão naquela sala.
- `temperatura`: Registra os valores de temperatura lidos a cada 10 segundos, com cada leitura indexada por seu respectivo horário.
- `umidade`: Registra os valores de umidade lidos a cada 10 segundos, também indexados por seu horário correspondente.

Essa padronização permite que, caso novas salas sejam incorporadas ao sistema, a estrutura de dados possa ser replicada de forma simples e eficiente.

A seguir seguem algumas definições utilizadas para o projeto tal como algumas funções utilizadas no desenvolvimento e suas explicações.

Bibliotecas Utilizadas

O sistema foi desenvolvido utilizando diversas bibliotecas para gerenciar as interações com hardware e serviços externos. Abaixo estão as principais e suas respectivas finalidades:

- SPI.h: Essencial para a comunicação SPI (Serial Peripheral Interface), que permite a troca de dados em alta velocidade entre o ESP32 e o módulo RFID.
- MFRC522.h: Biblioteca dedicada à interação com o sensor RFID MFRC522, facilitando a leitura de cartões e tags RFID.
- DHT.h: Permite a leitura de dados do sensor de temperatura e umidade DHT11, coletando informações ambientais da sala.
- WiFi.h: Habilita o ESP32 a se conectar a redes Wi-Fi
- Wire.h: Implementa a comunicação I2C (Inter-Integrated Circuit), protocolo utilizado para a interface com o display OLED.
- HT_SSD1306Wire.h: Uma biblioteca específica para o display OLED SSD1306 em placas Heltec V2, otimizando a exibição de informações visuais.
- Firebase_ESP_Client.h: Permite a comunicação e manipulação de dados no Firebase Realtime Database.
- addons/TokenHelper.h: Auxilia no gerenciamento de autenticação com o Firebase, garantindo a segurança das transações de dados.
- time.h: Oferece funções para o gerenciamento de tempo, crucial para obter timestamps precisos via NTP (Network Time Protocol) e registrar eventos.

Definição de Pinos e Constantes

Nosso grupo optou por definir constantes para os pinos do ESP32, associando cada componente a um pino específico. Essa prática melhora a legibilidade do código e facilita a manutenção, caso haja necessidade de alterar a pinagem de hardware. Ademais, como foi mencionado anteriormente, tivemos dificuldade em definir pinos funcionais, essa prática facilitou em testar diferentes pinos mais rapidamente. Além dos pinos, outras constantes foram estabelecidas para configurações importantes, como credenciais de rede Wi-Fi, informações de acesso ao Firebase e a lista de UIDs (identificadores únicos) de cartões RFID autorizados.

Funções Principais e Pontos Importantes do Código

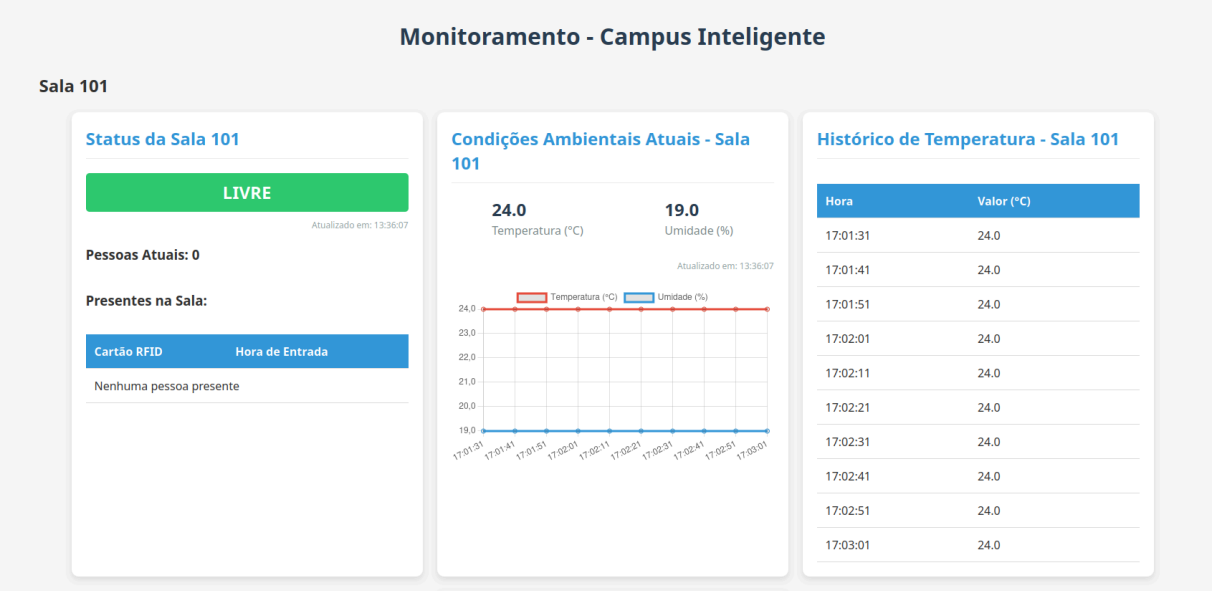
O código é estruturado em funções claras, cada uma com uma responsabilidade específica para organizar a lógica do sistema:

- Funções de Inicialização (`setup()`):
    - Configuração Inicial: O `setup()` é o ponto de partida do sistema, onde todos os módulos e serviços são inicializados. Isso inclui o display OLED, o módulo RFID, o sensor DHT11, a conexão Wi-Fi e a integração com o Firebase.
    - Sincronização de Tempo: Uma etapa crucial é a sincronização do tempo via NTP, garantindo que todos os registros no Firebase tenham *timestamps* precisos, essenciais para o rastreamento de eventos de acesso e leituras de sensores.
- Loop Principal (`loop()`):
    - `handleRFID()` (Controle de Acesso): Esta função gerencia a leitura de cartões RFID. Ela verifica se um cartão é autorizado e, em caso positivo, determina se o usuário está entrando ou saindo da sala, atualizando a contagem de pessoas e o status de ocupação no Firebase. Há também uma validação para impedir que um usuário seja registrado em duas salas simultaneamente.
    - `handleSensors()` (Monitoramento Ambiental): Responsável por periodicamente (a cada 10 segundos) ler os dados de temperatura e umidade do sensor DHT11. Após a leitura, os dados são exibidos no display OLED e enviados para o Firebase, permitindo o monitoramento ambiental em tempo real.
- Funções Auxiliares:
    - Comunicação com Firebase: Funções dedicadas garantem a correta interação com o Firebase, como `isPresentInOtherRoom` (para verificar a presença em outras salas) e `updateFirebaseSensors` (para enviar os dados dos sensores com *timestamps*).
    - Tratamento de Dados RFID: A função `getUID` é utilizada para processar e formatar o identificador único do cartão RFID de forma consistente.
    - Exibição no Display OLED: Múltiplas funções (`displayMessage`, `displayAccessDeniedOtherRoomSequence`, `displayAccessInfo`, `displaySensorData`) são empregadas para exibir mensagens claras e informativas no display OLED, fornecendo *feedback* visual ao usuário sobre o status do acesso e as condições ambientais.

SISTEMA WEB PARA VISUALIZAÇÃO

Considerando que o enfoque da disciplina não abrangia o desenvolvimento web e que uma parcela significativa do tempo foi dedicada à resolução de questões relacionadas à conectividade dos pinos, optou-se por solicitar a uma ferramenta de *IA conversacional* a elaboração da interface de visualização para o sistema. O *layout*

gerado, obtido por meio de um comando simplificado, foi considerado satisfatório e demandou apenas ajustes pontuais por parte da equipe.



A página de visualização, portanto, oferece um dashboard completo para o monitoramento das salas. Ela permite verificar rapidamente se uma sala está livre ou ocupada, bem como a identidade dos usuários presentes e o horário de sua entrada no sistema.

Além disso, a interface inclui um gráfico dinâmico que exibe os 10 últimos valores registrados de temperatura e umidade, proporcionando uma visão clara das condições ambientais. Complementando o gráfico, há duas tabelas de registro: uma dedicada ao histórico de temperatura e outra ao de umidade, detalhando as leituras ao longo do tempo.

Se for preciso adicionar novas salas, o caminho atual é replicar o código existente. Como o desenvolvimento front-end não era a prioridade do nosso projeto, não dedicamos tempo significativo para reestruturar essa parte.

No entanto, com a forma como os dados foram implementados no Firebase, o front-end poderia ser facilmente otimizado. Em vez de duplicar o código, seria possível tratar cada sala como um único elemento. Isso significaria usar um vetor (ou array) de salas, onde cada sala seria um item nesse vetor, e o sistema web a renderizaria dinamicamente. Para adicionar uma nova sala, bastaria incluir um novo elemento nesse vetor, o que simplificaria bastante a expansão do *dashboard*.

REFERÊNCIAS

https://github.com/miguelbalboa/rfid
https://github.com/adafruit/DHT-sensor-library
https://github.com/mobizt/Firebase-ESP-Client
https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series