INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA - CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JHONATAN GUILHERME
MARIA EDUARDA
MATEUS PIERRE

SISTEMAS EMBARCADOS
PROJETO FINAL - CONTROLE DE ACESSO

Campina Grande
2023

INTRODUÇÃO

No cenário dinâmico das cidades inteligentes, aprimorar a gestão e operação de instituições educacionais é crucial. Esta proposta de projeto visa implementar um sistema inteligente de controle e segurança para uma sala de aula, inserida no contexto de um Smart Campus, especificamente voltado ao laboratório de sistemas embarcados controle e automação do IFPB.

O cerne do sistema reside na capacidade de identificar e controlar o acesso de alunos e servidores, proporcionando não apenas segurança aprimorada, mas também eficiência na administração da presença. A coleta de informações de identificação dos usuários será realizada utilizando a conexão bluetooth e uma validação interna no sistema.

Além de prevenir a entrada de pessoas não autorizadas, o sistema propõe sua evolução para um projeto capaz de automatizar a verificação de presença, otimizando os processos de registro e permitindo uma gestão mais eficaz. Dessa forma, o projeto não apenas eleva os padrões de segurança, mas também oferece uma solução integrada para a administração de acessos e presenças na sala ou laboratório, contribuindo para um ambiente acadêmico mais inteligente e eficiente.

Funcionamento:

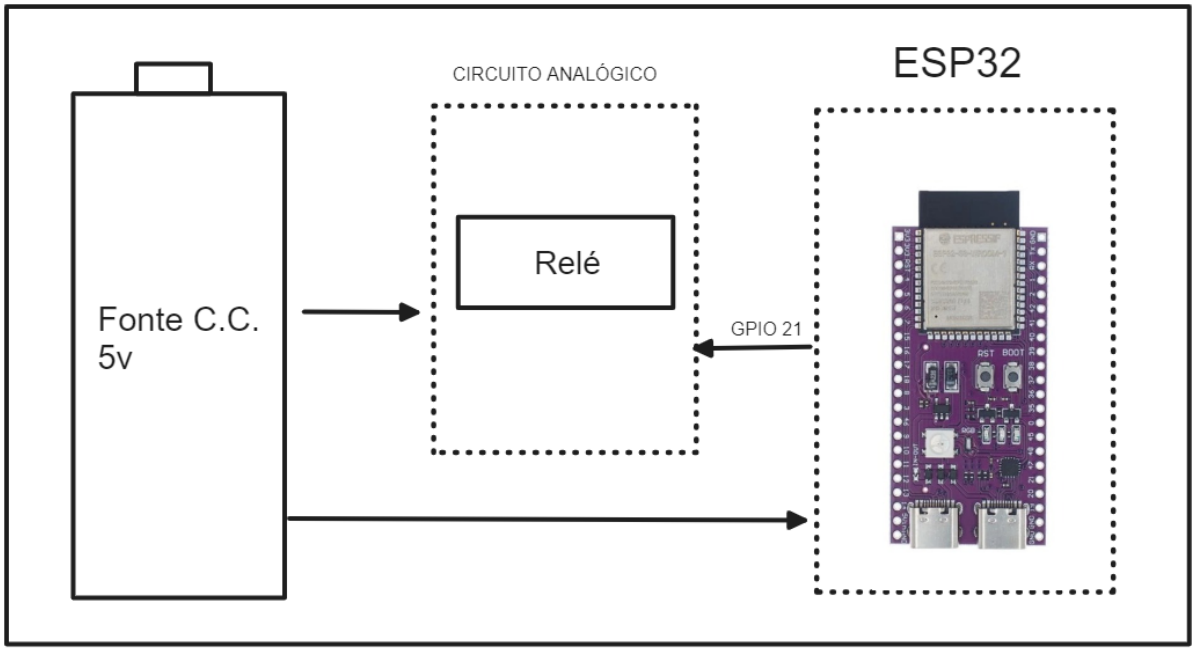
a. 1 ESP32 - utilizando Bluetooth para se conectar ao aplicativo e enviar as credenciais pelo Bluetooth
b. Beaglebone como gateway;
c. Middleware (MQTT) protocolo para se comunicar com o Beaglebone;
d. Conexão em nuvem para sincronizar credenciais

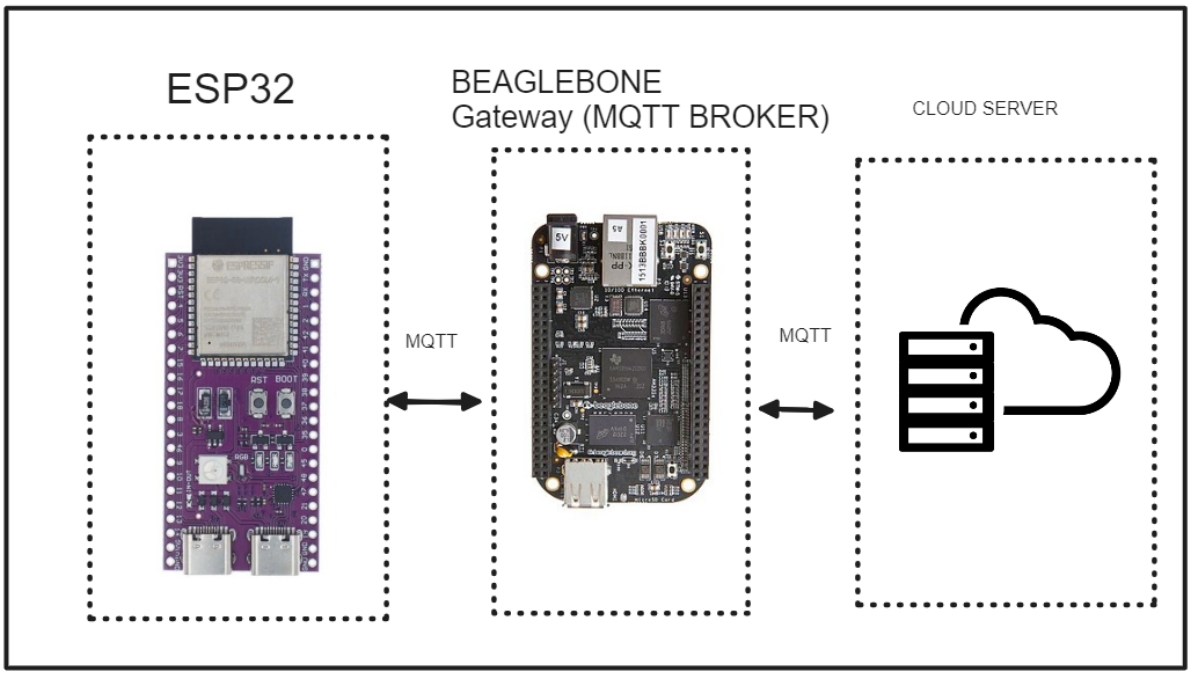
Componentes:

Descrição	Quantidade
ESP32 WROOM	1
Fonte de alimentação	1
Módulo relé 1 canal	1
BeagleBone	1
Fios	3

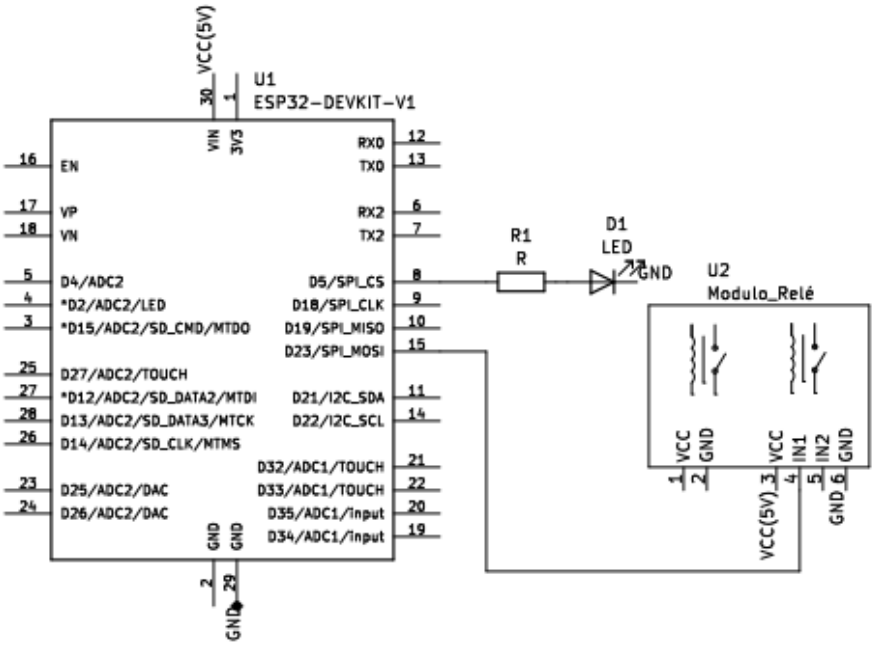
1. Hardware:

Diagrama de bloco do dispositivo:



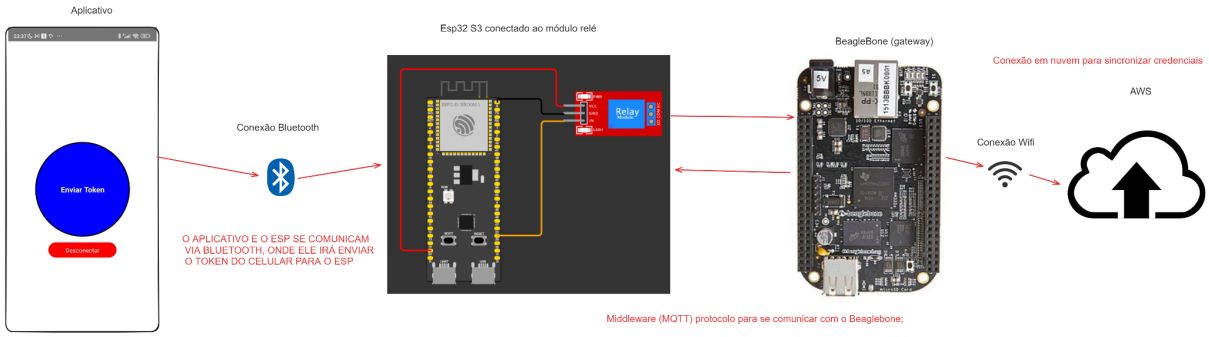


Esquema elétrico:



*O led está no diagrama pois foi adicionado ao protótipo para melhor visualização da ativação do módulo

Protótipo do dispositivo:



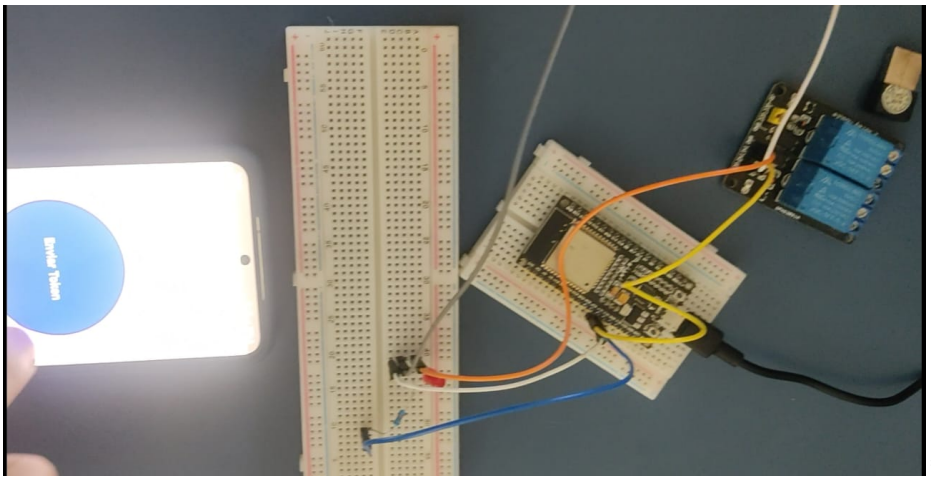


*Foto real do protótipo

2. Máquina de estado:

    1. Estado Inicial (Inicialização):

Descrição: O ESP inicializa o módulo Bluetooth.
Transição: Após a inicialização, passa para o estado "Aguardando Conexão".

    2. Estado Aguardando Conexão:

Descrição: O ESP aguarda a conexão de um dispositivo móvel através do aplicativo.
Transição: Quando um dispositivo se conecta, passa para o estado "Recebendo Token".

    3. Estado Recebendo Token:

Descrição: O ESP aguarda o recebimento do token enviado pelo aplicativo.
Transição: Ao receber o token, passa para o estado "Enviando Token para BeagleBone".

    4. Estado Enviando Token para BeagleBone:

Descrição: O ESP envia o token para a BeagleBone utilizando o protocolo MQTT.
Transição: Após o envio, passa para o estado "Aguardando Resposta da BeagleBone".

    5. Estado Aguardando Resposta da BeagleBone:

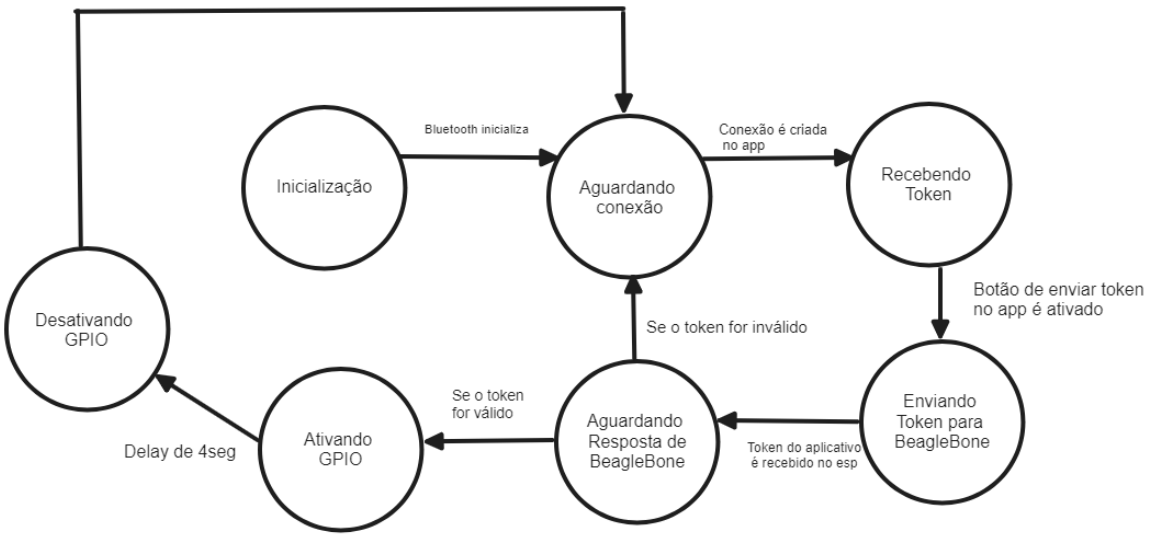
Descrição: O ESP aguarda a resposta da BeagleBone sobre a validade do token.
Transições:
Se o token for válido, passa para o estado "Ativando GPIO".
Se o token for inválido, retorna para o estado "Aguardando Conexão".

    6. Estado Ativando GPIO:

Descrição: O ESP ativa a GPIO conectada ao relé.
Transição: Após ativar a GPIO, passa para o estado "Desativando GPIO" (tempo de espera de 4 segundos).

    7. Estado Desativando GPIO:

Descrição: O ESP desativa a GPIO conectada ao relé após 4 segundos.
Transição: Após desativar a GPIO, retorna para o estado "Aguardando Conexão".



3. Bibliotecas

stdio.h	Biblioteca padrão de entrada/saída em C.
string.h	Biblioteca padrão de manipulação de strings em C.
freertos/FreeRTOS.h e freertos/task.h	Bibliotecas relacionadas ao FreeRTOS, um sistema operacional em tempo real para sistemas embarcados. Elas são usadas para criar tarefas e gerenciar o sistema de tempo real.
freertos/event_groups.h	Usada para criar e gerenciar grupos de eventos no FreeRTOS. Os eventos são usados para sincronização entre tarefas.
esp_event.h	Biblioteca relacionada a eventos no ESP-IDF (Espressif IoT Development Framework). Usada para gerenciar eventos no sistema.
nvs_flash.h	Biblioteca para operações de armazenamento não volátil (NVS). Usada para armazenar configurações persistentes.
esp_log.h	Biblioteca para logs no ESP-IDF. Usada para gerar logs em diferentes níveis (debug, info, erro, etc.).

esp_nimble_hci.h	Biblioteca relacionada à configuração do controlador Bluetooth de baixa energia no ESP-IDF.
nimble/nimble_port.h e nimble/nimble_port_freertos.h	Bibliotecas relacionadas ao NimBLE, uma pilha Bluetooth de baixa energia usada no ESP-IDF.
host/ble_hs.h	Biblioteca relacionada ao NimBLE Host, que lida com o lado do host da pilha Bluetooth de baixa energia.
services/gap/ble_svc_gap.h e services/gatt/ble_svc_gatt.h	Bibliotecas relacionadas aos serviços GAP (Generic Access Profile) e GATT (Generic Attribute Profile) no Bluetooth.
sdkconfig.h	Arquivo gerado pelo ESP-IDF durante a compilação. Contém configurações específicas do projeto.
driver/gpio.h	Biblioteca para operações GPIO no ESP-IDF. Usada para configurar e controlar pinos GPIO.
MQTTClient.h	Biblioteca para implementar o protocolo MQTT no ESP-IDF. Usada para comunicação MQTT.
Essas bibliotecas são essenciais para o desenvolvimento de aplicativos para ESP32 usando o ESP-IDF, proporcionando funcionalidades como manipulação de GPIO, BLE, armazenamento, comunicação MQTT, entre outras.	

DESCRIÇÃO DAS ATIVIDADES

Este projeto visa criar um sistema de controle de acesso usando um dispositivo ESP32, permitindo a ativação ou desativação de um dispositivo físico (representado por um módulo relé) por meio de comunicação Bluetooth de Baixa Energia (BLE) e protocolo MQTT, utilizando um BeagleBone como gateway e um aplicativo móvel para interação.

Atividades Realizadas:

Configuração do Ambiente de Desenvolvimento:

- Instalação e configuração do ambiente de desenvolvimento ESP-IDF (Espressif IoT Development Framework).
- Configuração do FreeRTOS para gerenciamento de tarefas.

Configuração do Bluetooth de Baixa Energia (BLE):

- Implementação do servidor BLE para comunicação com outros dispositivos Bluetooth.
- Definição de serviços e características BLE para leitura e escrita de dados.

Configuração do NimBLE (Pilha Bluetooth de Baixa Energia):

- Integração do NimBLE para gerenciamento do controlador Bluetooth de baixa energia no ESP32.

Configuração do MQTT:

- Implementação de funções para enviar mensagens MQTT para um servidor MQTT externo.
- Configuração de tópicos MQTT para enviar e receber mensagens relacionadas ao controle remoto.

Implementação da Lógica de Controle:

- Definição de um módulo relé físico (pino GPIO) como dispositivo controlado.
- Implementação de uma função para acionar ou desacionar o relé com base em mensagens MQTT recebidas.

Integração com BLE e MQTT:

- Configuração de uma função para processar dados recebidos por Bluetooth e enviá-los como requisições MQTT.
- Subscrição a um tópico MQTT para receber confirmações e acionar/desacionar o relé físico conforme necessário.

Configuração do Ambiente de Desenvolvimento para BeagleBone:

- Instalação e configuração do ambiente de desenvolvimento para a BeagleBone.
- Configuração do sistema operacional na BeagleBone.
- Configuração do BeagleBone como Gateway:
- Implementação de lógica para tradução e retransmissão de mensagens entre Bluetooth e MQTT.

Desenvolvimento do Aplicativo Móvel:

- Criação de um aplicativo móvel (Android) para interação com o ESP32 via Bluetooth.
- Implementação de interface gráfica para ativar/desativar o controle remoto.

4. APP -> Código (Repositório)

RESULTADOS

Durante o desenvolvimento do projeto, obtivemos resultados positivos em várias frentes, destacando conquistas significativas em áreas-chave. A seguir, apresento uma visão geral dos resultados obtidos:

Conexão Bluetooth:

A implementação da conexão Bluetooth foi bem-sucedida, proporcionando uma comunicação estável entre os dispositivos. Essa funcionalidade é crucial para o projeto, permitindo a interação efetiva entre o ESP32 e outros dispositivos.

Aplicativo Móvel:

A aplicação móvel desenvolvida alcançou os objetivos estabelecidos. A interface do usuário foi projetada para proporcionar uma experiência amigável e intuitiva, permitindo o controle efetivo do sistema por meio de dispositivos móveis.

Controle por Relé:

A integração e operação dos relés atenderam às expectativas. Esses componentes foram essenciais para controlar a ativação e desativação de dispositivos físicos, como a tranca, conforme as necessidades do projeto.

BeagleBone:

O BeagleBone desempenhou seu papel de maneira eficaz, contribuindo para a integração do sistema e fornecendo uma plataforma sólida para a execução de funções específicas do projeto. Sua operação está alinhada com as expectativas.

Conexão entre BeagleBone e ESP32:

A conexão entre o BeagleBone e o ESP32 apresentou desafios, com resultados um pouco abaixo do esperado. Apesar disso, a comunicação entre esses dispositivos foi estabelecida, permitindo funcionalidades básicas, mas pode exigir ajustes adicionais para otimizar a interação.

Embora tenhamos enfrentado algumas dificuldades na conexão entre o BeagleBone e o ESP32, é importante ressaltar que o projeto como um todo obteve sucesso em vários aspectos cruciais. O trabalho contínuo na otimização dessa conexão específica pode ser necessário para aprimorar ainda mais a integração entre esses dois elementos-chave do sistema.

CONSIDERAÇÕES FINAIS

Este projeto demonstra a integração de tecnologias como BLE e MQTT para criar um sistema de controle de acesso. Pode ser expandido para incluir mais dispositivos e funcionalidades, oferecendo uma base sólida para projetos de IoT (Internet das Coisas) com o ESP32.