

Fechadura Eletrônica com Leitor de RFID

Rebecca Quintino Do Ó

rebeccaquintino@gmail.com, Engenharia Elétrica
EEL7323 - 08235

1 Introdução

O projeto tem como finalidade desenvolver uma fechadura eletrônica usando um módulo leitor rádio-frequência (RFID) modelo MFRC522. Além disso, terá o uso de um Teclado Matricial de Membrana de 16 Teclas para permitir que o usuário cadastre novas tags através de uma senha. Ademais, temos um botão de reset para a fechadura ser destravada a qualquer momento dentro do ambiente. Todos esse sensores estarão conectados a um microcontrolador ESP32S ESP-WROOM-32 que também estará conectado aos leds de sinalização, nosso atuador. Lembrando que os logs de leituras das tags estarão sendo enviados ao computador que também estará conectado ao microcontrolador.

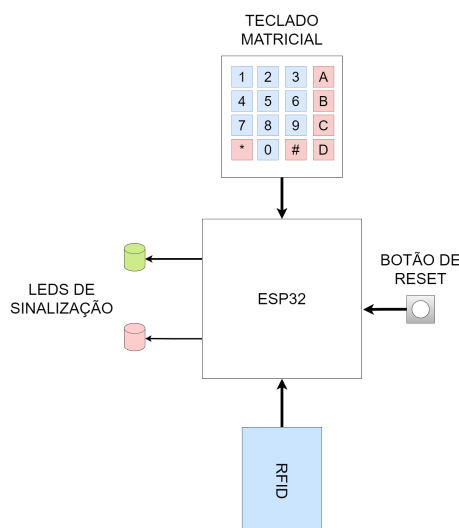


Figure 1: Diagrama simplificado do projeto

2 Sistema Embarcado

2.1 Funcionamento

Como mencionado anteriormente o sensor de RFID será o responsável por ler e gravar a tag. Assim, para entrar no modo de leitura é necessário que o usuário aproxime a tag no sensor e após verificar se a tag tem seu buffer igual a um

character específico, a porta se abrirá. Por outro lado, para entrar no modo de gravação, é necessário que a pessoa aperte uma tecla qualquer para posteriormente digitar uma senha específica. Dessa forma, ela conseguirá gravar algum dado na tag e posteriormente gravar o identificador da tag no sensor RFID. Para que o usuário consiga digitar a senha teremos um Teclado Matricial de 16 teclas. Caso também a pessoa esteja dentro do ambiente e queira abrir a fechadura eletrônica, terá disponível um botão assíncrono que abrirá a porta automaticamente. Temos um resumo do funcionamento descrito neste fluxograma abaixo.

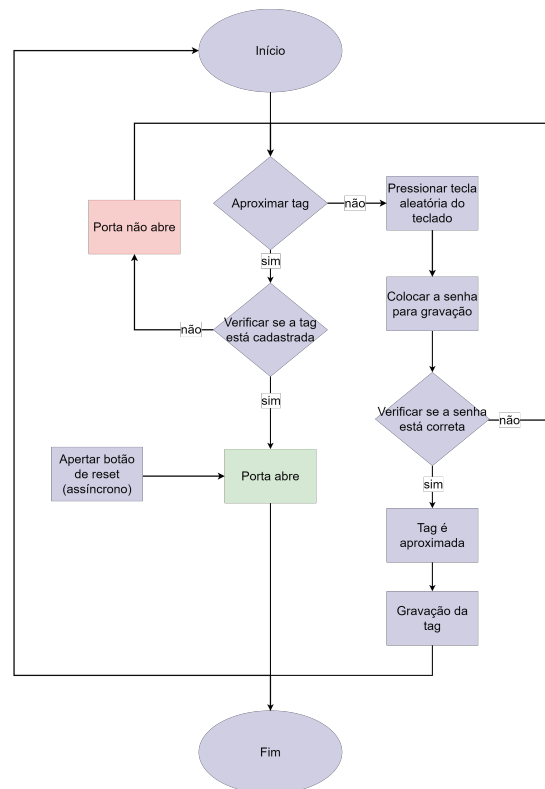


Figure 2: Fluxograma de funcionamento

2.2 Classes

Implementando todo o conjunto de forma embarcada e utilizando programação C++ com programação orientada à objetos, foi criada a classe *"Peripheral"*, a classe base, que terá como derivadas as classes *"Keyboard"* e *"Buttons"*. Essa classe base terá como função membro virtual *take_action()* e a função *Friend init()*, responsáveis pela inicialização do periférico e seu funcionamento básico. As classes derivadas implementarão o Teclado Matricial e botão responsável pela abertura da porta por dentro do ambiente. Ademais, teremos a classe *"RFID"* utilizará a biblioteca externa do RFID para implementar as funções de leitura *read_tag()* e de gravação *write_tag()*. Além disso, utilizando polimorfismo através de um ponteiro do tipo *Peripherals*, esse ponteiro vai acessar os

resultados adquiridos dos sensores, apontando para o periférico específico. Isso tudo será usado na função *handle_events* para abrigar os inputs e fazer a lógica de funcionamento do sistema.

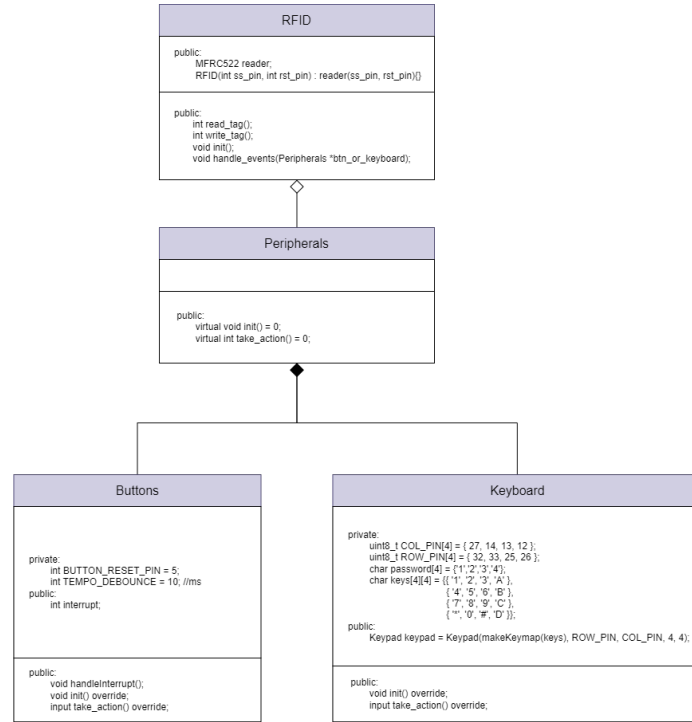


Figure 3: Diagrama de classes RFID, Peripherals, Buttons e Keyboard

3 Software Computador

3.1 Envio dos logs

Inicialmente, o sistema embarcado enviará logs com informações de entrada de pessoas e saídas. Dessa forma, utilizaremos as classes *ClockCalendar*, *Queue* e *Node* de forma embarcada para enviar os logs de entrada e saída juntamente com a data e hora do acesso. A classe *ClockCalendar* utiliza as bibliotecas **WiFi**, **NTPClient** e **WiFiUdp** para logar o ESP32 com a rede wifi e assim obter a hora e a data da ação a partir da função *now()*. O struct *Node* é o responsável por criar os nós para serem adicionados em uma lista na classe *Queue*. É usado *Sobrecarga do operador%* para concatenar os dados de acesso com a data/hora e ação realizada, esses serão inseridos na lista a partir da função *insert()* da classe *Queue* e depois enviados via UART para o computador na função *printLog()*.

3.2 Recepção dos logs

A recepção dos logs como já comentado, vai ser realizada com a leitura da porta serial do computador onde está conectado o microcontrolador. Foi feita uma

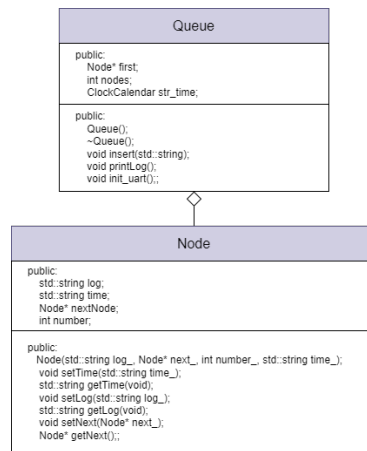


Figure 4: Diagrama de classes Queue e Node

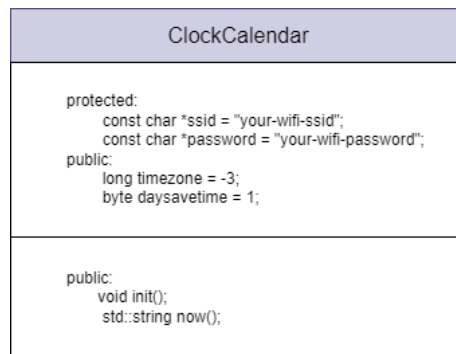


Figure 5: Diagrama de classes do ClockCalendar

classe *UART* para encapsular as ações da UART, dentre elas:

- O construtor *UART()* para abrir e configurar a porta serial;
- O destrutor *~UART()* para fechar a porta serial;
- O método *isValid()* para verificar se a porta serial é válida.
- O método *readData()* lê os dados vindos da serial.

Foi feita também a classe *DataLogger* para encapsular as ações relacionadas a criação do .txt, dentre elas:

- O construtor *DataLogger()* para escrever e concatenar os arquivos já criados em um .txt;
- A função *logData()* para enviar os logs para o arquivo .txt;
- O método *isValid()* para verificar se o arquivo é válido.

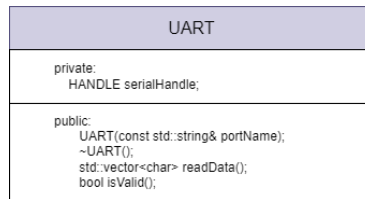


Figure 6: Diagrama de classes UART

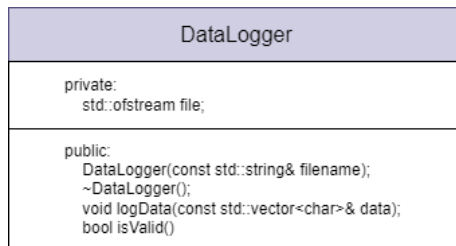


Figure 7: Diagrama de classes do DataLogger

Este código possui alguns tratamentos de erros para lidar com possíveis falhas durante a abertura da porta serial, configuração da porta serial, leitura da porta serial, abertura do arquivo e gravação no arquivo. Os principais pontos relacionados ao tratamento de erros são a abertura da Porta Serial e configuração da Porta Serial (construtor UART), leitura da Porta Serial (readData method), abertura do arquivo (construtor DataLogger), gravação no arquivo (logData). Além desses tratamentos de erros específicos, o código também verifica se ambas as instâncias (UART e DataLogger) são válidas antes de entrar no loop principal em main, garantindo que as operações dentro do loop sejam realizadas apenas se as condições necessárias forem atendidas.

3.2.1 Funcionamento

Será possível efetuar a recepção dos dados através das portas USB do computador. Ao abrir o terminal e executar o arquivo *main.exe* o usuário poderá escolher se quer fazer a leitura dos dados ou efetuar o filtro. Na opção leitura será mostrado em tempo real os logs que estão sendo feitos no leitor. Na opção de filtragem o usuário poderá filtrar por data, desse modo mostrando todos os logs realizados naquele período. Lembrando que os logs serão apresentados da seguinte forma: **Date: ano-mês-dia - Time: hora:minuto - UID: uid_da_tag - Status: status**

4 Plano de testes

4.1 Etapa 1: Inspeção dos subsistemas

Objetivo: Verificar se os componentes que serão utilizados estão operando adequadamente.

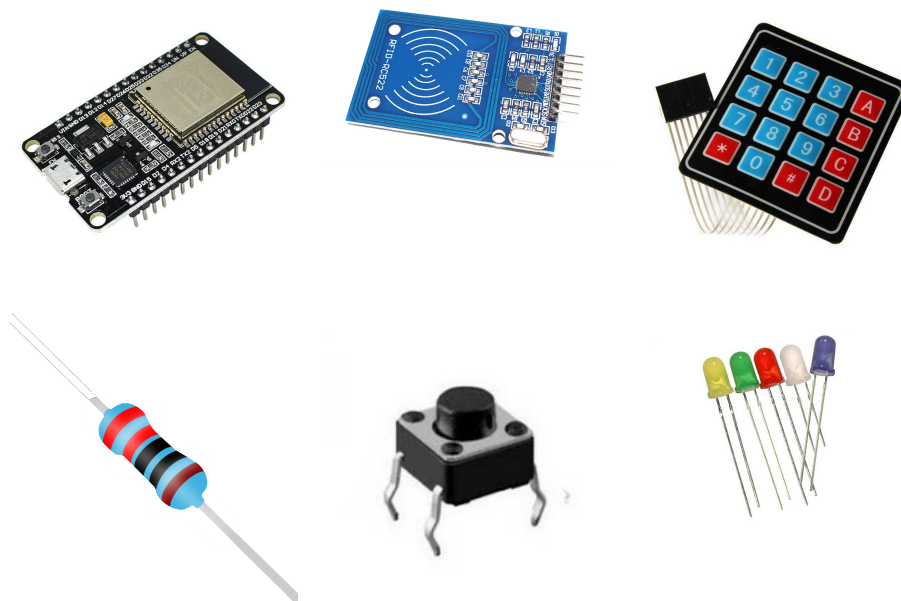


Figure 8: Componentes usados

Nesta fase, certifique-se de que os dispositivos a serem empregados não apresentem curtos em seus módulos e estejam em pleno funcionamento. Dê atenção especial ao microcontrolador, assegurando-se de que estava operacional antes de ser empregado neste projeto. No caso dos demais componentes, verifique se os LEDs e os resistores de 220R não estão danificados e se o botão está funcionando corretamente.

4.2 Etapa 2: Montagem

Objetivo: Montagem do circuito conforme a *Figura 9*.

4.3 Etapa 3: Inspeção Visual

Objetivo: Verificar a existência de algum curto *Figura 9*.

Após a montagem verificar a existência de algum curto com o uso de um multímetro *Figura 11*. Lembre-se de que o microcontrolador não deve estar conectado na alimentação nesta etapa. Se houver algum curto, refazer a montagem conforme a *Figura 9*.

4.4 Etapa 4: Upload do Firmware no ESP32

Objetivo: Realizar o upload do firmware no microcontrolador.

Para garantir uma operação adequada, é essencial inserir os dados da rede na qual o sistema operará. Isso pode ser feito acessando o arquivo `ClockCalendar.hpp` e modificando os parâmetros `"ssid-wifi"` e `"password"` de acordo

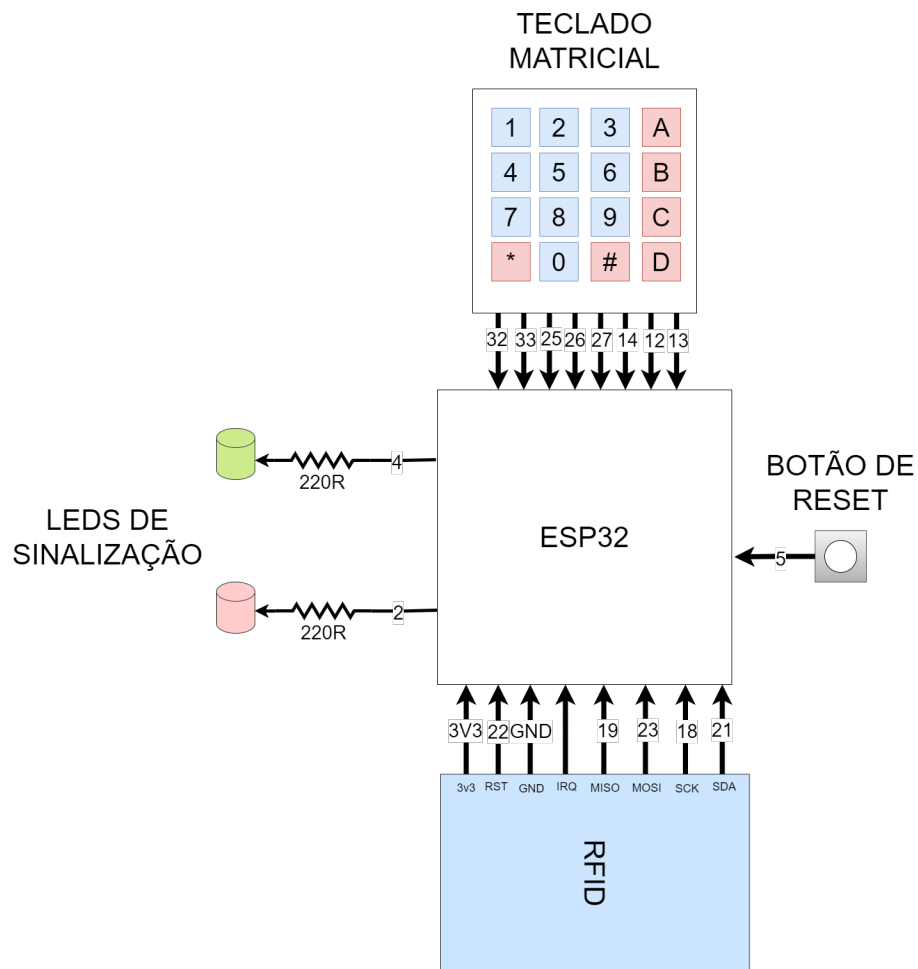


Figure 9: Conexões do circuito descritos pela GPIO



Figure 10: Equipamento utilizado para a inspeção visual

com as informações da sua rede. É crucial que a rede opere em 2.4GHz, pois o ESP32 não é compatível com redes de 5GHz.

4.5 Etapa 5: Teste do Sistema Embarcado

Objetivo: Verificar se o firmware do sistema embarcado está funcionando de acordo.

Para esta etapa é necessário que o microcontrolador esteja conectado a porta USB do seu computador. Além disso, é necessário que o sistema operacional seja o Windows. Após essas verificações você irá executar o arquivo main.exe usando o terminal e selecione a opção de leitura de dados da serial, para assim podermos visualizar os logs do sistema embarcado. Posteriormente, iniciaremos a testagem seguindo o fluxograma da *Figura 2*:

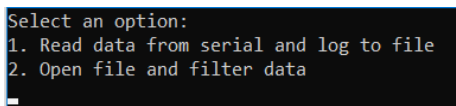
1. Aproxime uma tag não registrada no sistema; o LED vermelho deve piscar e um log com status de "no entry" deve aparecer no terminal e no .txt gerado.
2. Cadastrar a tag. Para isso pressione uma tecla aleatória no teclado matricial e em seguida insira a senha para efetuar o cadastro (senha padrão: 1234) e aproxime a tag por 5 segundos.
3. Aproxime a tag registrada; o LED verde deve piscar e um log com status de "entry" deve aparecer no terminal e no .txt gerado.
4. Aperte o botão; o LED verde deve piscar.

Se algum dos itens não funcionar, refazê-los ou retornar para a *Etapa 4.1*.

4.6 Etapa 6: Teste do Filtro do Software do PC

Objetivo: Avaliar o correto funcionamento do filtro de data no software do PC.

Para realizar esta etapa, é imprescindível que o microcontrolador esteja conectado à porta USB do computador e que o sistema operacional seja o Windows. Após essas verificações, execute o arquivo main.exe pelo terminal e selecione a opção de filtro de dados. Isso permitirá visualizar os registros do sistema embarcado de acordo com uma data específica. O usuário deve inserir a data no formato ano-mês-dia para realizar a filtragem. Dessa maneira, todos os logs ocorridos naquela data serão exibidos no terminal.



```
Select an option:
1. Read data from serial and log to file
2. Open file and filter data
_
```

Figure 11: Executando o main.exe