Fechadura Eletrônica com Leitor de RFID

Rebecca Quintino Do Ó

rebeccaqquintino@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. Ademas, 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 microncontrolador.

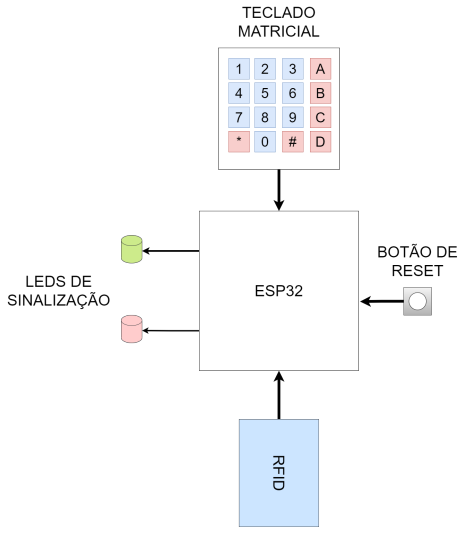

Figure 1: Diagrama simplificado do projeto

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 psoteriormente digitar uma senha específica. Dessa forma, ela consiguirá gravar algum dado na tag e posteriormente gravar o identicador 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 funcionamente descrito neste fluxograma abaixo.

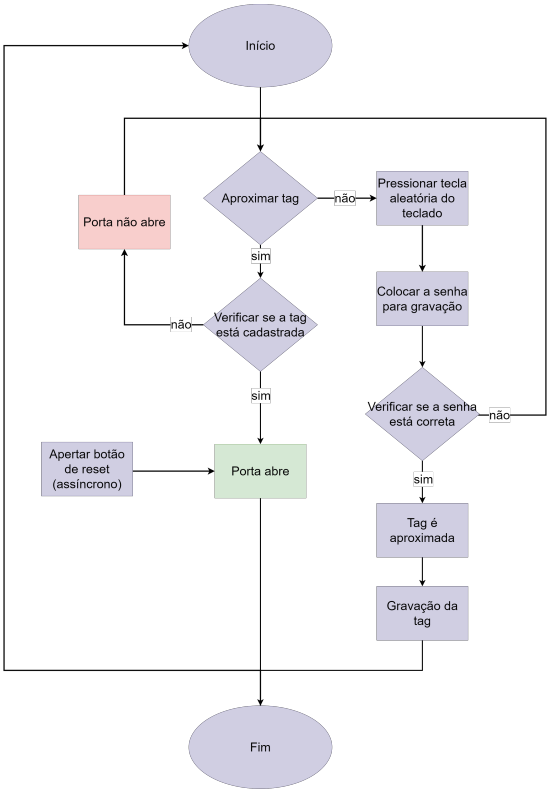

Figure 2: Fluxograma de funcionamento

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. Ademas, 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.

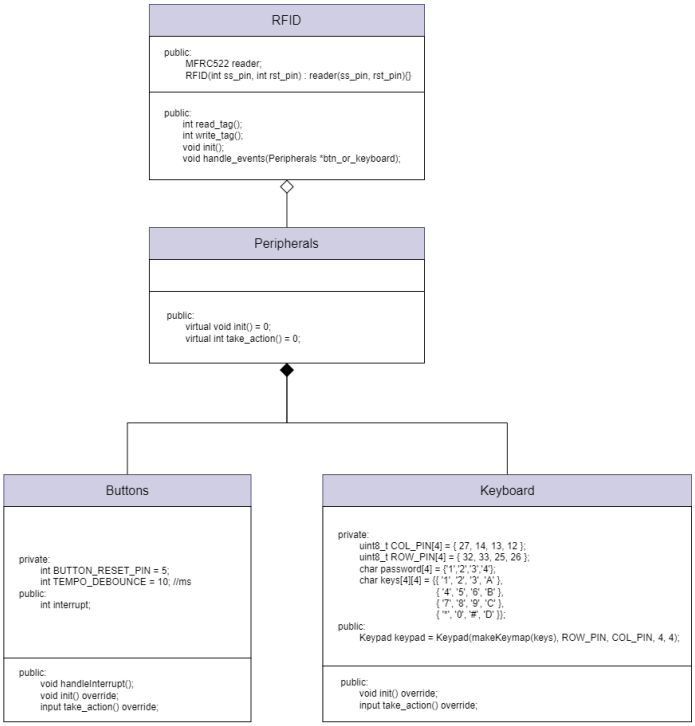

Figure 3: Diagrama de classes RFID, Peripherals, Buttons e Keyboard

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

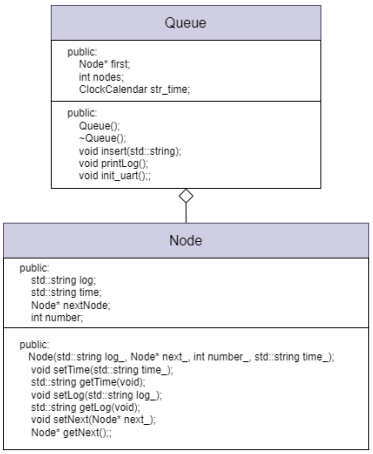

Figure 4: Diagrama de classes Queue e Node

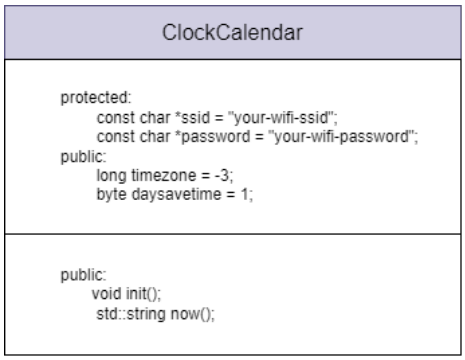

Figure 5: Diagrama de classes do ClockCalendar

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

- O construtor *UART()* para abrir e configurar a porta serial;
- O destrutor *ÛART()* 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 encapcular 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.

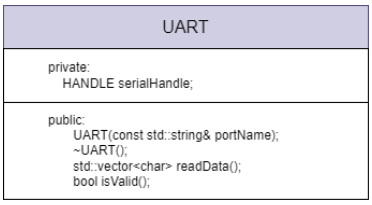

Figure 6: Diagrama de classes UART

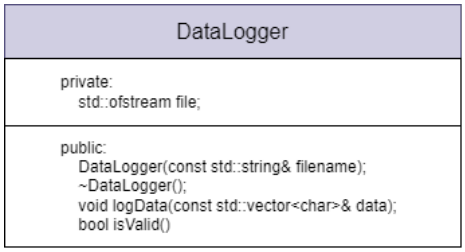

Figure 7: Diagrama de classes do DataLogger

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 algum tipo de filtro. Na opção leitura ele verá em tempo real os logs que estão sendo feitos no leitor. Na opção de filtragem o usuário poderá filtrar quais logs vai querer ver, se serão os de entrada (entry), saída (exit) ou nenhuma entrada (no entry). Lembrando que os logs serão apresentados da seguinte forma: **Date: ano-mês-dia - Time: hora:minuto - UID: uid_da_tag - Status: status**