CENTRO UNIVERSITÁRIO DE VALENÇA – UNIFAA

CURSO DE ENGENHARIA ELÉTRICA

JOSÉ RENATO SOUZA DA SILVA – 22717

LEANDRO DA CRUZ LARANJA – 22218

LÍVIA SILVA DA MORAES – 22853

LUAN GERALDO GUSTAVO DOS REIS – 21213

LUANA APARECIDA BRUNO MENDES – 21481

FECHADURA ELETRÔNICA COM MICROCONTROLADOR PIC18F4550

VALENÇA

2023

JOSÉ RENATO SOUZA DA SILVA – 22717
LEANDRO DA CRUZ LARANJA – 22218
LÍVIA SILVA DA MORAES – 22853
LUAN GERALDO GUSTAVO DOS REIS – 21213
LUANA APARECIDA BRUNO MENDES – 21481

FECHADURA ELETRÔNICA COM MICROCONTROLADOR PIC18F4550

Projeto Integrador apresentado ao Centro Universitário de Valença – UNIFAA, como requisito para avaliação nas disciplinas de Eletrônica Analógica e de Microcontroladores.

Orientadores: José Eduardo Henriques da Silva, Mariana Brinoti Altomar

VALENÇA

2023

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

As disciplinas PIN de Eletrônica Analógica e de Microcontroladores, ministradas pelos professores Mariana Brinoti Altomar e José Eduardo Henriques da Silva, respectivamente, requerem um projeto integrador (PIN) envolvendo ambas e a partir disso, foi elaborado pelos professores um projeto envolvendo uma fechadura eletrônica. A partir disso, os alunos deveriam se dividir em grupos de até 5 membros e, construir e programar um circuito de fechadura eletrônica que cumprisse os requisitos de hardware e de software estipulados no roteiro do projeto (mencionados no capítulo 2.1 e 2.2, respectivamente).

1.1 DESCRIÇÃO DO PROJETO

O projeto de fechadura eletrônica deve ser baseado em microcontroladores PIC e conter os seguintes componentes: um teclado matricial (4x4 ou 4x3), um display LCD 16x2, um buzzer, um sistema de fechadura que pode ser um relé ou um servomotor e elementos de eletrônica analógica para proteção, entretanto, deve haver nele no mínimo, um regulador de tensão de 5V.

Seu funcionamento deve ocorrer da seguinte maneira: o usuário deve digitar no teclado matricial uma senha e pressionar um botão de confirmação. Se a senha digitada for correta, circuito deverá realizar as seguintes tarefas: o buzzer deverá emitir 2 *bips* curtos, o display LCD deverá exibir uma mensagem de sucesso e um sinal precisa ser enviado para o dispositivo de fechadura, que o abrirá. Caso a senha esteja incorreta, a fechadura não abrirá, o display LCD informará que a senha incorreta e o buzzer irá disparar um *bip* longo.

1.2 OBJETIVOS

O objetivo principal é montar e programar um circuito de fechadura eletrônica, com os componentes obrigatórios (citado no capítulo 2.1) e com as funcionalidades exigidas no software (descrito no capítulo 2.2).

1.3 JUSTIFICATIVA

O critério para a escolha do microcontrolador foi de acordo com as necessidades do projeto, entretanto, durante os debates dos integrantes sobre a escolha, foi determinado o PIC16F628A por conta deste PIC ser o utilizado nas aulas de microcontroladores e haver mais familiaridade com ele, entretanto, durante a fase planejamento da montagem “virtual”, foi encontrado certa dificuldade em relação a quantidade de pinos para a elaboração completa do projeto, consequentemente, o grupo optou por trocar de PIC e encontrar um substitutivo que apresentasse uma maior quantidade de pinos e que atendesse dois requisitos importantes: que fosse barato (preço do produto mais valor do frete) e disponibilidade para entrega dentro do prazo necessário para a montagem física do projeto, pois era preciso haver um tempo entre a montagem real e a apresentação, para corrigir eventuais erros ou imprevistos.

Após uma análise mais aprofundada, foi identificado o PIC18F4550 como uma opção adequada. Ele apresenta uma quantidade maior de pinos em comparação ao PIC16F628A e atende ao requisito de prazo de entrega rápido e custo-benefício. Essa escolha foi feita considerando a possibilidade de resolver eventuais problemas durante a montagem, caso surgissem, dentro do tempo disponível até a data de entrega do projeto.

Dessa forma, a troca para o microcontrolador PIC18F4550 foi fundamentada na necessidade de um maior número de pinos e nas vantagens que ele oferecia em termos de prazo e disponibilidade, o que nos proporcionaria mais segurança e flexibilidade na implementação do projeto. Além disso, a troca do microcontrolador proporcionou conhecer um novo microcontrolador, no qual não havia familiaridade e assim compreender algumas de suas diferenças e possibilidades em comparação aquele utilizado em aula.

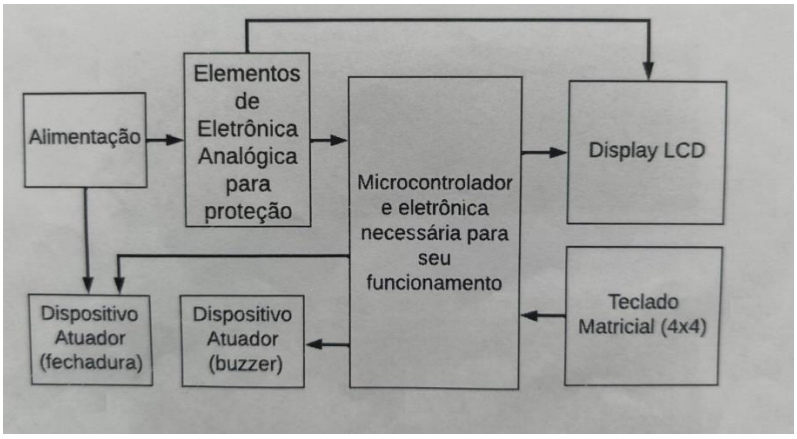
2 DESENVOLVIMENTO

Os requisitos se dividem em duas seções principais, Hardware e Software. Abaixo será abordada cada uma individualmente e também será apresentada mais sobre cada componente e seu funcionamento.

2.1 DESCRIÇÃO DO HARDWARE

Os componentes essenciais para a elaboração e o funcionamento do projeto são: uma fonte de alimentação, um microcontrolador PIC, um teclado matricial (4x4 ou 4x3), um display LCD 16x2, um buzzer, um sistema de fechadura (relé ou servomotor) e elementos de eletrônica analógica para proteção (como um regulador de tensão de 5V). O diagrama de bloco abaixo corresponde ao projeto de fechadura eletrônica.

Figura 1 - Diagrama de Blocos do Circuito de Fechadura Eletrônica



Fonte: Projeto Integrador - Fechadura Eletrônica baseada em microcontroladores PIC (2023, p. 1)

2.1.1 Componentes

Os componentes serão abordados brevemente nesta seção, além de apresentar as exigências de cada componente em relação ao projeto e figuras dos componentes no software Proteus 8 Professional e imagens reais dos mesmos. Outras componentes utilizadas que não necessitam de um tópico próprio, são: resistores (10kΩ, 1kΩ, 330Ω, 220Ω), capacitores (22pF), transistores, potenciômetro 22kΩ, Diodo, Diodo Zener, botão e Cristal Oscilador (8MHz).

2.1.1.1 PIC18F4550

O microcontrolador PIC18F4550 é um componente eletrônico utilizado em diversos projetos, incluindo sistemas de controle e automação, como uma fechadura eletrônica. Desenvolvido pela Microchip Technology, o PIC18F4550 é um microcontrolador de 8 bits que faz parte da família PIC18.

Ele possui uma arquitetura avançada e recursos que o tornam adequado para aplicações de médio a alto desempenho. O PIC18F4550 possui uma frequência de operação de até 48 MHz, o que o torna capaz de lidar com tarefas complexas e exigentes.

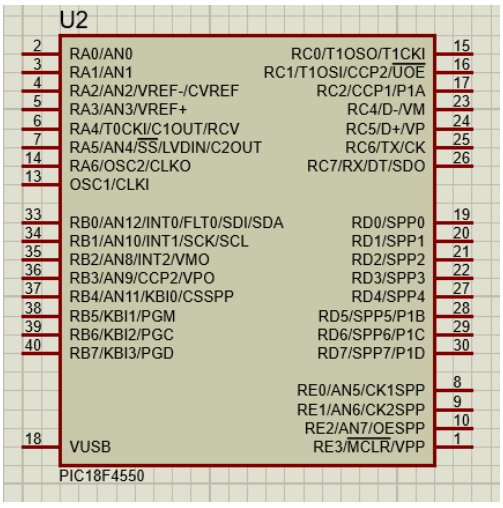
Além disso, o microcontrolador oferece uma ampla gama de recursos, como uma grande capacidade de memória, incluindo memória flash para armazenamento de código e memória RAM para armazenamento temporário de dados. Isso permite que programas complexos sejam executados e dados sejam processados de forma eficiente.

O PIC18F4550 também possui uma variedade de periféricos integrados, como conversores analógico-digital (ADC), comunicação USB, portas de entrada/saída (A, B, C, D, E), temporizadores, entre outros. Esses periféricos fornecem recursos essenciais para a interface com dispositivos externos, como o teclado matricial, display LCD, buzzer e relé, que podem ser usados em um sistema de fechadura eletrônica.

Figura 2 - PIC18F4550

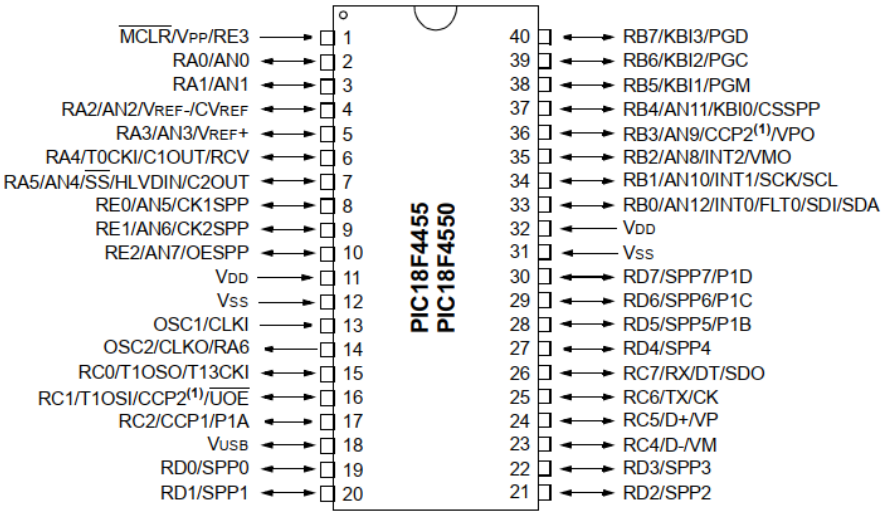
Fonte: Elaborada pelo autor

Figura 3 - PIC18F4550 no Proteus 8



Fonte: Captura de tela no software Proteus 8 Professional

Figura 4 - Diagrama de Pinos do PIC18F4455/PIC18F4550



Fonte: Captura de tela. Disponível em: Datasheet PIC18F2455/2550/4455/4550

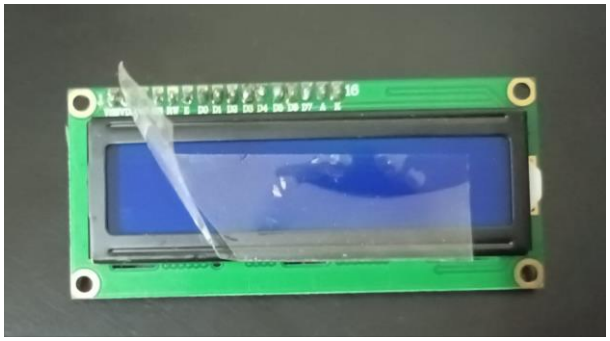
2.1.1.2 Display LCD 16x2

O display LCD (*Liquid Crystal Display*) é um componente amplamente utilizado em sistemas eletrônicos, incluindo fechaduras eletrônicas, devido à sua capacidade de fornecer informações visuais de forma clara e legível. O display LCD consiste em uma matriz de pixels, formada por células de cristal líquido, que podem ser controladas individualmente para exibir texto, números, símbolos ou até mesmo gráficos simples. Eles são compostos por camadas de material polarizador e filtros de cor, que permitem a passagem da luz em diferentes intensidades, dependendo do sinal elétrico aplicado.

Uma das vantagens do display LCD é sua baixa demanda de energia, o que o torna adequado para dispositivos alimentados por bateria, como uma fechadura eletrônica. Além disso, o display LCD oferece uma boa legibilidade em diferentes condições de iluminação, desde ambientes com luz ambiente até situações de baixa luminosidade.

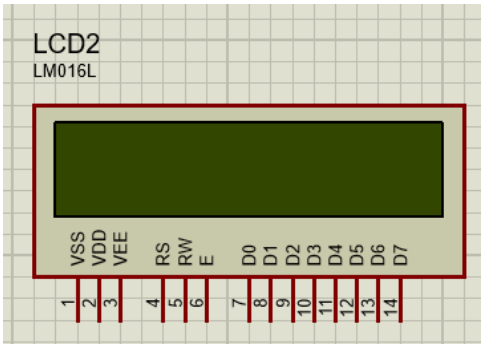
Os displays LCD utilizados em fechaduras eletrônicas são geralmente do tipo alfanumérico, como o display LCD 16x2, que possui 16 colunas e 2 linhas, permitindo a exibição de até 32 caracteres.

Figura 5 - Display LCD 16x2



Fonte: Elaborada pelo autor

Figura 6 - Display LCD 16x2 no Proteus 8



Fonte: Captura de tela no software Proteus 8 Professional

2.1.1.3 Teclado Matricial 4x4

Um teclado matricial 4x4 é um dispositivo de entrada que permite ao usuário inserir informações em sistemas eletrônicos, como uma fechadura eletrônica. Ele consiste em uma matriz de botões organizados em 4 linhas e 4 colunas, totalizando 16 teclas.

Cada botão do teclado matricial está associado a um valor específico, como um número, letra ou símbolo. Quando um botão é pressionado, um circuito é fechado entre a linha correspondente e a coluna correspondente, permitindo que o microcontrolador identifique qual botão foi pressionado.

O teclado matricial 4x4 é comumente utilizado em sistemas eletrônicos devido à sua simplicidade de uso e baixo custo. Ele oferece uma interface de entrada compacta, ideal para aplicações com restrições de espaço.

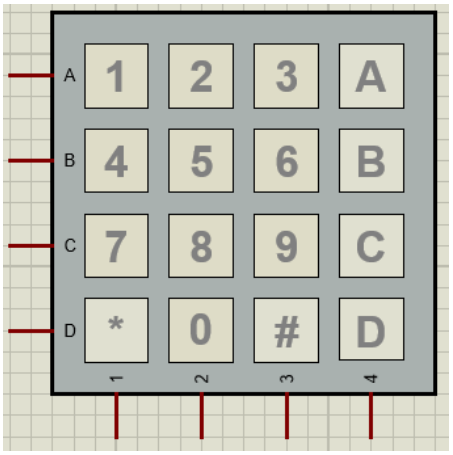
Neste projeto de uma fechadura eletrônica, o teclado matricial é utilizado para que o usuário possa inserir a senha de acesso. Ao pressionar as teclas correspondentes aos dígitos da senha, o microcontrolador pode ler os sinais gerados pelo teclado e processá-los para verificar se a senha está correta.

É relatado que o teclado matricial pode conter 12 ou 16 botões, utilizando o princípio da multiplexação ou o módulo pronto. Em vista disso, foi preferível utilizar o módulo pronto de um teclado matricial 4x4, por conta da praticidade na montagem (física e no Proteus 8 Professional) e por conta da compra dos componentes, para evitar gastas desnecessários com frete, foi utilizado uma mesma loja para a compra dos componentes e o teclado 4x4 estava disponível e o teclado 4x3 não.

Figura 7 - Teclado Matricial 4x4

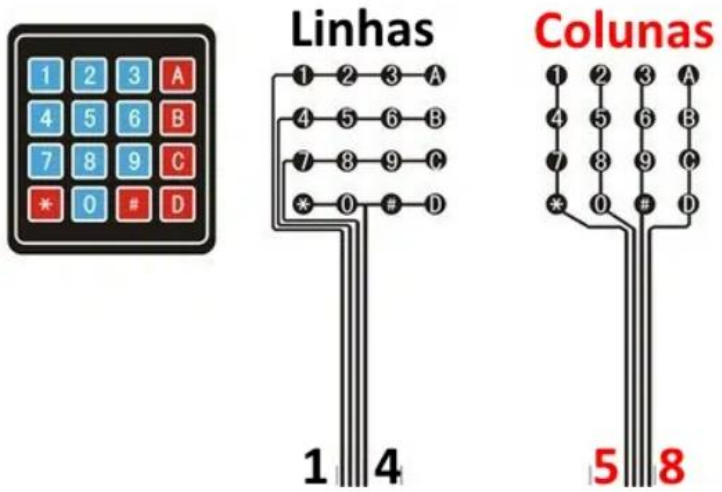
Fonte: Elaborada pelo autor

Figura 8 - Teclado Matricial 4x4 no Proteus 8



Fonte: Captura de tela no software Proteus 8 Professional

Figura 9 - Linhas referentes a cada coluna e linha



Fonte: Captura de tela no site MakerHero[1]

É fundamental saber qual linha/fio representa cada coluna e linha para que sejam realizadas as conexões corretas entre o teclado matricial e os pinos do microcontrolador, responsáveis por enviar e receber os dados.

[1] Disponível em: <https://www.makerhero.com/blog/teclado-matricial-4x4-arduino/>. Acesso em 17 de Junho de 2023.

2.1.1.4 Regulador de Tensão 5V

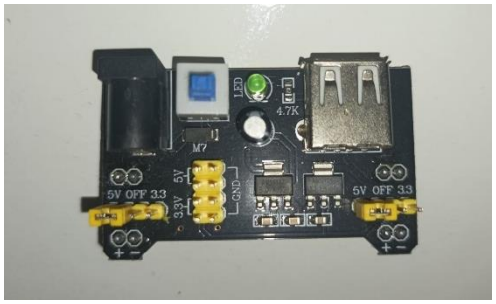
Um regulador de tensão de 5V é um dispositivo eletrônico utilizado para fornecer uma saída de tensão estável de 5V, independentemente das variações na tensão de entrada. Ele desempenha um papel essencial em muitos circuitos eletrônicos, garantindo que os componentes recebam a tensão adequada para operar corretamente.

O regulador de tensão de 5V é comumente utilizado em projetos eletrônicos para alimentar microcontroladores, circuitos integrados, sensores e outros dispositivos que requerem uma fonte de alimentação estável de 5V. Ele é projetado para lidar com variações na tensão de entrada, como flutuações na rede elétrica ou oscilações geradas por outros componentes do circuito.

Existem diferentes tipos de reguladores de tensão de 5V disponíveis, e são divididos em duas subcategorias: os reguladores por derivação (*shunt*) e os reguladores em série (*series*). Os reguladores de tensão em série (series) e os reguladores por derivação (shunt) são dois tipos comuns de circuitos reguladores de tensão. Os reguladores em série funcionam conectando um elemento regulador em série com a carga, mantendo uma tensão de saída estável independentemente de variações na tensão de entrada. Por outro lado, os reguladores por derivação são conectados em paralelo com a carga e desviam parte da corrente para fornecer uma tensão regulada à carga. Ambos os tipos são amplamente utilizados em aplicações eletrônicas e sistemas que requerem uma fonte de alimentação estável e confiável. A escolha entre eles depende das necessidades específicas do projeto, como a faixa de tensão de entrada, eficiência energética e requisitos de regulação de tensão.

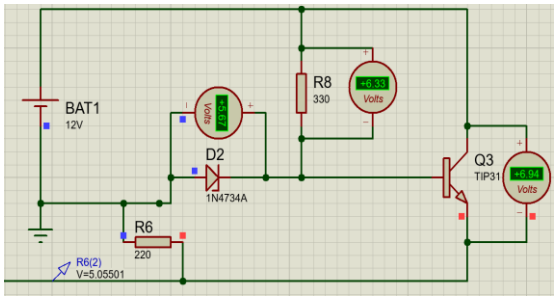
O regulador de tensão de 5V neste projeto de fechadura eletrônica, é responsável por fornecer uma alimentação estável de 5V para o microcontrolador, o display LCD, o teclado matricial e o buzzer. Isso garante que esses componentes funcionem corretamente e de forma confiável, independentemente de flutuações na tensão de entrada. No Proteus 8, foi elaborado um regulador de tensão em série, recebendo 12V na entrada e na saída para o microcontrolador, é fornecido uma tensão de 5V. Na montagem real da fechadura, foi utilizado a fonte ajustável para protoboard 3.3V e 5V.

Figura 10 - Fonte Ajustável para Protoboard



Fonte: Elaborada pelo autor

Figura 11 - Regulador de Tensão de 5V no Proteus



Fonte: Captura de tela no software Proteus 8 Professional

2.1.1.5 Dispositivo Atuador (Fechadura): Relé

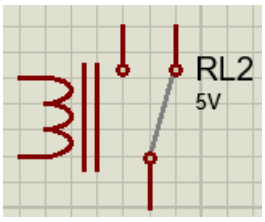
A escolha do dispositivo atuador (fechadura) foi o relé, em vez do servomotor, por conta de um dos integrantes já possuir o relé.

Um relé é um componente eletrônico que atua como um interruptor controlado por um sinal elétrico. Ele é composto por um eletroímã e um conjunto de contatos elétricos que abrem ou fecham um circuito quando o relé é acionado. Quando uma corrente elétrica passa pelo eletroímã do relé, ele cria um campo magnético que atrai ou repele um conjunto de contatos móveis. Esses contatos, por sua vez, estão conectados a outros circuitos elétricos. Quando o relé é acionado, os contatos podem ser movidos para abrir ou fechar o circuito desses outros circuitos, permitindo ou interrompendo o fluxo de corrente elétrica.

Figura 12 - Relé

Fonte: Elaborada pelo autor

Figura 13 - Relé no Proteus 8



Fonte: Captura de tela no software Proteus 8 Professiona

2.1.1.6 Buzzer

O buzzer é um componente eletrônico utilizado para produzir sons audíveis em circuitos eletrônicos. Ele consiste em um transdutor que converte um sinal elétrico em ondas sonoras. O buzzer pode ser classificado em dois tipos principais: ativo e passivo. O buzzer ativo possui um oscilador interno que produz o som quando é aplicada uma tensão contínua. Já o buzzer passivo não possui um oscilador interno e requer um sinal elétrico externo para produzir o som. O funcionamento básico de um buzzer é simples. Quando uma corrente elétrica é aplicada ao buzzer, ele gera vibrações mecânicas que são convertidas em ondas sonoras audíveis. A frequência do som produzido depende das características do buzzer e do sinal elétrico aplicado.

Neste projeto, caso a senha esteja correta, o buzzer deverá emitir dois bips curtos, em torno de 150 milissegundos, e caso a senha esteja incorreta, deverá emitir um bip longo, em torno de 500 milissegundos.

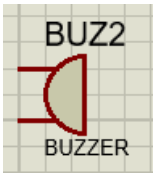
Figura 14 – Buzzer, parte frontal

Fonte: Elaborada pelo autor

Figura 15 - Buzzer, parte posterior

Fonte: Elaborada pelo autor

Figura 16 - Buzzer no Proteus 8



Fonte: Captura de tela no software Proteus 8 Professional

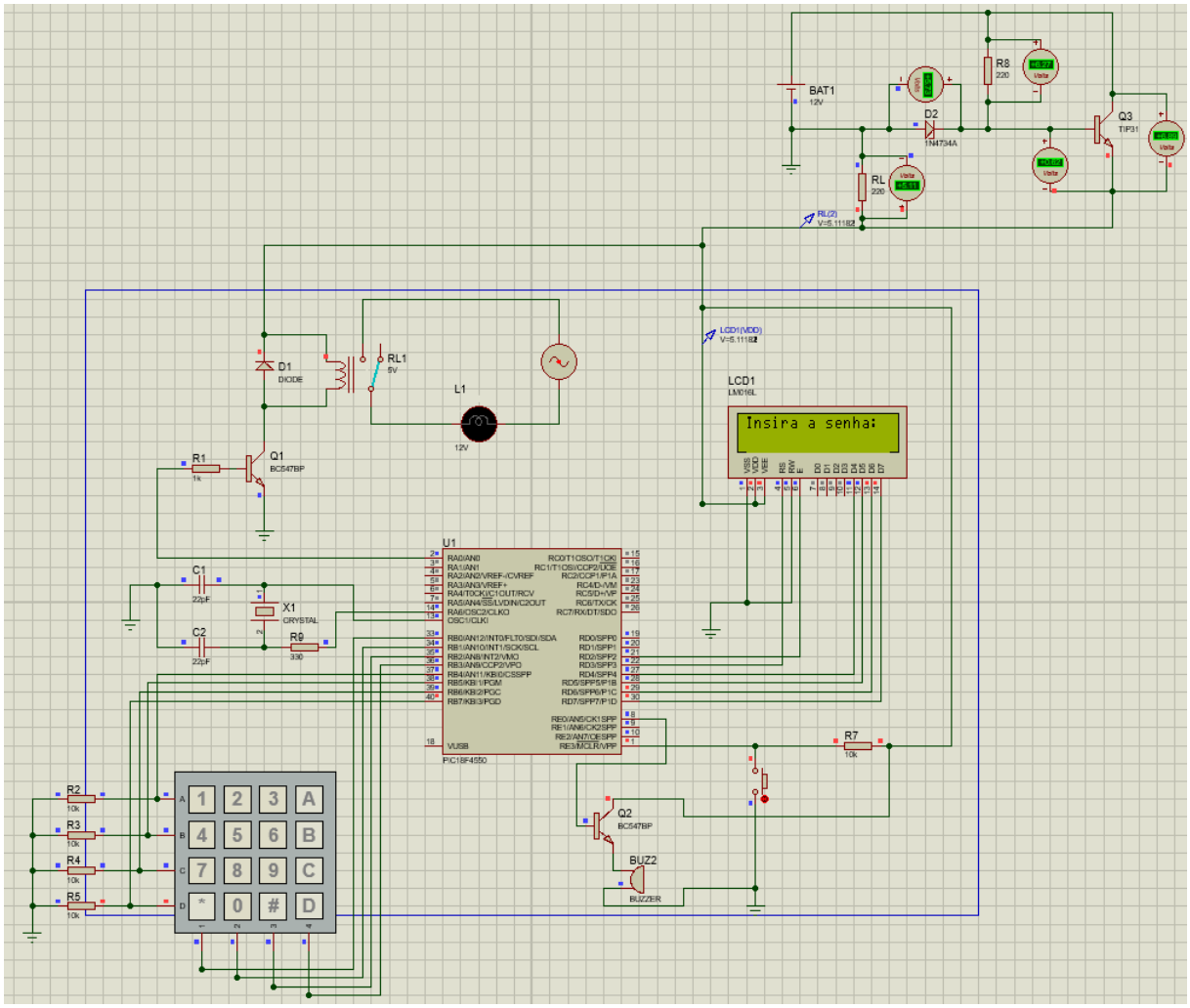
2.1.2 Montagem No Proteus 8 Professional

Foram realizadas duas montagens, uma “virtual” realizada no software Proteus 8 Professional e a outra, foi a elaboração real da fechadura. A montagem no Proteus 8 é fundamental pois é através dela que a montagem real será baseada, além disso, através do software é realizado as simulações e análises do circuito, permitindo identificar erros e problemas antes de ser realizado a montagem real da fechadura eletrônica.

Algumas diferenças ocorreram entre a montagem no Proteus 8 e a fechadura real, a principal é o regulador de tensão 5V, no Proteus 8 utilizou-se um regulador de tensão em série, enquanto na montagem real usou-se a fonte ajustável para protoboard 3.3V e 5V, isso ocorreu por estar elaborado o projeto na própria protoboard, seria mais prático usar a própria fonte ajustável vinda junta a protoboard em vez de construir uma própria.

Além disso, no Proteus 8, o microcontrolador não apresenta o V_{DD} e V_{SS}, então não havia como alimenta-lo, entretanto, outros componentes deveriam ser alimentados por uma tensão de 5V e com isso, optou por criar o regulador de tensão de 5V utilizando-se conceitos vistos em aula (como diodo Zener e transistores). Sendo assim, no software, este regulador de tensão em série foi responsável por alimentar o Display LCD, o relé, o buzzer e o MCLR. Abaixo se encontra o sistema completo da fechadura eletrônica criada no Proteus 8 Professional.

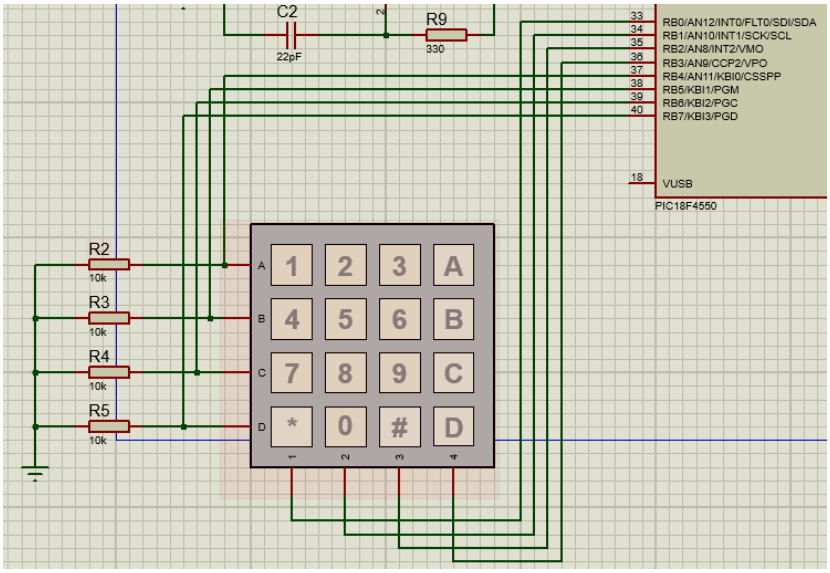
Figura 17 - Sistema de Fechadura Eletrônica no Proteus 8



Fonte: Captura de tela no software Proteus 8 Professional

O teclado foi conectado a PORTB do microcontrolador, as colunas (1, 2, 3, 4) são conectadas aos pinos RB0-RB3, respectivamente, e as linhas (A, B, C, D) são conectadas aos pinos RB4-RB7, respectivamente. As linhas recebem nível lógico ALTO alternadamente e as colunas estão nível lógico BAIXO quando nenhuma tecla está sendo pressionada. Quando o usuário pressiona uma tecla, é criado uma conexão entre a linha e a coluna, a partir disso, a coluna passa a ter nível lógico ALTO e o microcontrolador recebe este sinal, identificando qual a tecla foi pressionado. Além disso, são colocados resistores de *pull-down* no teclado para garantir que não haja leituras indesejadas ou inconsistentes, ou seja, eles garantem que a leitura permaneça em nível lógico BAIXO quando nenhum botão está sendo pressionado.

Figura 18 - Conexões entre PIC e Teclado



Fonte: Captura de tela no software Proteus 8 Professional

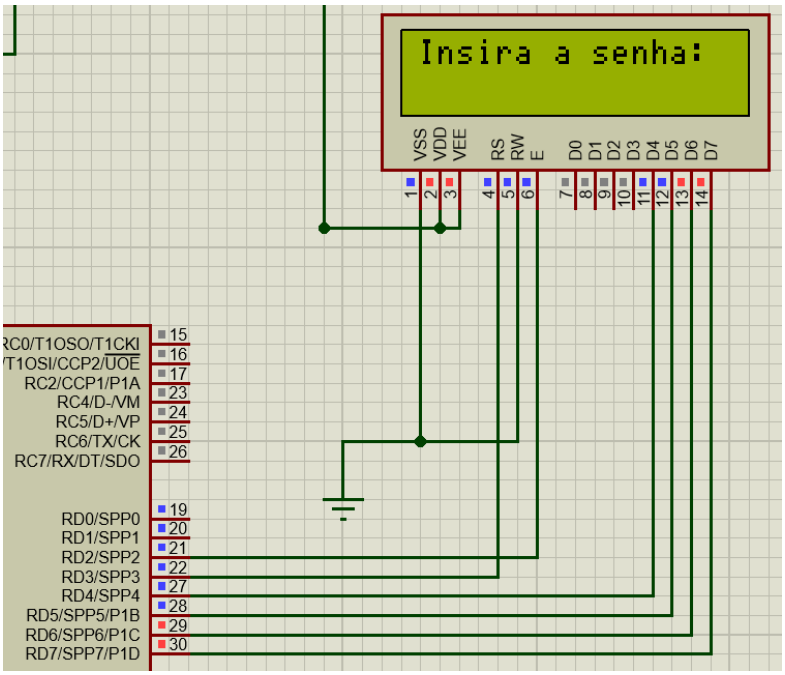
O display está conectado no PORTD, os pinos V_{DD} (responsável pela alimentação do display) está conectado ao regulador de tensão e o pino V_{EE} ou V_O está conectado a um potenciômetro que está conectado ao regulador de tensão, este pino é responsável pelo ajuste de contraste no display. E os pinos V_{SS} (o Terra do display) e RW estão aterrados. O pino RW é chamado de Read/Write e ele determina se o display atuara em modo de leitura (caso receba 1) ou envio de dados (caso receba 0), mas como o display receberá informações pelo microcontrolador para serem exibidas, ele precisa receber nível lógico BAIXO e por isso é aterrado.

O pino RS (*Register Select*) controla em qual parte do LCD está sendo enviando os dados, dentro do display LCD há um microcontrolador e o PIC18F4550 envia dados para este

microcontrolador do display, determinando que os dados são instruções ou escrita, através deste pino RS que está conectado ao RD3 do PIC18F4550.

Os pinos D0-D7 controlam o estado de entrada, ou seja, o envio dos dados que serão exibidos no display, entretanto, a preferência é utilizar apenas 4 bits (D4-D7). E o pino E (*enable*) é responsável por habilitar o envio de dados, ele está conectado ao pino RD2 do microcontrolador e os pinos D4-D7 do display estão conectados aos pinos RD4-RD7 respectivamente.

Figura 19 - Conexões entre PIC e Display LCD



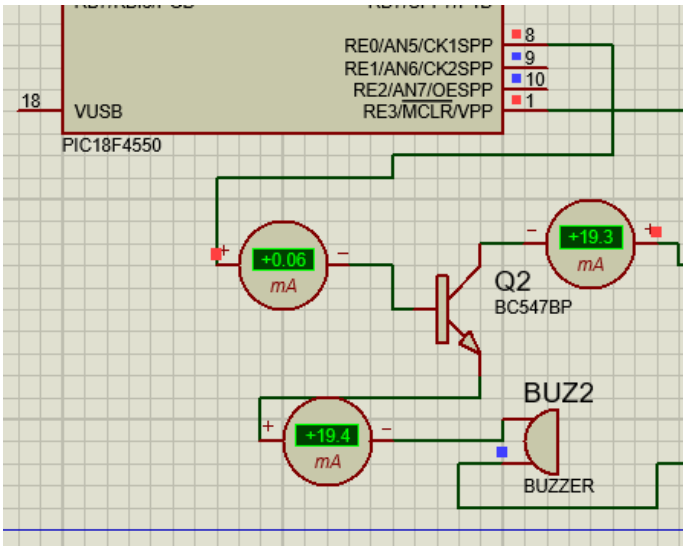
Fonte: Captura de tela no software Proteus 8 Professional

O buzzer está conectado ao emissor do transistor PNP BC547 e ao terra, o coletor do transistor está conectado ao regulador de tensão 5V e a base está conectada ao pino RE0. Ou seja, no momento em que RE0 recebe nível lógico ALTO, há passagem de corrente e tensão no transistor, consequentemente, o buzzer é ativado e emiti um som. Enquanto, RE0 for nível lógico BAIXO, não haverá passagem de corrente e de tensão no transistor, logo, o buzzer não será ativado.

Apesar do microcontrolador fornecer tensão necessária (em torno de 4-5V) para o buzzer, ele não fornece uma corrente suficiente para o buzzer, é oferecido aproximadamente 3mA. Por isso, é fundamental a presença do transistor, para atuar como um amplificador de corrente e assim, é fornecer uma corrente superior à corrente proporcionado pelo PIC, isso pode ser visto ao considerar a resistência de carga (220Ω) e a tensão (4-5V) do buzzer, sendo assim,

o buzzer consome uma corrente, entre 18mA e 22,7mA. Ao realizar a simulação foi encontrado uma corrente de 0,06mA vinda do PIC e da fonte 19,3mA, resultando em uma corrente no emissor (i_E) de 19,4mA, dentro da faixa prevista pela Lei de Ohm no buzzer. Sendo assim, o buzzer é ativado corretamente. Na montagem real da fechadura, foi identificado uma tensão de 4V e uma corrente de aproximadamente 27mA no buzzer (geralmente, a corrente máxima no buzzer é de 40mA), e utilizando a Lei de Ohm, foi constado uma resistência 150Ω através dos valores obtidos e da Lei de Ohm, estes valores.

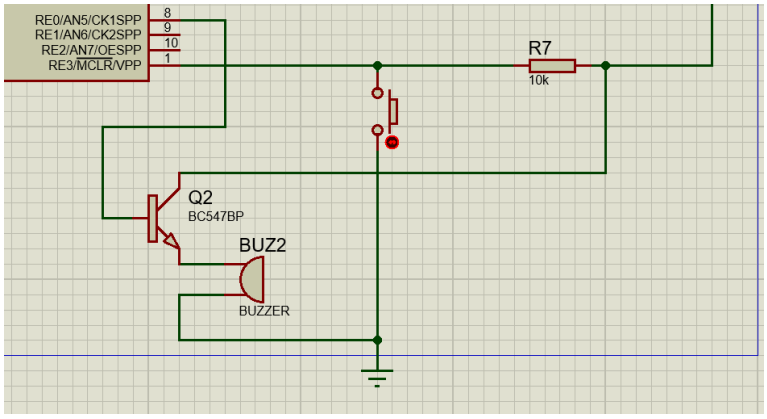
Figura 20 - Valores de Corrente no Transistor do Buzzer



Fonte: Captura de tela no software Proteus 8 Professional

O pino RE3/MCLR está conectado ao regulador de tensão 5V e enquanto este pino receber nível ALTO, o fuser MCLR não ativará e quando o botão conectado a este pino for pressionado (conectando o Terra), o pino RE3/MCLR receberá nível BAIXO e ativará o MCLR, resetando o microcontrolador.

Figura 21 - Conexão entre PIC, Buzzer e Alimentação do MCLR

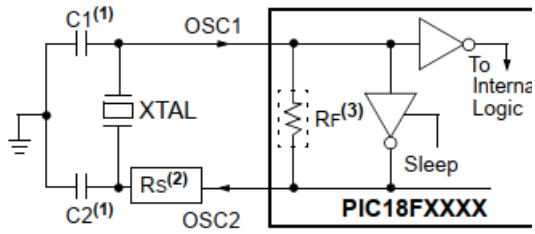


Fonte: Captura de tela no software Proteus 8 Professional

Uma diferença entre a montagem física da fechadura eletrônica e montagem do circuito no Proteus 8, é a presença do oscilador. No software, é possível o funcionamento do componente sem a presença de um oscilador, entretanto, na montagem real da fechadura, o componente é fundamental para o funcionamento para garantir a estabilidade e a precisão do sistema.

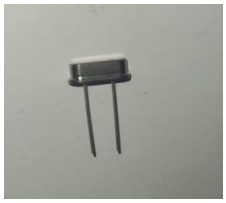
No caso do microcontrolador PIC18F4550, os pinos 14 (RA6/OSC2/CLKO) e 13 (OSC1/CLK1) são designados para o oscilador. Conforme indicado no datasheet, é necessário utilizar dois capacitores de 22pF, um cristal oscilador de 8MHz e um resistor de 330Ω conectados a esses pinos, seguindo as orientações da Figura 21.

Figura 22 – Configuração do Cristal Oscilador



Fonte: Datasheet PIC18F2455/2550/4455/4550

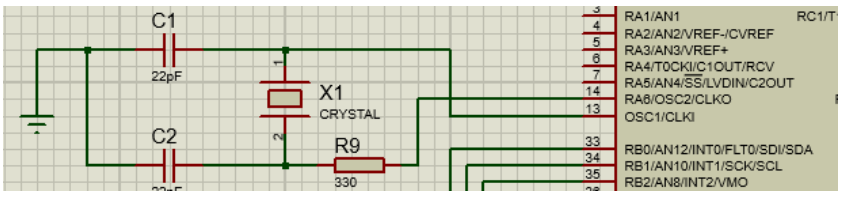
Figura 23 - Cristal Oscilador 8MHz



Fonte: Elaborada pelo autor

Para este projeto de fechadura eletrônica, uma faixa de frequência típica de 4MHz a 16MHz é o suficiente para oferecer um bom equilíbrio entre velocidade de processamento, eficiência energética e estabilidade. Por isso, escolheu-se um valor intermediário entre esta faixa de frequência, ou seja, optou por colocar a frequência de 8MHz e assim, o cristal oscilador 8MHz oferece um equilíbrio entre desempenho e consumo de energia, atendendo às necessidades de processamento rápido e eficiente do sistema. Além disso, o oscilador de cristal proporciona uma excelente imunidade a variações de temperatura e interferências elétricas, garantindo um funcionamento confiável e preciso da fechadura eletrônica em diferentes condições ambientais.

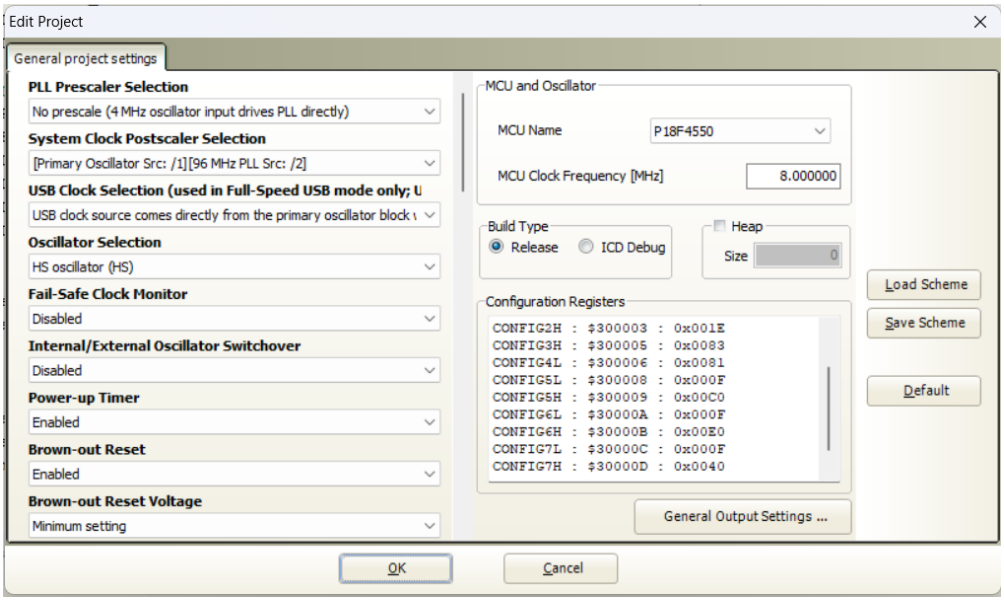
Figura 24 - Conexão entre PIC e Cristal Oscilador



Fonte: Captura de tela no software Proteus 8 Professional

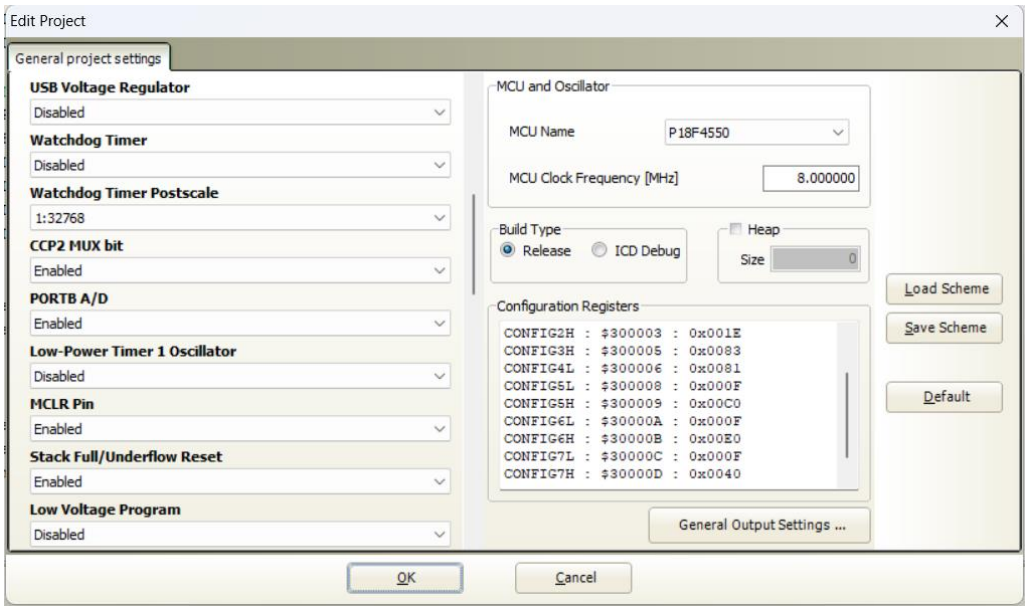
É fundamental fazer uma verificação dos fuses para identificar se algum deles poderia gerar alguma adversidade ou utilidade durante a elaboração do projeto, entretanto, como a configuração vinda durante a criação do projeto era o suficiente para garantir um bom funcionamento do microcontrolador PIC18F4550 para a fechadura eletrônica elaborada. Apenas o fuses de oscilador chegou a ser alterado, para HS oscillator (HS) com frequência de 8MHz, o restante dos fuses se manteve na configuração "de fábrica", como o MCLR habilitado etc.

Figura 25 - Configuração dos Fuses - Parte I



Fonte: Captura de tela no software MikroC Pro for PIC

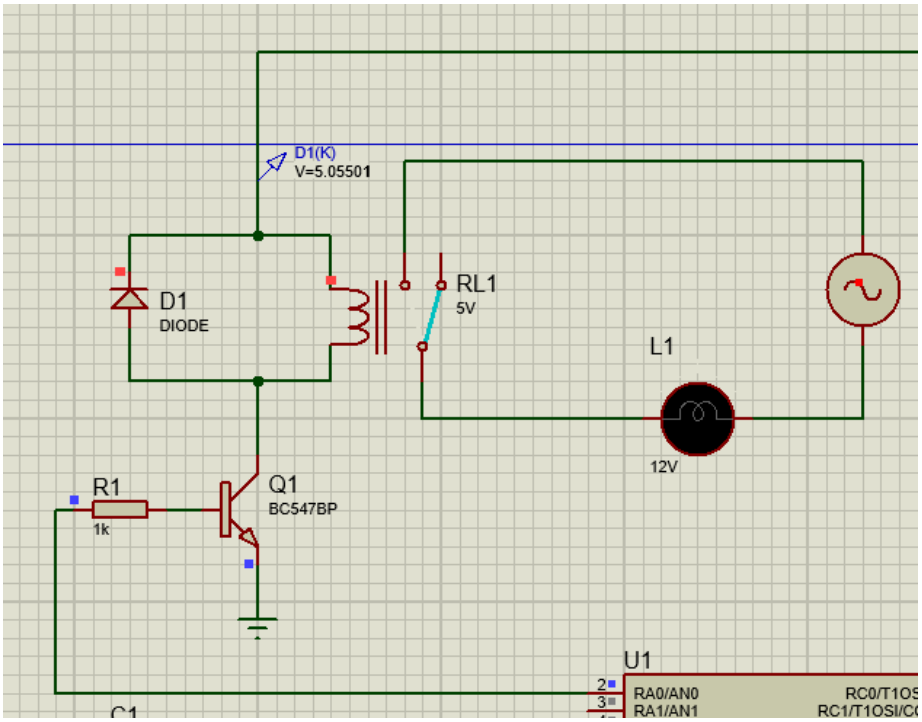
Figura 26 - Configuração dos Fuses - Parte II



Fonte: Captura de tela no software MikroC Pro for PIC

O pino RA0 do microcontrolador é responsável por enviar um sinal ao circuito do dispositivo atuador (fechadura), ou seja, no momento em que este pino envio um nível logico ALTO para a fechadura, ela é destravada e quando o dispositivo passa a receber nível lógico BAIXO, a fechadura é travada.

Figura 27 - Conexão entre o PIC e o Dispositivo Atuador (fechadura/relé)



Fonte: Captura de tela no software Proteus 8 Professional

Primeiro é preciso analisar o circuito, o relé é um dispositivo que quando recebe uma corrente em sua bobina (vinda da fonte de alimentação), essa corrente gera um campo magnético que atrai a chave.

No caso da simulação no Proteus 8, a chave está em uma posição onde se encontra um circuito aberto e no momento em que ela é atraída, mudando de posição, ela fecha um circuito que acende uma lâmpada, mas na prática, em vez de uma lâmpada, haverá uma fechadura que será aberta. Todavia, para a corrente passar pelo relé (bobina) é preciso que haja um caminho para ela e este caminho é fornecido pelo transistor, que atua como uma chave para acionar o relé, ou seja, quando o transistor recebe nível lógico alto na base, ele permite a passagem de corrente do coletor (vinda do relé) para o emissor (indo para o GND), assim, o relé gera um campo magnético que atrai a chave e destrava a fechadura.

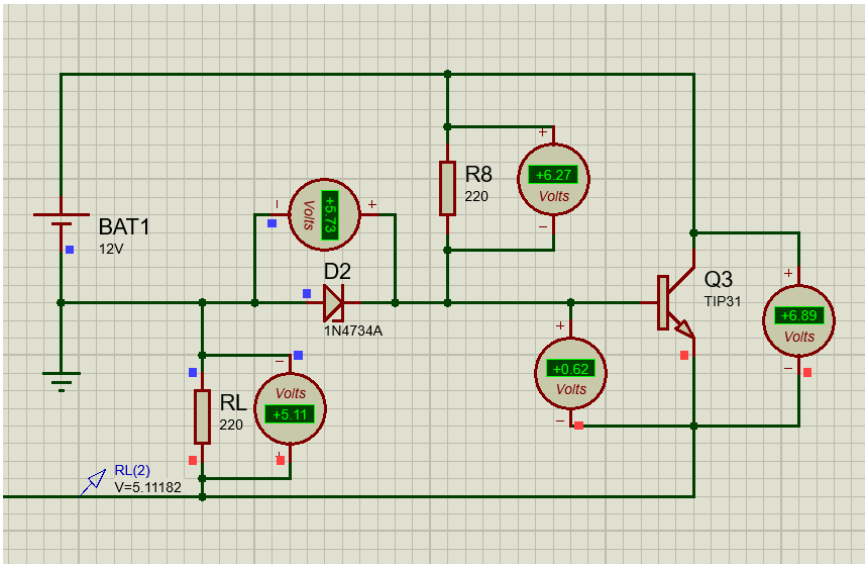
Quando a base do transistor recebe nível lógico baixo, o mesmo não permite a passagem de corrente entre o coletor e o emissor, assim, não há caminho para a corrente e

consequentemente, não há geração de campo magnético na bobina por não haver corrente na bobina.

O diodo que está em paralelo ao relé, tem como objetivo proteger o transistor contra a corrente da bobina, isto é, quando o transistor está ativado e há corrente passando pela bobina e o transistor muda, parando de conduzir, resta uma corrente residual na bobina e a tendência da bobina é forçar esta corrente para o transistor, mas ao inserir um diodo em paralelo, em vez de forçar esta corrente no transistor, a corrente residual circula pelo diodo e se gasta na própria bobina, por isso, o diodo é visto como proteção ao transistor.

Este circuito é utilizado para alimentar e ativar o relé, pois o dispositivo relé não pode ser alimentado e ativado ao mesmo tempo apenas pelo microcontrolador, já que o PIC não tem a capacidade de fornecer uma alta corrente para alimentar o relé e consequentemente gerar um campo magnético que atraia a chave, isso ocasionaria em uma sobrecarga na saída do PIC, por isso utiliza-se este circuito, conhecido como “driver de relé”. É possível definir o “driver de relé” como um circuito utilizado para controlar o acionamento de um relé através de um sinal de baixa potência, composto por um transistor, um diodo de proteção e um resistor. O transistor atua como um interruptor controlado pelo sinal de entrada, permitindo ou interrompendo o fluxo de corrente através da bobina do relé. O diodo de proteção em paralelo à bobina do relé proteger o transistor contra picos de tensão reversa gerados pela desenergização da bobina. Isso evita danos ao transistor e melhora a confiabilidade do circuito. E o resistor é utilizado para limitar a corrente de base do transistor, garantindo um acionamento adequado.

Figura 28 - Regulador de Tensão em Série 5V



Fonte: Captura de tela no software Proteus 8 Professional

A figura acima, demonstra um circuito que representa um regulador de tensão em série de 5V, este regulador está conectado a uma bateria de 12V e a tensão de saída do regulador é conectada ao dispositivo atuador (relé/fechadura), ao pino MCLR, ao buzzer e ao display LCD, além disso, no Proteus 8 Professional o microcontrolador não apresenta um V_{DD} e um V_{SS}, mas na prática, estes pinos do microcontrolador também estariam conectados a tensão de saída do regulador. Como a fonte de alimentação utilizada para fechadura real não ultrapassará 12V, optou-se pela elaboração pelo modelo mais simples de regulador de tensão para a fechadura no Proteus 8.

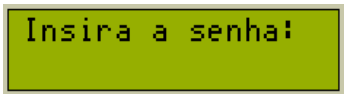
Neste circuito, o diodo Zener tem como função estabelecer uma tensão de referência na base do transistor, assim, o transistor conduz de tal forma a manter constante a tensão no emissor, ou seja, o transistor regula a corrente para a carga mantendo a tensão de saída constante (que é conectada ao display LCD, microcontrolador e ao relé) e diodo Zener polarizado pela resistência R8 (220Ω) estabelece um valor de referência para o regulador. A tensão de saída é dada por: $V_o = V_{Zener} + V_{BE}$. Assim, se V_{Zener} é constante e a tensão de saída se altera, V_{BE} se altera proporcionalmente diminuindo ou aumentando a corrente de saída para restabelecer a tensão de saída, V_o. Ou, se a tensão de entrada (bateria/fonte) for alterada, a tensão de referência V_{Zener} permanece quase inalterada, mantendo a corrente inicial e, portanto, a tensão de saída V_o constante. Vale mencionar que os componentes deste circuito devem dissipar as potências.

2.2 DESCRIÇÃO DO SOFTWARE

Em relação a parte do software, é estabelecido uma série de comandos a serem seguidos, que serão abordados abaixo. Entretanto, antes de aborda-los, é preciso mencionar que a senha possui algumas regras como: conter 4 dígitos no mínimo, ser *hard-coded* como variável ou programável pelo teclado matricial e armazenada na EEPROM.

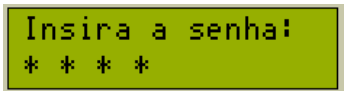
Com relação ao roteiro de funcionamento, assim que for ativado o código, o display tem que apresentar na primeira linha “Insira a senha:” (Fig. 31) e na segunda linha deve aparecer “*” quando o usuário digitar a senha (Fig. 32), além disso, é preciso que haja um espaço de distância entre cada dígito da senha. Após digitar a senha, deve haver um botão de confirmação, e ao pressionar o display deve ser limpo e apresentar a mensagem “Verificando...” na primeira linha pelo tempo de um segundo (Fig. 33).

Figura 29 - Tela Inicial no Display LCD



Fonte: Captura de tela no software Proteus 8 Professional

Figura 30 - Tela com Senha Digitada



Fonte: Captura de tela no software Proteus 8 Professional

Figura 31 - Tela de Verificação de Senha

Fonte: Captura de tela no software Proteus 8 Professional

Depois, o display deve exibir na segunda linha se a senha está correta (Fig. 34) ou incorreta (Fig. 36), por um intervalo de dois segundos. Caso a senha esteja correta, o display irá exibir “Liberado” e assim permanecer (Fig. 35). Caso a senha esteja incorreta, o display irá exibir o que exibe no começo do código, ou seja, o código retorna ao início.

Figura 32 - Tela com Senha Correta

Fonte: Captura de tela no software Proteus 8 Professional

Figura 33 - Tela de Sistema Liberado

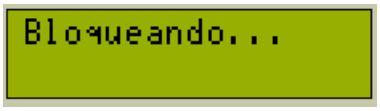
Fonte: Captura de tela no software Proteus 8 Professional

Figura 34 - Tela com Senha Incorreta

Fonte: Captura de tela no software Proteus 8 Professional

Além disso, deve haver um botão de bloqueio de sistema, que tem como função retornar ao estado original do atuador da fechadura (travada) e exibir o texto “Bloqueando...” por um segundo (Fig. 37) e depois retornar ao início do código, ou seja, pedir a senha novamente e exibir “Insira a senha: ” no display (Fig. 31).

Figura 35 - Tela de Bloqueio de Sistema



Fonte: Captura de tela no software Proteus 8 Professional

2.2.1 Arquitetura do Software

A arquitetura de software desenvolvida para a fechadura eletrônica baseada no microcontrolador PIC18F4550 foi projetada para garantir um bom funcionamento do sistema, evidentemente, ele poderia seguir outros caminhos e há possibilidade de diversos aperfeiçoamentos, contudo, optou-se por mantê-lo desta forma, pois, a partir do momento em que se alcançou o objetivo estabelecido pelo roteiro de projeto, não era essencial buscar um a

aperfeiçoamento do código. O código-fonte do software é composto por diversas funções interconectadas que desempenham papéis específicos na operação da fechadura.

O código foi produzindo no software MikroC PRO for PIC, na Linguagem de Programação C e utilizou-se as bibliotecas: Keypad4x4, Lcd, Lcd_Constants disponíveis no software. O código-fonte pode ser visto nas figuras a seguir:

Figura 36 - Variáveis e Configuração do Display LCD

```
// CONFIGURAÇÃO DO PORT
sbit LCD_RS at RD3_bit;
sbit LCD_EN at RD2_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

// CONFIGURAÇÃO DO TRIST
sbit LCD_RS_Direction at TRISD3_bit;
sbit LCD_EN_Direction at TRISD2_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;


char keypadPort at PORTB;                       // Teclado é configurado para o PORTB
char teclas[4][4] = {{'1', '2', '3', 'A'},
                     {'4', '5', '6', 'B'},
                     {'7', '8', '9', 'C'},
                     {'*', '0', '#', 'd'}};
char senhadigitada[4];                          // Receberá 4 valores da matriz teclas
char senhamestre[4] = {'1', '2', '3', '4'};
int i=0, m=0, kp, cfm = 0, x;

/*
i = usado na hora de "pegar" o valor selecionado no teclado e no preenchimento do display
m = usado para avaliar se a senha está correta ou incorreta
kp = usado para leitura do teclado
cfm = usado para ativar a função de COMPARAR SENHA
x = representa a posição de cada CHAR e comparar as mesmas posições          */
```

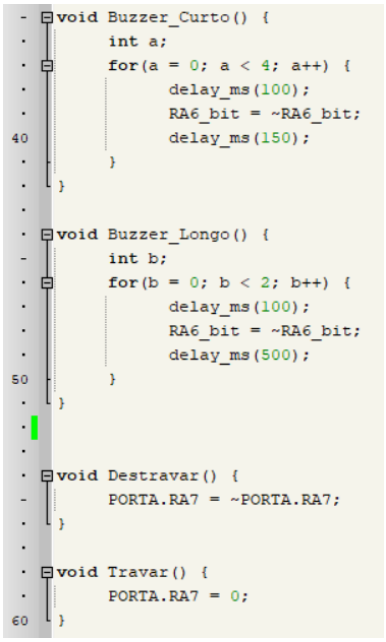
Fonte: Captura de tela no software MikroC PRO For PIC

Nesta primeira parte, utiliza-se a biblioteca Lcd para a configuração do PORTD e TRISD já que o display LCD está conectado aos pinos D, e como o teclado está conectado aos pinos B, utiliza-se a biblioteca Keypad4x4 para configurar o teclado matricial ao PORTB. Depois é criado uma matriz bidimensional (chamada teclas) de 4x4 para armazenar os valores possíveis, que representam cada valor presente no teclado 4x4. Outras duas variáveis são criadas, uma chamada de “senhadigitada”, que receberá os valores digitados no teclado pelo usuário e a outra variável é uma senha *hard-code*, nomeada de “senhamestre”, ela contém os seguintes dados tipo char ‘1’, ‘2’, ‘3’ e ‘4’. Lembre-se que todos os dados presentes nessas variáveis são do tipo char e não do tipo int, isso foi realizada da seguinte forma para tornar fácil a comparação entre cada valor, pois não se obteve sucesso ao tentar utilizar a função de comparação entre char, *strcmp()*.

Outras variáveis criadas, são do tipo int, nomeadas de ‘i’, ‘m’, ‘kp’, ‘cfm’ e ‘x’. Cada uma delas será utilizada para alguma funcionalidade específica dentro do código, assim será possível compreender melhor a funcionalidade de cada uma, no momento em que elas aparecerem, mas será abordado brevemente sobre cada uma a seguir:

- a variável ‘i’ é o índice usado para identificar a posição em que será armazenada o valor digitado pelo usuário;
- a variável ‘m’ é incrementada sempre que as mesmas posições nas variáveis “senhamestre[]” e “senhadigitada[]” possuírem os mesmos valores, ela serve para identificar se senha está correta ou incorreta;
- a variável ‘kp’ é utilizada para armazenar o valor obtido no momento em que alguma tecla é pressionada e esse valor é usado na declaração switch;
- a variável ‘cfm’ é usada para garantir que uma operação de confirmação de senha ocorrerá no momento em que o botão de confirmação (*) for pressionado;
- a variável ‘x’ é utilizada em um loop for para realizar complementar a operação de confirmar da senha.

Figura 37 - Funções do Buzzer e do Relé/Fechadura



```
void Buzzer_Curto() {
     int a;
     for(a = 0; a < 4; a++) {
          delay_ms(100);
          RA6_bit = ~RA6_bit;
          delay_ms(150);
     }
}

void Buzzer_Longo() {
     int b;
     for(b = 0; b < 2; b++) {
          delay_ms(100);
          RA6_bit = ~RA6_bit;
          delay_ms(500);
     }
}

void Destravar() {
     PORTA.RA7 = ~PORTA.RA7;
}

void Travar() {
     PORTA.RA7 = 0;
}
```

Fonte: Captura de tela no software MikroC PRO For PIC

Essa parte do código envolvem a criação de 4 funções importantes, as duas primeiras envolvendo a emissão de som pelo buzzer e as outras duas envolvendo o dispositivo atuador por relé (fechadura).

A primeira função “Buzzer_Curto()” controla o buzzer para gerar um som curto. Ela utiliza um loop for para repetir a sequência de operações por 4 vezes. Dentro do loop, é utilizado um atraso de 100ms, em seguida, o estado do pino RE0 é invertido usando o operador ~, o que faz com que o buzzer emita um som. Após o segundo atraso de 150ms, o loop continua até que a condição “a < 4” seja falsa. Sendo assim, isso alternada o bit de RE0 de 0 para 1, de 1 para 0, de 0 para 1 e 1 para 1, sendo assim, o buzzer será ativado 2 vezes com um som de duração de 150ms.

A segunda função “Buzzer_Longo()” controla o buzzer para gerar um som longo. Ela utiliza um loop for semelhante ao anterior, mas com apenas 2 iterações. Após cada atraso de 100ms, o estado do pino RE0 é invertido, o que faz com que o buzzer emita um som. Em seguida, há um atraso de 500ms antes de continuar para a próxima iteração. Assim como na função anterior, o loop é executado até que a condição “b < 2” seja falsa. Sendo assim, o buzzer emitirá som por 500ms, uma única vez.

A terceira e a quarta função são chamadas de “Destravar()” e “Travar()”, respectivamente, elas utilizam a mesma lógica. A primeira inverte o bit do pino RA7, assim, ele muda de 0 para 1, isso ativa o relé, ou seja, destrava a porta. E a função “Travar” transforma o RA7 em saída de nível lógico baixo (0), desativando o relé, sendo assim, a porta é travada.

Figura 38 - Função de Verificação de Senha

```
void confirmar(char senha1[], char senha2[]) {
    for(x = 0; x < 4; x++) {
            if(senha1[x] == senha2[x]) { ...
    }
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Out_CP("Verificando...");
    delay_ms(1000);

    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Cmd(_LCD_SECOND_ROW);

    if(m == 4) {
            delay_ms(200);
            Lcd_Out_CP("Senha correta!");
            delay_ms(2000);
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Cmd(_LCD_SECOND_ROW);
            Lcd_Out_CP("Liberado");
            Buzzer_Curto();
            Destravar();
            m = 0;
    } else {
            delay_ms(200);
            Buzzer_Longo();
            Lcd_Out_CP("Senha incorreta!");
            delay_ms(2000);
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out_CP("Insira a senha: ");
            Lcd_Cmd(_LCD_SECOND_ROW);
            i = 0;
            m = 0;
    }
}
```

Fonte: Captura de tela no software MikroC PRO For PIC

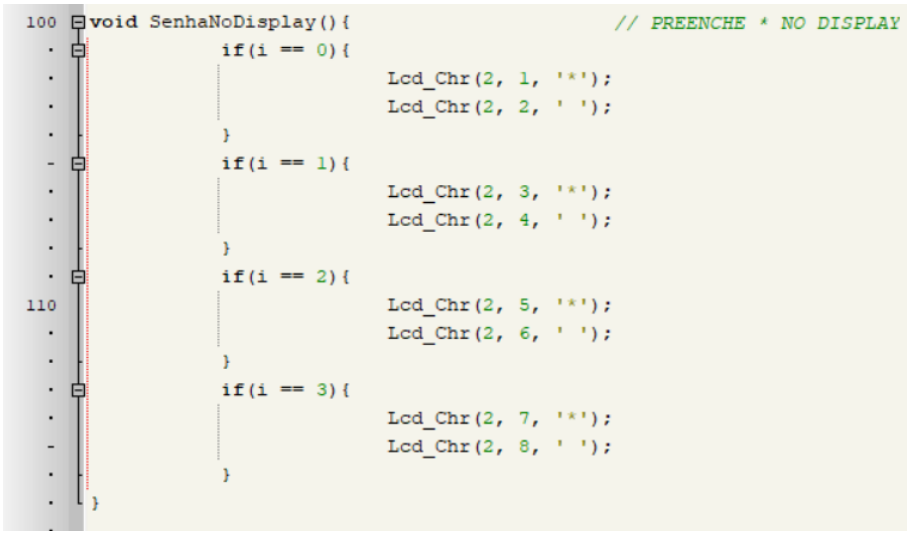
A função “confirmar()” recebe dois parâmetros: senha1, que é um array de caracteres representando a senha digitada pelo usuário, e senha2, que é um array de caracteres representando a senha mestre predefinida. A função começa com um loop for que percorre cada posição dos arrays senha1 e senha2. Para cada posição, verifica-se se os caracteres são iguais. Caso sejam iguais, a variável ‘m’ é incrementada.

Em seguida, o código realiza a exibição de mensagens no display LCD. Primeiro, limpa o display e desativa o cursor. Em seguida, exibe a mensagem "Verificando..." por um período de 1 segundo. Após isso, limpa o display novamente, desativa o cursor e move o cursor para a segunda linha do display. Em seguida, verifica-se se a variável ‘m’ é igual a 4. Se isso for verdadeiro, significa que todas as posições das senhas coincidem, indicando que a senha digitada está correta.

Dentro do bloco if, exibe a mensagem "Senha correta!" no display por período de 2 segundos, após isso, o buzzer emite um som através da função Buzzer_Curto() e depois ocorre o destravamento da fechadura utilizando a função Destravar(), e no final é redefinido a variável ‘m’ para o valor inicial, 0.

Caso a condição if não seja satisfeita, ou seja, a senha digitada não coincide com a senha mestre, o código exibe a mensagem "Senha incorreta!" no display por 2 segundos e emite um som longo do buzzer utilizando a função Buzzer_Longo(), o display é limpo novamente e exibe a mensagem "Insira a senha: " na primeira linha do display, movendo o cursor para a segunda linha. E também é redefinido as variáveis ‘i’ e ‘m’ para os valores iniciais, 0.

Figura 39 - Função de Preenchimento de Senha no Display LCD



```
100 void SenhaNoDisplay(){                    // PREENCHE * NO DISPLAY
            if(i == 0){
                        Lcd_Chr(2, 1, '*');
                        Lcd_Chr(2, 2, ' ');
            }
            if(i == 1){
                        Lcd_Chr(2, 3, '*');
                        Lcd_Chr(2, 4, ' ');
            }
            if(i == 2){
110                     Lcd_Chr(2, 5, '*');
                        Lcd_Chr(2, 6, ' ');
            }
            if(i == 3){
                        Lcd_Chr(2, 7, '*');
                        Lcd_Chr(2, 8, ' ');
            }
  }
```

Fonte: Captura de tela no software MikroC PRO For PIC

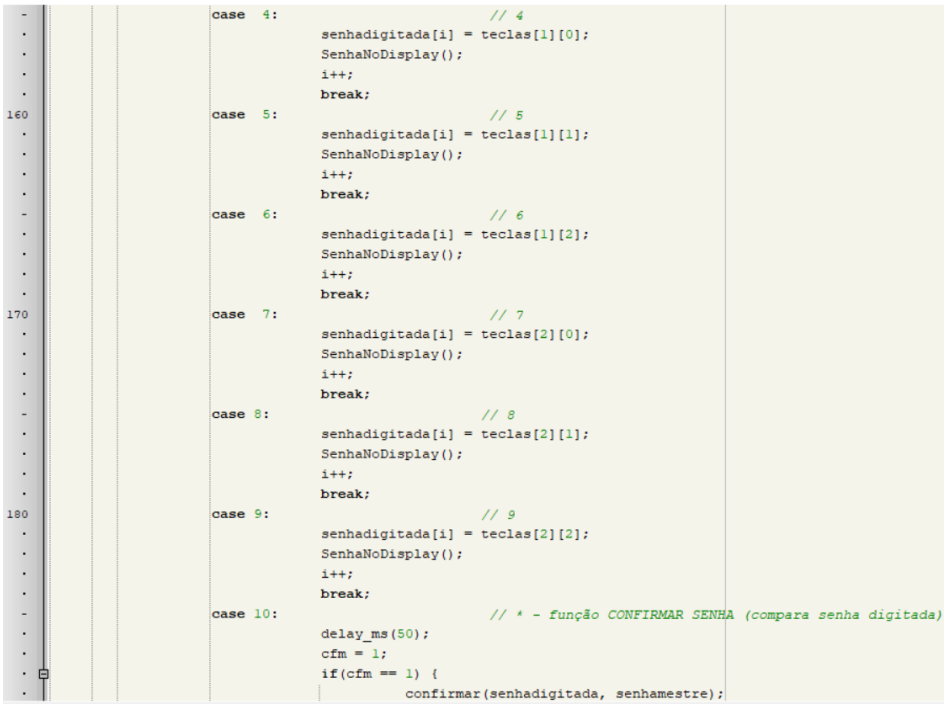
A função "SenhaNoDisplay" preenche o display LCD com asteriscos (*) para representar a quantidade de dígitos da senha digitada até o momento. A função verifica o valor da variável i, que indica a quantidade de dígitos digitados até o momento, e utiliza a função Lcd_Chr para exibir asteriscos nas posições correspondentes do display. Se ‘’i for igual a 0, os asteriscos são exibidos nas posições 2 e 3 do segundo linha do display. Se ‘i’ for igual a 1, os asteriscos são exibidos nas posições 3 e 4. Se ‘i’ for igual a 2, os asteriscos são exibidos nas posições 5 e 6. E se ‘i’ for igual a 3, os asteriscos são exibidos nas posições 7 e 8.

Figura 40 - Função de Leitura de Teclado – Parte I

```
120 void DigitarSenha() {
        Keypad_Init();
        Lcd_Init();
        Lcd_Cmd(_LCD_CLEAR);


        do{
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out_CP("Insira a senha: ");
        Lcd_Cmd(_LCD_SECOND_ROW);
130
           do{
                  kp = 0;
                  cfm = 0;

                  do{
                     kp = Keypad_Key_Click();
                  } while(!kp);

                  switch(kp) {
140                   case  1:                          // 1
                                 senhadigitada[i] = teclas[0][0];
                                 SenhaNoDisplay();
                                 i++;
                                 break;
                        case  2:                          // 2
                                 senhadigitada[i] = teclas[0][1];
                                 SenhaNoDisplay();
                                 i++;
                                 break;
150                   case  3:                          // 3
                                 senhadigitada[i] = teclas[0][2];
                                 SenhaNoDisplay();
                                 i++;
                                 break;
```

Fonte: Captura de tela no software MikroC PRO For PIC

Figura 41 - Função de Leitura de Teclado - Parte II



```
                case  4:                        // 4
                      senhadigitada[i] = teclas[1][0];
                      SenhaNoDisplay();
                      i++;
                      break;
                case  5:                        // 5
                      senhadigitada[i] = teclas[1][1];
                      SenhaNoDisplay();
                      i++;
                      break;
                case  6:                        // 6
                      senhadigitada[i] = teclas[1][2];
                      SenhaNoDisplay();
                      i++;
                      break;
                case  7:                        // 7
                      senhadigitada[i] = teclas[2][0];
                      SenhaNoDisplay();
                      i++;
                      break;
                case 8:                        // 8
                      senhadigitada[i] = teclas[2][1];
                      SenhaNoDisplay();
                      i++;
                      break;
                case 9:                        // 9
                      senhadigitada[i] = teclas[2][2];
                      SenhaNoDisplay();
                      i++;
                      break;
                case 10:                        // * - função CONFIRMAR SENHA (compara senha digitada)
                      delay_ms(50);
                      cfm = 1;
                      if(cfm == 1) {
                               confirmar(senhadigitada, senhamestre);
```

Fonte: Captura de tela no software MikroC PRO For PIC

Figura 42 - Função de Leitura de Teclado - Parte III

```
                case 10:                        // * - função CONFIRMAR SENHA (compara senha digitada)
                      delay_ms(50);
                      cfm = 1;
                      if(cfm == 1) {
                               confirmar(senhadigitada, senhamestre);
                      }
                      break;
                case 11:                         // 0
                      senhadigitada[i] = teclas[4][1];
                      SenhaNoDisplay();
                      i++;
                      break;
                case 12:                         // # - função BLOQUEAR (retornar ao estágio inicial)

                      Lcd_Cmd(_LCD_CLEAR);
                      Lcd_Cmd(_LCD_CURSOR_OFF);
                      Lcd_Out_CP("Bloqueando...");
                      delay_ms(1000);
                      Travar();
                      Lcd_Cmd(_LCD_CLEAR);
                      Lcd_Out_CP("Insira a senha: ");
                      Lcd_Cmd(_LCD_SECOND_ROW);
                      i = 0;
                      break;
                }
        } while(1);
    } while(1);
}
```

Fonte: Captura de tela no software MikroC PRO For PIC

A função "DigitarSenha()" é a função responsável por quase toda a lógica do software, pois ela utiliza todos as funções anteriores e o restantes das operações necessárias para o funcionamento da fechadura. De certa forma, a função captura e processa a entrada da senha

pelo teclado matricial e exibe a senha digitada no display LCD. As seguintes funções são obtidas através das bibliotecas:

- **Keypad_Init();**
  Inicializa o teclado matricial, configurando as portas e os pinos necessários para a leitura das teclas.
- **Lcd_Init();**
  Inicializa o display LCD, configurando as portas e os pinos necessários para a comunicação com o display.
- **Lcd_Cmd(_LCD_CLEAR);**
  Limpa o display LCD.

O código entra em um loop "do-while" externo que será executado continuamente. Dentro do loop externo, o código exibe a mensagem "Insira a senha:" no display LCD na primeira linha e desativa o cursor. Em seguida, inicia-se um loop "do-while" interno para capturar a entrada do teclado. Dentro do loop interno, a função “Keypad_Key_Click()” é chamada para aguardar o clique de uma tecla. O valor da tecla é armazenado na variável ‘kp’.

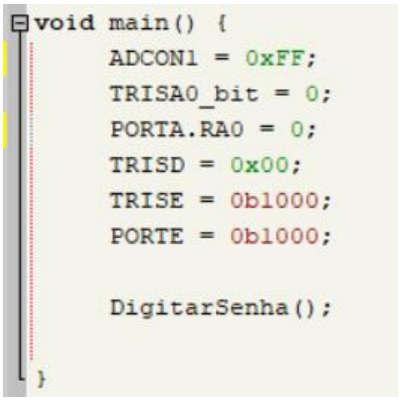
Em seguida, é feito um switch-case para determinar a ação a ser executada com base na tecla pressionada. Se a tecla pressionada corresponder a um dígito numérico (0 a 9), o valor correspondente é armazenado no vetor “senhadigitada”, a função “SenhaNoDisplay()” é chamada para exibir o asterisco (*) correspondente ao dígito digitado no display LCD, e o contador ‘i’ é incrementado.

Se a tecla pressionada for o asterisco (*), a variável ‘cfm’ é definida como 1 (confirmação) e a função “confirmar()” é chamada, passando a senha digitada (senhadigitada[]) e a senha mestre (senhamestre[]) como argumentos. A função “confirmar()” verifica se a senha digitada é igual à senha mestre e realiza as ações correspondentes.

Se a tecla pressionada for o símbolo de cardinal (#), é exibida a mensagem "Bloqueando..." no display LCD por um segundo, a função “Travar()” é chamada para bloquear o acesso, o display LCD é limpo e é exibida novamente a mensagem "Insira a senha:" na primeira linha, reiniciando o processo de digitação da senha. O código possui algumas teclas (A, B, C, D) que não possuem função específica definida.

O loop interno é executado continuamente, aguardando o pressionamento de teclas e executando as ações correspondentes até que o programa seja interrompido. O loop externo também é executado continuamente, garantindo que o processo de digitação da senha seja reiniciado após a sua conclusão.

Figura 43 - Bloco de Execução da Linguagem C



Fonte: Captura de tela no software MikroC PRO For PIC.

A função "main()" é a função principal do programa, é o ponto de partida para a execução do programa. Dentro desta função, são realizadas as seguintes funções:

- **ADCON1 = 0xFF;**
  Configura o registro ADCON1 com o valor 0xFF. Isso desativa todas as entradas analógicas do microcontrolador PIC18F4550, garantindo que todas as portas estejam configuradas como entradas digitais.
- **TRISA0_bit = 0;**
  Configura a PORTA A0 como saída digital, como os outras PORTA não são utilizadas, não foi necessário configura-las, exceto a porta RA6 que é configurada já pelo fuser Oscilador.
- **PORTA.RA0 = 0;**
  Define o valor inicial da PORTA A0 como 0, ou seja, o pino A0 inicia em nível lógico BAIXO.
- **TRISD = 0x00;**
  Configura todas as portas do PORTD como saídas digitais.
- **TRISE = 0b1000;**
  Configura a porta RE3 como entrada digital, enquanto as demais portas do PORTE são configuradas como saídas digitais.
- **PORTE = 0b1000;**
  Define o valor inicial da porta RE3 como 1, e o restante das portas como 0.
- **DigitarSenha();**
  Chama a função DigitarSenha(), que é responsável por capturar e processar a entrada do teclado, exibir o resultado no display LCD. Essa função contém a lógica principal do

programa para o controle da fechadura eletrônica com base na senha digitada pelo usuário.

2.3 MONTAGEM DA FECHADURA ELETRÔNICA

A montagem da fechadura eletrônica ocorreu seguindo o modelo criado no software Proteus 8 Professional, entretanto, como mencionado anteriormente, há uma diferença entre a montagem no software e a montagem real da fechadura eletrônica, que é o regulador de. Na prática, o regulador de tensão utilizado foi a fonte ajustável para protoboard 3.3V e 5V com interruptor (**Figura 10**) em vez do regulador de tensão em series (**Figura 11**), isso ocorreu pois como o próprio projeto já estava sendo elaborado na protoboard, se tornou mais prático usar a própria fonte ajustável vinda com a protoboard, assim economizando tempo e assim facilitar a realização de testes e ajustes durante o processo de desenvolvimento.

Primeiro, foi preciso realizar a gravação do código no microcontrolador e utilizou o software PICkit 3 e o dispositivo gravador PICkit3 (Fig. 46). Para realizar a gravação foi preciso conectar os pinos V_{PP}/MCLR (pino 1), V_{DD} (pino 11), V_{SS} (pino 12), PGD (pino 40) e PGC (pino 39) do PIC18F45550 ao pino corresponde no gravador. E depois utilizar o software PICkit3 para importar o código (em formato .hex) criado no MikroC PRO for PIC para o PIC.

Figura 44 - Gravaodr PICkit 3

Fonte: Captura de tela no site Mercado Livre[2]

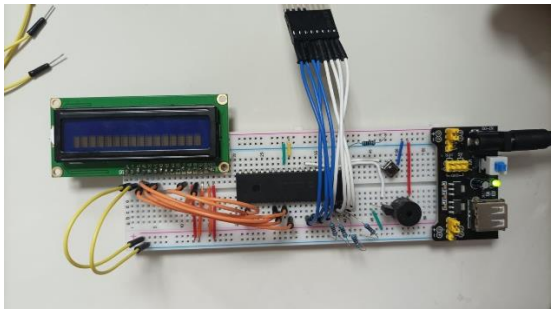
Dois problemas ocorreram durante a gravação, o primeiro foi má conexão entre os pinos na protoboard com os fios conectado ao gravador, ao resolver esta adversidade. O segundo erro

[2] Disponível em: < https://produto.mercadolivre.com.br/MLB-3175791201-kit-pickit-3-gravador-pic-usb-programador-socket-zif-nfe-_JM#is_advertising=true&position=1&search_layout=grid&type=pad&tracking_id=8be05df9-6538-49e3-9b29-2f5ba2794614&is_advertising=true&ad_domain=VQCATCORE_LST&ad_position=1&ad_click_id=NzJjZTAwNWUtNjhkYy00M2FjLThmZTEtNzJiYjgwNTg0NWEz>. Acesso em: 17 de Julho de 2023.

mostrou que algo armazenado no PIC, entretanto, não foi possível identificar se o PIC veio gravado de loja/fábrica ou acabou sendo gravado pelo próprio grupo durante as tentativas "falhas" quando as conexões estavam imperfeitas. Todavia, o arquivo armazenado foi apagado, e gravado um novo arquivo no PIC, o arquivo do código elaborado pelos alunos.

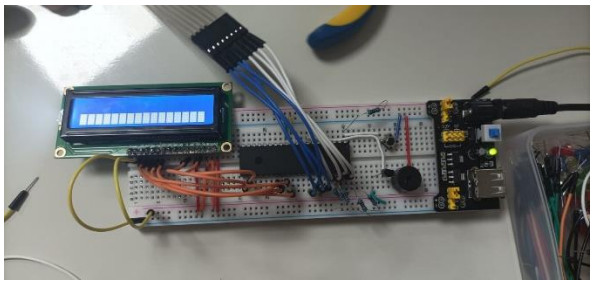
Durante a montagem e o teste do funcionamento foram surgindo alguns erros simples que foram logo resolvidos, como conexões que não estavam conectadas perfeitamente e também não havia a conexão do *backlight* do display LCD, assim o display estava com o visor bem mais escuro (Fig. 47) e foi realizada a conexão do pino A do display ao polo positivo protoboard e o pino K do display ao polo negativo da protoboard, permitindo uma visualização melhor no display (Fig. 48).

Figura 45 - Display LCD com Backlight Desativado



Fonte: Elaborada pelo autor

Figura 46 - Display LCD com Backlight Ativado



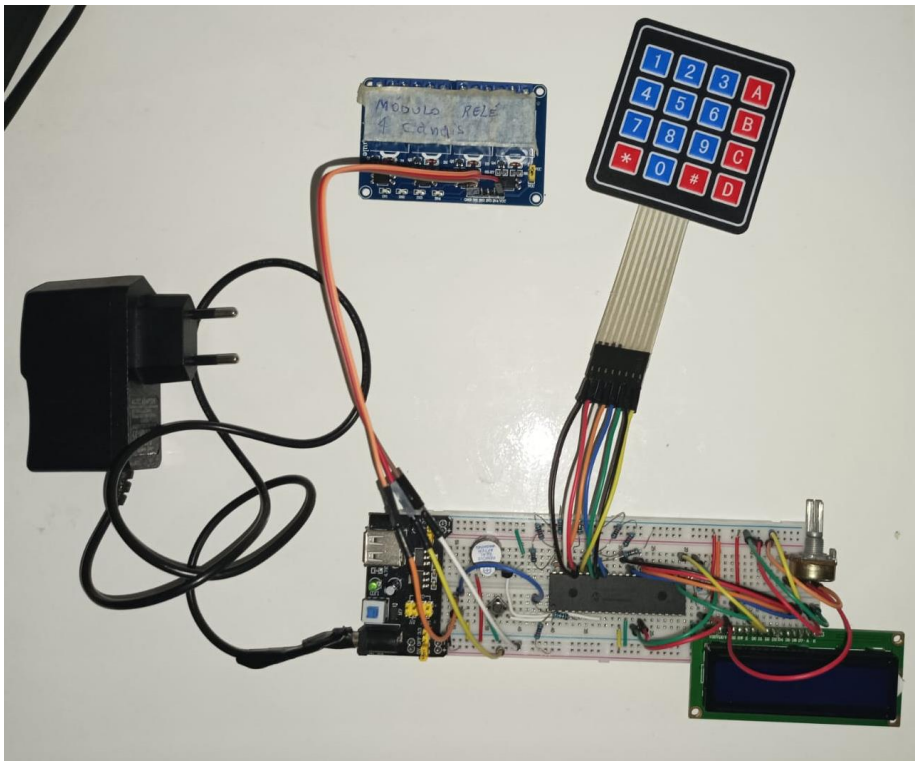
Fonte: Elaborada pelo autor

A montagem foi realizada em 2 etapas, pois a cristal oscilador 8MHz ainda não tinha sido entregue e por isso, 95% do projeto foi concluído na primeira etapa e a segunda etapa correspondeu apenas a montar o oscilador no projeto. Após a entrega do oscilador, a montagem foi concluída e começou a realização de testes. Vale mencionar que durante a segunda montagem, as cores dos fios foram trocadas para melhorar a identificação e isso facilitou na resolução do problema que será abordado em seguida.

A primeiro teste identificou um erro relacionado ao teclado, os fios relacionados ao teclado não apresentavam a mesma ordem identificada na **Figura 9**, as posições estavam trocadas. A configuração usada no Proteus 8 foi que as linhas do teclado eram conectadas respectivamente aos pinos 37-40 do PIC e as colunas eram conectadas aos pinos 33-36 do PIC. E assim, os pinos 37-40 seriam conectados aos fios das linhas 1 a 4 do teclado e os pinos 33-36 aos fios da coluna 5 a 8. Contudo, esta configuração estava errada, pois o dígito 1 ficou no botão D, o dígito 2 no botão C, o dígito 3 no botão B e assim por adiante.

Após identificar as posições de cada dígito, foi possível realizar a correção das posições e assim os pinos 33-36 do PIC foram conectados aos fios das linhas 1 a 4 e os pinos 37-40 foram conectados aos fios da coluna 5 a 8. E assim os dígitos passaram a estar na tecla correspondente. Abaixo se encontra uma foto da montagem completa da fechadura eletrônica, com o dispositivo desligado:

Figura 47 - Montagem Completa da Fechadura Eletrônica



Fonte: Elaborada pelo autor

3 RESULTADOS

O código-fonte elaborado no MikroC PRO for PIC foi compilado com sucesso, sendo assim, não aparentando nenhum erro. Ao aplicar o código no sistema elaborado no Proteus 8 Professional, a fechadura funcionou corretamente, executando todas as funcionalidades mínimas exigidas no projeto. Assim, é possível concluir que os resultados obtidos na elaboração e na montagem virtual da fechadura eletrônica foram um sucesso, pois alcançam todos os requisitos do projeto.

Durante a montagem física da fechadura, ocorreram certas adversidades, já relatadas, que foram possíveis de serem solucionadas e assim a fechadura eletrônica cumpriu todas as funções exigidas pelo projeto. Ou seja, o display apresentou a mensagem “Insira a senha: “, cada tecla pressionada apresentava “*” no display, na segunda linha, e com um espaço entre cada dígito. A senha apresentava quatro dígitos, o mínimo exigido.

O botão * era utilizada para verificar se a senha estava correta ou incorreta, ao pressionado, era informando a mensagem “Verificando...” na primeira linha, pelo tempo de 1 segundo, após isso, se a senha estivesse correta, seria informada “Senha correta” na segunda linha do display pelo tempo de dois segundos e depois apresentar a mensagem “Liberado” e assim permanecer, até que fosse pressionado o botão #, que era responsável por bloquear, ou seja, voltar ao estado original da fechadura, e no momento em que fosse pressionado deveria informar “Bloqueando...” na primeira linha do display pelo tempo de um segundo. E caso, a senha esteja errada, é informado “Senha incorreta” na segunda linha do display por dois segundos e após isto, retorna ao estado inicial de “Insira a senha: ” no display. Além disso, caso a seja esteja correta, o buzzer emite 2 bips curtos de 150 milissegundos e é enviando um sinal para o relé, destravando a fechadura, e caso esteja incorreta, um bip longo de 500 milissegundos é emitido pelo buzzer. É importante mencionar também outro requisito cumprido, um botão de reset no microcontrolador. Logo abaixo, há imagens das etapas mencionadas acima, em relação ao display e ao relé ligado, que foram cumpridas com sucesso.

Figura 48 - Insira a senha

Fonte: Elaborada pelo autor

Figura 49 - Aparecendo * ao digitar a senha

Fonte: Elaborada pelo autor

Figura 50 - Verificando a senha digitada

Fonte: Elaborada pelo autor

Figura 51 - Senha correta!

Fonte: Elaborada pelo autor

Figura 52 - Liberado

Fonte: Elaborada pelo autor

Figura 53 - Bloqueando

Fonte: Elaborada pelo autor

Figura 54 - Senha incorreta

Fonte: Elaborada pelo autor

4 CONCLUSÃO

Em relação a elaboração do código no MikroC Pro for PIC e a montagem da fechadura eletrônica tanto na prática quanto no software Proteus 8 Professional, foi possível desenvolver todo o projeto de forma clara e objetiva utilizando os conhecimentos adquiridos durante as aulas, sendo assim, fica evidente o quão produtivo e importante foi a elaboração deste projeto, precisando quando o resultado final do projeto é alcançado.

Em vista disso, a fechadura eletrônica desenvolvida utilizando o microcontrolador PIC18F4550 oferece uma solução prática para controle de acesso. A integração do PIC com os demais componentes eletrônicos permitirá uma operação funcional da fechadura e a utilização do PIC proporcionou um processamento rápido e confiável, garantindo o funcionamento adequado do sistema.

4.1 LIMITAÇÕES E MELHORIAIS

No entanto, é importante ressaltar algumas limitações identificadas no projeto. A utilização de uma senha *hard-coded* limita a flexibilidade do sistema, pois não permite a troca da senha pelo teclado ou então, na possibilidade de armazenar diversas senhas como ocorre com as fechaduras eletrônicas vendidas nas lojas. Além disso, a ausência de um mecanismo de bloqueio em casos de múltiplas tentativas de erros pode comprometer a segurança da fechadura e a ausência de um sistema anti-arrombamento para prevenir possíveis violações.

Para melhorar o sistema, pode-se implementar um menu de configurações que permita a troca da senha por meio do teclado, proporcionando maior flexibilidade ao usuário. A inclusão de um mecanismo de bloqueio temporário após um número específico de tentativas de senha incorretas ajudaria a prevenir ataques de força bruta. Além disso, é recomendável adicionar recursos de segurança adicionais, como criptografia de dados e detecção de intrusão, para aumentar a proteção contra ameaças externas. Essas melhorias visam aprimorar a funcionalidade e segurança da fechadura eletrônica, proporcionando uma experiência mais robusta e confiável aos usuários.

REFERÊNCIAS

ALT microcontroller. **CURSO PIC mikroC PRO**. Youtube, 2015-2016. Disponível em: <https://www.youtube.com/playlist?list=PLMFrcR3zyPe9wWfvBeWr2pn3ZCUdb38_P>. Acesso em: 17 de Junho de 2023.

C. BRAGA, Newton. **Curso de Eletrônica: Eletrônica Analógica**. São Paulo: Instituto NCB, 2012. v. 2.

Circuito de driver de relé de transistor com fórmula e cálculos. **JF Parede.** Disponível em: <https://jf-parede.pt/transistor-relay-driver-circuit-with-formula>. Acesso em 22 de Junho de 2023.

DA SILVA, José Eduardo H.; **Projeto Integrador – Fechadura Eletrônica baseada em microcontroladores PIC.** 30 de Abril de 2023.

DE CASTRO, Giovani. Introdução ao Regulador de Tensão. **RoboCore**. Disponível em: <https://www.robocore.net/tutoriais/introducao-regulador-de-tensao>. Acesso em: 17 de Junho de 2023.

Equipe MakerHero. Como utilizar o Display LCD 16x2 no Arduino?. **MakerHero**, 2022. Disponível em: <https://www.makerhero.com/blog/como-utilizar-o-display-lcd-16x2/>. Acesso em: 17 de Junho de 2023.

Microchip Technology Inc. **PC18F2455/2550/4455/4550 Data Sheet**. 2009.

Motorola Inc. **Amplifier Transistors BC546, B BC547, A, B, C BC548, A, B, C**. 1996.

Relé. **Electronica PT.** Disponível em: <https://www.electronica-pt.com/rele>. Acesso em 22 de Junho de 2023.

SENAI Play. **Transistor como chave acionando o relé | SENAI Play.** YouTube, 2022. Disponível em: <https://www.youtube.com/watch?v=a6ruOx26ybA>. Acesso em 22 de Junho de 2023.

Teclado matricial 4x4 con PIC. **TECmikro**. Disponível em: <https://tecmikro.com/content/74-teclado-matricial-4x4-pic>. Acesso em: 17 de Junho de 2023.

THOMSEN, Adilson. Como usar o Teclado Matricial 4×4 com Arduino. **MakerHero**, 2014. Disponível em: <https://www.makerhero.com/blog/teclado-matricial-4x4-arduino/>. Acesso em: 17 de Junho de 2023.

Tudo Sobre Relés (livro completo). **Newton C Braga,** 2009. Disponível em: <https://www.newtoncbraga.com.br/index.php/como-funciona/597-como-funcionam-os-reles?showall=1&limitstart>. Acesso em 22 de Junho de 2023.

VIEIRA, Carol Correia. Como utilizar o teclado matricial 4×4 keypad com Arduino. **Blog da Robótica**, 2022. Disponível em: <https://www.blogdarobotica.com/2022/06/30/como-utilizar-o-teclado-matricial-4x4-keypad-com-arduino/>. Acesso em: 17 de Junho de 2023.

Wels. **Tutorial Completo Microcontrolador PIC18F4550**. Youtube, 2018-2023. Disponível em: <https://www.youtube.com/playlist?list=PLO92aMMVufR9-Iw9g5wYBYfCHTpADyVuZ>. Acesso em: 17 de Junho de 2023.

WR Kits. **Curso de PIC Avançado**. Youtube, 2017-2018. Disponível em: <https://www.youtube.com/playlist?list=PLZ8dBTV2_5HQO03YC3RsgYvhwA375Ou79>. Acesso em: 17 de Junho de 2023.