CENTRO UNIVERSITÁRIO MAURÍCIO DE NASSAU
CURSO DE GRADUAÇÃO DE ENGENHARIA DA COMPUTAÇÃO

THIAGO FIGUEIREDO E MATHEUS PEREIRA

PROJETO COFRE

RECIFE
2020

THIAGO FIGUEIREDO E MATHEUS PEREIRA

MATRÍCULAS RESPECTIVAS: 01219936 01223448

PROJETO COFRE

Trabalho apresentado ao Curso de Graduação de Engenharia da Computação do Centro Universitário Maurício de Nassau do estado de Pernambuco, como pré-requisito para obtenção de nota da disciplina Sistema Microcontrolados, sob orientação do Professor Ph.D. Henrique Thadeu Baltar de Medeiros Cabral Moraes.

RECIFE

2020

UNINASSAU
CENTRO UNIVERSITÁRIO MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

LISTA DE FIGURAS

LISTA DE TABELAS

UNINASSAU
CENTRO UNIVERSITÁRIO MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

SUMÁRIO

1 INTRODUÇÃO

Não é de hoje que o ser humano tenta facilitar sua vida, os equipamentos eletrônicos estão bastante presentes na vida das pessoas, existem diversos componentes eletrônicos atualmente, todos com um objetivo específico, esses podem trabalhar em conjunto com outros, abrindo assim um caminho onde é possível utilizar elementos diversos para atender as necessidades das pessoas ou o mesmo pode funcionar sem utilizar componentes externos, dessa forma, podendo ocorrer várias limitações.

A finalidade de um cofre é garantir que um objeto permanece guardado em um local seguro e seu acesso deve ser limitado, ou seja, somente pessoas com autorização podem ter acesso ao conteúdo do cofre.

Sendo assim foi desenvolvido um sistema de segurança para garantir a integridade dos objetos que se encontram no cofre, o mesmo é protegido por senha, possui um *display LCD* (*liquid crystal display*) para ter uma interação visual entre o usuário e o dispositivo com sinais sonoros auxiliando quando os caracteres da senha são digitados.

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver um sistema para automação de um cofre usando componentes eletrônicos.

2.2 Objetivos Específicos

- Ao pressionar qualquer tecla emitir um sinal sonoro.
- A cada tecla pressionada exibir no LCD um asterisco.
- Ao pressionar asterisco (*) no teclado limpa o que está sendo exibido no LCD e apaga o conteúdo das variáveis de senhas.
- Ao pressionar a tecla cardinal (#) verifica se a senha digitada é igual à do sistema acontecendo a liberação ou não da trava do cofre.

3 DESCRIÇÃO TÉCNICA

Nesse projeto foi utilizado um ESP-32, onde foi ligado ao microcontrolador um display LCD 16x2 utilizando a biblioteca *LiquidCrystal* tendo a comunicação feita através de pinos *I/O (Input/Output)*, ligado ao display existe um potenciômetro, que por meio dele é possível controlar o brilho do mesmo, um teclado membrana 4x4 está conectado ao microcontrolador utilizando a biblioteca *Keypad* para fazer o mapeamento e a inserção dos caracteres da senha do cofre, a cada tecla pressionada um sinal sonoro é emitido através de um *buzzer* ativo, a trava solenoide necessita de uma alimentação externa para o seu funcionamento, para isso foi utilizado um relé, o mesmo está conectado ao ESP-32 recebendo assim a informação de quando é necessário liberar a trava e o relé também recebe a energia de uma fonte de alimentação externa que no caso do projeto foi de 9 V.

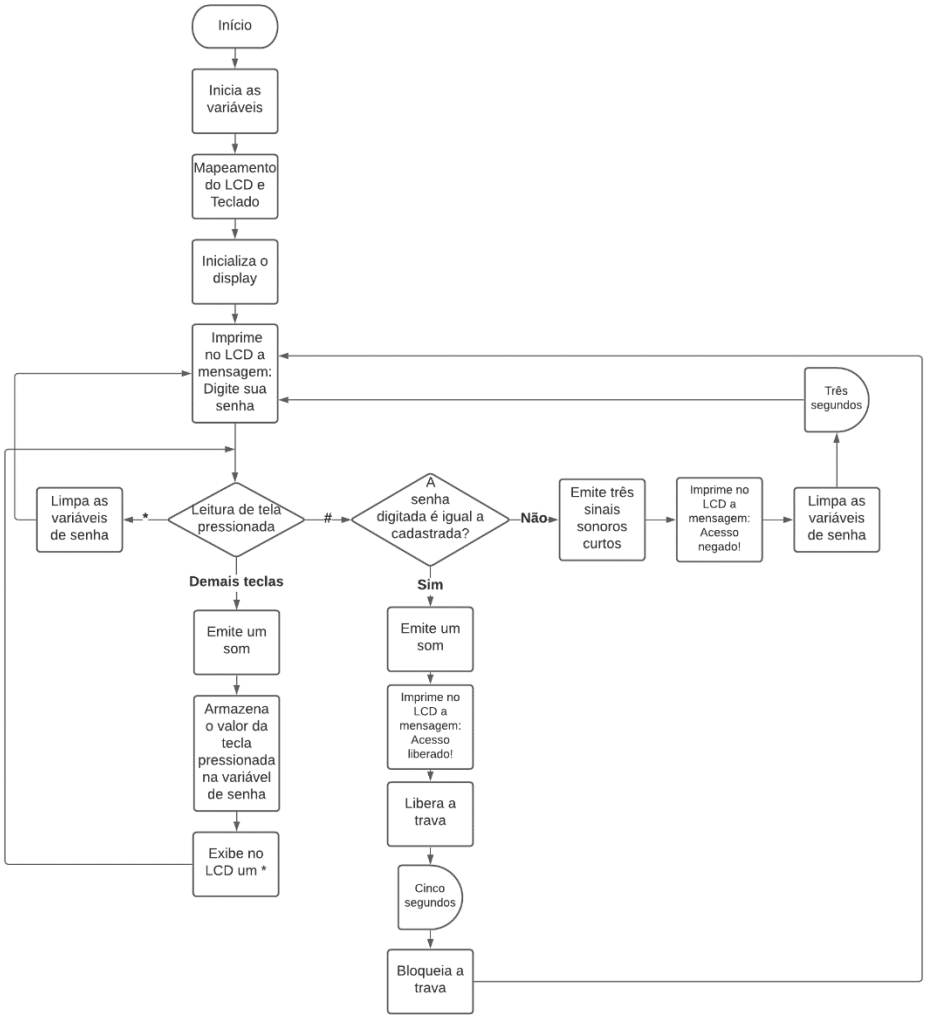

Figura 1. Fluxograma de funcionamento do cofre.

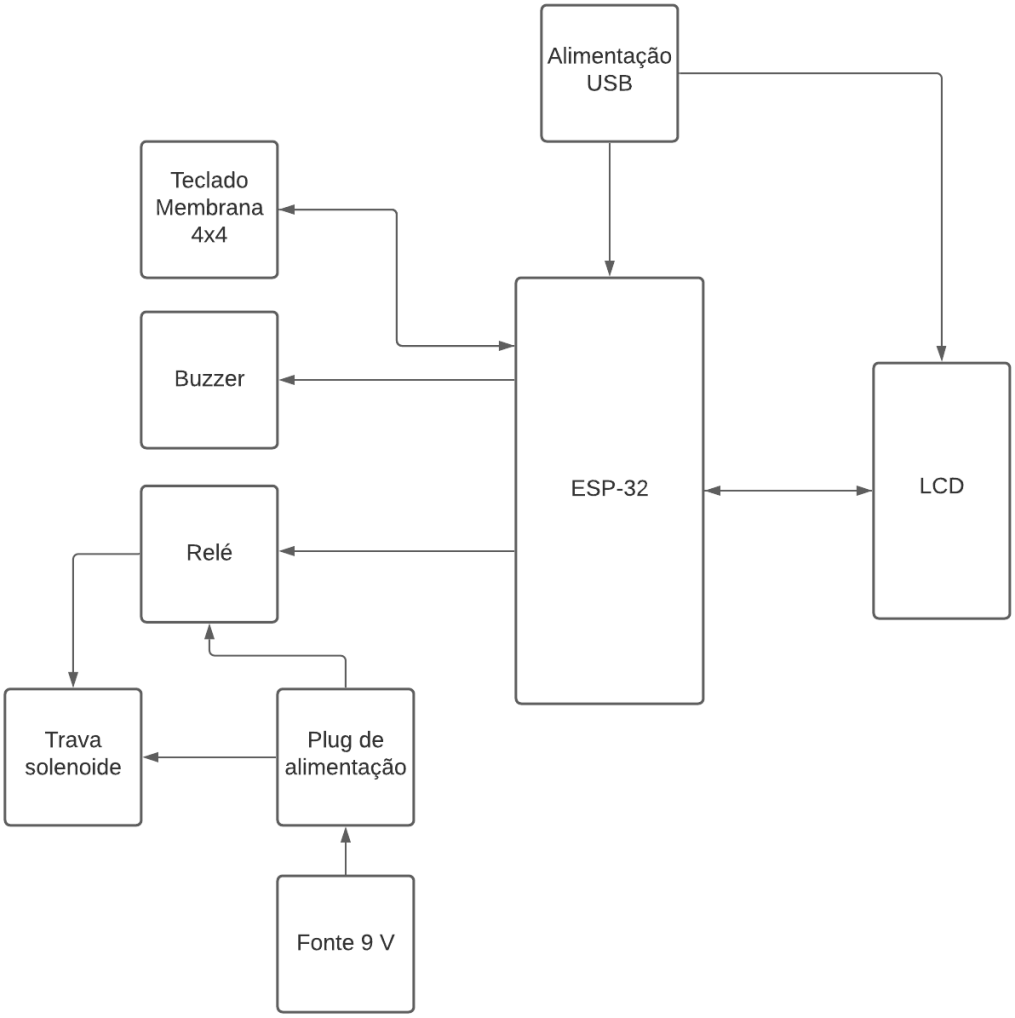

Figura 2. Diagrama de blocos do cofre.

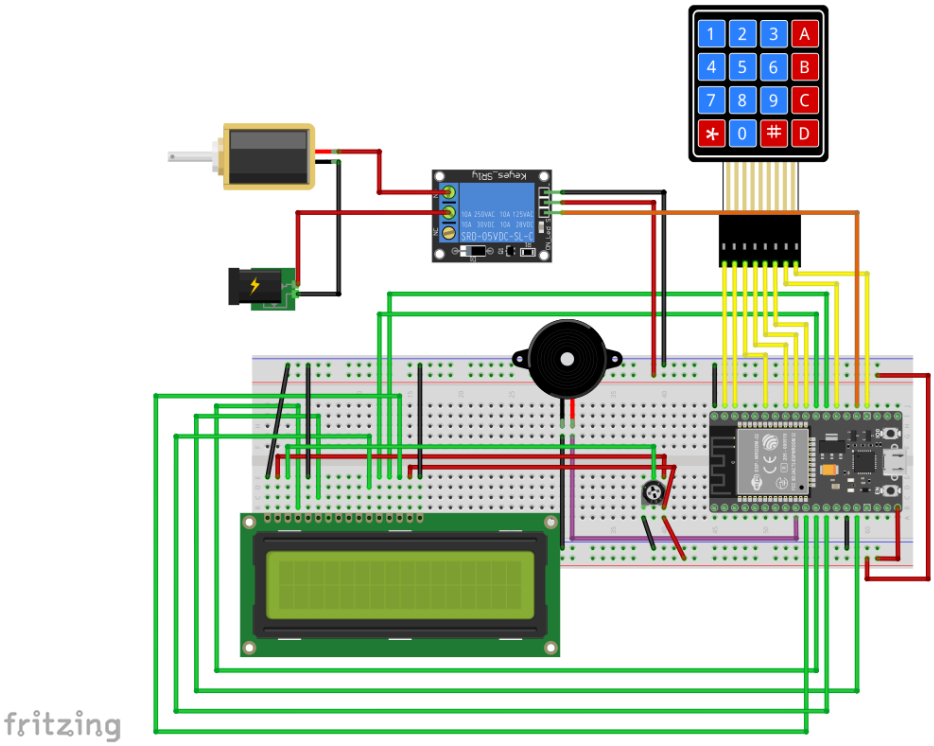

Figura 3. Esquemático do cofre.

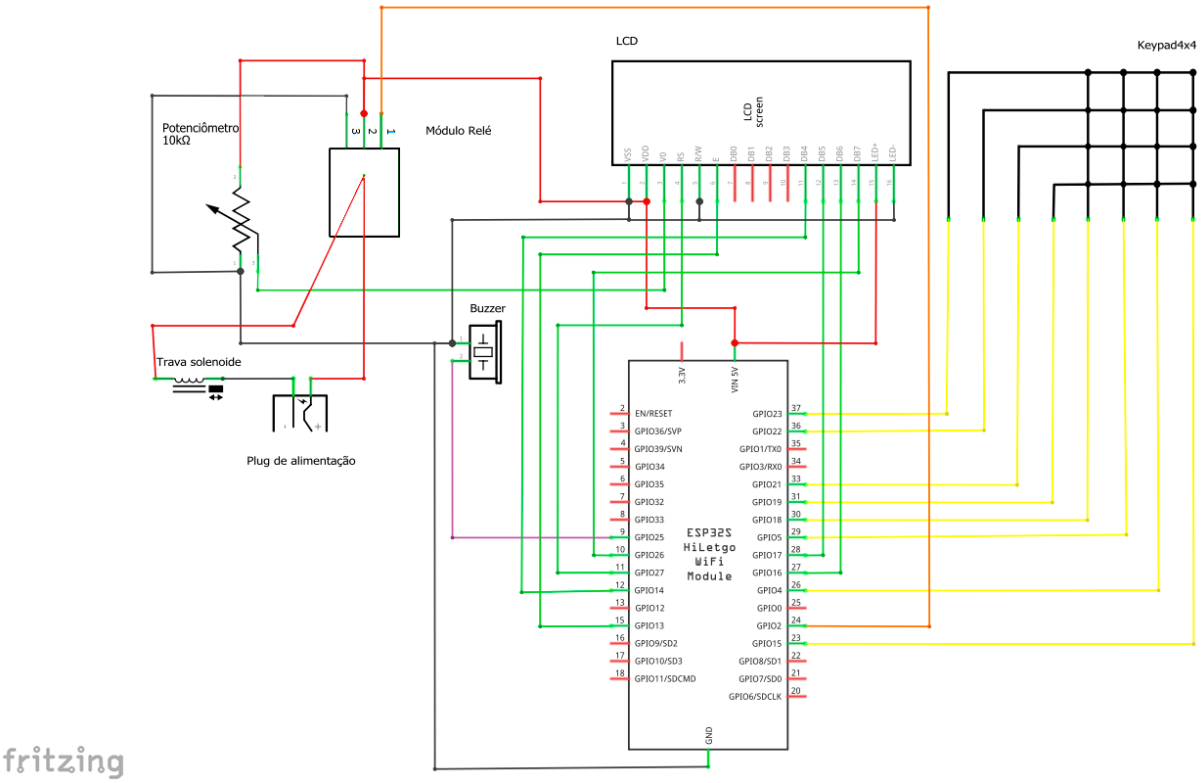

Figura 4. Esquemático elétrico do cofre.

3.1 Fundamento e cálculo da escolha dos componentes

A maioria dos componentes utilizados já estavam disponíveis para a realização do projeto por prévia aquisição, alguns dos que não estavam disponíveis foram: o ESP-32, a fonte de 9 V e o cabo micro USB.

Item	Descrição	Categoria	Quantidade	Valor (R$)	Total (R$)	Link
1	*Buzzer*	Ativo	1	1,49	1,49	www.baudaeletronica.com.br
2	Cabo micro USB	2.0	1	9,90	9,90	www.kabum.com.br
3	Display	LCD 16x2 com *backlight* verde	1	24,90	24,90	www.filipeflop.com
4	ESP-32	Microcontrolador	1	49,90	49,90	www.mercadolivre.com.br
5	Fechadura solenoide	Dc 12 V	1	22,29	22,29	www.aliexpress.com
6	Fonte	9 V	1	18,49	18,49	www.eletrogate.com
7	Kit de Jumpers	Macho/macho Fêmea/Macho	1	8,90	8,90	www.eletrogate.com
8	Plug de alimentação	P4 Fêmea	1	1,00	1,00	www.robocore.net
9	Potenciômetro	10 k	1	1,33	1,33	www.robocore.net
10	Protoboard	400 furos	1	14,90	14,90	www.mercadolivre.com.br

11	Protoboard	830 furos	1	17,90	17,90	www.mercadolivre.com.br
12	Relé	5 V 15 A	1	12,90	12,90	www.robocore.net
13	Teclado membrana	4x4	1	9,90	9,90	www.eletrogate.com
				Total	193,80	

Tabela 1. *BOM (bill of materials).*

4 INFORMAÇÕES PARA O DESENVOLVIMENTO DO *HARDWARE*

4.1 Tabelas de pinagem e utilização de cada componente

Buzzer (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	VDD, recebe a informação de quando é necessário emitir o som	ESP-32: GPIO25	Saída de dados
2	GND	ESP-32: GND	Aterramento

Tabela 2. Pinagem e utilização do componente buzzer.

Display *LCD* (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	VSS, aterramento	ESP-32: GND	Aterramento
2	VDD, alimentação 5 V	ESP-32: VIN	Alimentação
3	VO, ajuste de contraste do LCD	Potenciômetro: 3 (*Wiper*)	Disponibiliza uma tensão variável
4	RS, seleção de comandos (0) ou dados (1)	ESP-32: GPIO27	Saída de dados
5	R/W, leitura (1) ou escrita (0)	ESP-32: GND	Permite a escrita no LCD
6	E, *enable,* ativa (1) e desativa (0)	ESP-32: GPIO13	Saída de dados

7	D0, pino de dados 0	X	Utilizado somente na interface de 8 *bits*
8	D1, pino de dados 1	X	Utilizado somente na interface de 8 *bits*
9	D2, pino de dados 2	X	Utilizado somente na interface de 8 *bits*
10	D3, pino de dados 3	X	Utilizado somente na interface de 8 *bits*
11	D4, pino de dados 4	ESP-32: GPIO14	Transmissão de dados na interface de 4 ou 8 bits
12	D5, pino de dados 5	ESP-32: GPIO17	Transmissão de dados na interface de 4 ou 8 bits
13	D6, pino de dados 6	ESP-32: GPIO16	Transmissão de dados na interface de 4 ou 8 bits
14	D7, pino de dados 7	ESP-32: GPIO26	Transmissão de dados na interface de 4 ou 8 bits
15	A, ânodo do LED de backlight	ESP-32: VIN	Alimentação

16	K, cátodo do LED de backlight	ESP-32: GND	Aterramento

Tabela 3. Pinagem e utilização do componente display LCD.

Fechadura solenoide (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	VDD, 12 V	Relé: NO (*normally open*)	Alimentação quando o módulo estiver ligado
2	GND	Plug de alimentação: GND	Aterramento

Tabela 4. Pinagem e utilização do componente fechadura solenoide.

Plug de alimentação (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	GND	Fechadura solenoide: GND	Aterramento
2	VDD	Relé: C (*common*)	Chaveamento entre o NO (*normally open*) e o NC (*normally closed*)

Tabela 5. Pinagem e utilização do componente plug de alimentação.

Potenciômetro (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	GND	ESP-32: GND	Aterramento
2	VDD	ESP-32: VIN	Alimentação

3	*Wiper*, verifica a mudança de tensão	LCD: VO	Disponibiliza uma tensão variável

Tabela 6. Pinagem e utilização do componente potenciômetro.

Relé (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	*Input,* entrada de dados	ESP-32: GPIO2	Saída de dados
2	VDD, 5 V	ESP-32: VIN	Alimentação
3	GND	ESP-32: GND	Aterramento
4	NC (*normally closed*)	X	Alimentação quando o módulo estiver desligado
5	C (*common*)	Plug de alimentação: VDD	Chaveamento entre o NO (*normally open*) e o NC (*normally closed*)
6	NO (*normally open*)	Fechadura solenoide: VDD	Alimentação quando o módulo estiver ligado

Tabela 7. Pinagem e utilização do componente relé.

Teclado Membrana (pino)	Descrição do pino	Pino de conexão	Descrição da utilização do pino
1	Mapeia a linha 1	ESP-32: GPIO23	Saída de dados
2	Mapeia a linha 2	ESP-32: GPIO22	Saída de dados

3	Mapeia a linha 3	ESP-32: GPIO21	Saída de dados
4	Mapeia a linha 4	ESP-32: GPIO19	Saída de dados
5	Mapeia a coluna 1	ESP-32: GPIO18	Entrada de dados
6	Mapeia a coluna 2	ESP-32: GPIO5	Entrada de dados
7	Mapeia a coluna 3	ESP-32: GPIO4	Entrada de dados
8	Mapeia a coluna 4	ESP-32: GPIO15	Entrada de dados

Tabela 8. Pinagem e utilização do componente teclado membrana.

4.2 Descrição do *firmware*

O código principal começa com a inclusão das bibliotecas necessárias provenientes do *framework* Arduino que foram: a *Keypad* e a *Liquid Crystal*. São declaradas todas as variáveis , em seguida é feito o mapeamento do teclado e do display. Os conjuntos de códigos que se repetem são definidos em funções para deixar o código mais organizado, na função *setup* são definidas as configurações iniciais dos componentes conectados ao microcontrolador, já na função *loop* é onde encontra-se a lógica principal do projeto, esse código está sempre se repetindo tornando possível recuperar o valor das teclas digitadas no teclado, a partir de determinada tecla é possível limpar o conteúdo do display, verificar se a senha digitada está igual a senha cadastrada, fazer a liberação ou não da fechadura do cofre e emitir sinais sonoros. Na parte final desse documento encontra-se em anexo o conteúdo do *firmware*, o código está todo comentado, sendo possível obter uma explicação mais detalhada dos processos.

A biblioteca *Keypad* é responsável pelas interações que são feitas no teclado, fazendo o uso da mesma não se faz necessário o uso de resistores na protoboard do projeto, pois ela aciona os resistores *pull-up* do ESP-32 via código. Criada para facilitar a leitura do código, dispõe de várias funções, algumas delas servem para verificar o estado atual da tecla, permite também recuperar se alguma tecla teve mudança na sua configuração, dentre várias outras funcionalidades, dessa forma, a biblioteca omite no código principal as definições necessárias (pinos de entrada ou saída, escrita

e leitura digital) para utilizar o teclado e converte diversas variáveis utilizando o mínimo de espaço no microcontrolador.

A biblioteca *Liquid Crystal* permite a comunicação do framework do Arduino para displays e aceita de 4 a 8 *bits*, com isso é possível usar 4 ou 8 linhas de dados. Visando diminui a complexidade da manipulação dos *bits* do *LCD* tornando o mapeamento mais simples, com ela é possível definir o tamanho do *display,* assim como, a posição de onde será inserido as informações, desloca com facilidade dados na tela seja para esquerda ou para direita, sendo assim possível fazer diversas funções de forma simples.

5 CONCLUSÕES

Com este trabalho, foi possível verificar o funcionamento e a implementação de um sistema de cofre com fechadura e sinais sonoros, com o auxílio de programas, como: *fritzing* para desenvolver o esquemático e esquemático elétrico, *visual studio code* para build e upload do firmware no microcontrolador, *lucidchart* para o criar o fluxograma de funcionamento e o diagrama de blocos. Dessa forma foi possível obter o entendimento prático da disciplina de sistemas microcontrolados.

Utilizando o ESP-IDF (*Espressif IoT Development Framework*) houveram algumas dificuldades para conectar o LCD, como o framework utiliza a linguagem C, era necessário as inclusões de algumas bibliotecas, mesmo após muitas tentativas não foi possível conectar o *display* nesse *framework*, dessa forma foi autorizado a utilização do framework do Arduino, com isso a biblioteca *Liquid Crystal* foi inclusa no projeto e o LCD funcionou perfeitamente.

Ao implementar o *buzzer* houve dificuldade em encontrar a sonoridade adequada para liberação e negação do acesso e o intervalo dos sons ao pressionar as teclas.

Houve dificuldade na implementação do modo *sleep*, onde foi observado que ao entrar nesse estado, ele reduz a energia no microcontrolador mas não impede a energia de ir para o display, várias tentativas foram feitas para desligar o *backlight* do LCD e não foi possível desligar a luminosidade do *display*, somente o que ele exibia.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, Yure. **ESP32 pinout – Guia Básico de GPIOs.** Disponível em: <https://blog.smartkits.com.br/esp32-pinout-guia-basico-de-gpios>. Acesso em: 3 nov. 2020.

ANDREWS, Christopher. **Keypad Library.** Disponível em: <https://playground.arduino.cc/Code/Keypad>. Acesso em: 6 nov. 2020.

ARDUINO. **LiquidCrystal Library.** Disponível em: <https://www.arduino.cc/en/Reference/LiquidCrystal>. Acesso em: 14 nov. 2020.

INSTITUTO FEDERAL DE SANTA CATARINA. **AULA 1 - Microcontroladores - Engenharia.** Disponível em: <https://wiki.ifsc.edu.br/mediawiki/index.php/AULA_1_-_Microcontroladores_-_Engenharia>. Acesso em: 17 nov. 2020.

MURTA, Gustavo. **Guia completo do Display LCD – Arduino.** Disponível em: <https://blog.eletrogate.com/guia-completo-do-display-lcd-arduino>. Acesso em: 14 nov. 2020.

OLIVEIRA, Jailson. **ESP32 – Especificação Técnica.** Disponível em: <https://xprojetos.net/esp32-especificacao-tecnica>. Acesso em: 3 nov. 2020.

ROBOCORE. **Módulo Relé Arduino.** Disponível em: <https://www.robocore.net/tutoriais/modulo-rele-arduino>. Acesso em: 23 nov. 2020.

THOMSEN, Adilson. **Como usar o Teclado Matricial 4×4 com Arduino.** Disponível em: <https://www.filipeflop.com/blog/teclado-matricial-4x4-arduino>. Acesso em: 6 nov. 2020.

ANEXO

```
1.  // Inclusão das blibliotecas utilizadas
2.  #include "Keypad.h"
3.  #include <LiquidCrystal.h>
4.
5.  // Define os GPIOs a serem utilizados
6.  #define TRAVA 2
7.  #define BUZZER 25
8.
9.  // Define uma senha padrão para o acesso
10. const String SENHA_DO_COFRE = "123A";
11. // Variável que recebe a senha
12. String senha_digitada = "";
13. // Variável que concatena os símbolos de "*" que aparecerão no LCD
14. String senha_oculta = "";
15.
16. // Define as quantidades de linhas e colunas que serão utilizadas no teclado
17. const byte LINHAS = 4;
18. const byte COLUNAS = 4;
19.
20. // Matriz dos valores que serão utilizadas quando as teclas forem pressionadas
21. char teclas[LINHAS][COLUNAS] = {
22.     {'1', '2', '3', 'A'},
23.     {'4', '5', '6', 'B'},
24.     {'7', '8', '9', 'C'},
25.     {'*', '0', '#', 'D'}};
26.
27. // Indica os Gpios que serão utilizados no esp-32
28. byte linhasGpio[LINHAS] = {23, 22, 21, 19};
29. byte colunasGpio[COLUNAS] = {18, 5, 4, 15};
30.
31. // Faz o mapeamento das teclas de acordo com o teclado
32. Keypad keypad = Keypad(makeKeymap(teclas), linhasGpio, colunasGpio, LINHAS, COLUNA
    S);
33.
34. // Faz o mapeamento do LCD
35. LiquidCrystal lcd(27, 13, 14, 17, 16, 26);
36.
37. // Função que contém a configuração inicial do LCD
38. void telaInicial()
39. {
40.   lcd.clear();
41.   lcd.print("Digite sua senha");
42. }
43.
44. // Função que limpa o que tiver contido nas variáveis de senha
45. void limpaSenhas()
46. {
47.   senha_digitada = "";
48.   senha_oculta = "";
49. }
50.
51. // Função de configuração inicial do microcontrolador
52. void setup()
53. {
54.   // Taxa de transferência de bits por segundo
55.   Serial.begin(9600);
56.   // Define os GPIOs como saída
57.   pinMode(BUZZER, OUTPUT);
58.   pinMode(TRAVA, OUTPUT);
59.   // Define o tamanho do LCD, colunas e linhas respectivamente
60.   lcd.begin(16, 2);
```

```
61.   // Chama a função telaInicial
62.   telaInicial();
63. }
64.
65. // Função de repetição
66. void loop()
67. {
68.   // Variável armazena a tecla digitada
69.   char tecla_pressionada = keypad.getKey();
70.
71.   // Verifica se uma tecla foi pressionada
72.   if (tecla_pressionada)
73.   {
74.
75.     // Testa se a tecla pressionada possui o valor "*"
76.     if (tecla_pressionada == '*')
77.     // Bloco responsável por retornar a configuração inicial
78.     {
79.       telaInicial();
80.       limpaSenhas();
81.     } // Se o valor da tecla não foi "*", verifica se a tecla pressionada possui o
    valor "#"
82.     else if (tecla_pressionada == '#')
83.     {
84.       // Bloco responsável por verificar se a senha digitada está igual a padrão
85.       if (SENHA_DO_COFRE == senha_digitada)
86.       // Se a senha estiver correta, executa o bloco de código a seguir
87.       {
88.         // Limpa a tela, remove tudo que estiver escrito na tela
89.         lcd.clear();
90.         // Imprime na tela "Acesso Liberado!"
91.         lcd.print("Acesso Liberado!");
92.         // Emite um sinal sonoro
93.         digitalWrite(BUZZER, HIGH);
94.         // Aguarda 250 milissegundos
95.         delay(250);
96.         // Omite o sinal sonoro
97.         digitalWrite(BUZZER, LOW);
98.         // Abre a trava
99.         digitalWrite(TRAVA, HIGH);
100.                // Aguarda 5000 milissegundos
101.                delay(5000);
102.                // Fecha a trava
103.                digitalWrite(TRAVA, LOW);
104.                // Retorna as configurações iniciais
105.                limpaSenhas();
106.                telaInicial();
107.              }
108.              else
109.              // Se a senha estiver incorreta, executa o bloco de código a seguir

110.              {
111.                // Limpa a tela, remove tudo que estiver escrito na tela
112.                lcd.clear();
113.                // Imprime na tela "Acesso Negado!"
114.                lcd.print("Acesso Negado!");
115.                // Repete o bloco a seguir três vezes
116.                for (int i = 0; i < 3; i++)
117.                {
118.                  // Aguarda 150 milissegundos
119.                  delay(150);
120.                  // Emite um sinal sonoro
121.                  digitalWrite(BUZZER, HIGH);
122.                  // Aguarda 150 milissegundos
123.                  delay(150);
124.                  // Omite o sinal sonoro
```

```
125.                 digitalWrite(BUZZER, LOW);
126.               }
127.               // Aguarda 3000 milissegundos
128.               delay(3000);
129.               // Retorna as configurações iniciais
130.               limpaSenhas();
131.               telaInicial();
132.             }
133.           } // Se a tecla não teve o valor "*" nem "#", executa o bloco de códig
    o a seguir
134.           else
135.           {
136.             // Concatena na váriável as teclas que foram pressionadas
137.             senha_digitada += tecla_pressionada;
138.             // Concatena "*" na váriável a quantidade de vezes que a tecla foi p
    ressionada
139.             senha_oculta += "*";
140.             // Imprime na tela o valor contido na variável senha_oculta
141.             lcd.print(senha_oculta);
142.             // Emite um sinal sonoro
143.             digitalWrite(BUZZER, HIGH);
144.             // Aguarda 150 milissegundos
145.             delay(150);
146.             // Omite o sinal sonoro
147.             digitalWrite(BUZZER, LOW);
148.           }
149.         }
150.         // Define a posição do ponteiro
151.         lcd.setCursor(0, 1);
152.       }
```