Izabela Lima da Costa
Heloisa Rodrigues

Relatório Final - Tranca eletrônica

Disciplina: Projeto Integrador 3B
Curso: Engenharia Elétrica
Docente: Prof. Marcelo Carvalho Tosin

Londrina
2025

RESUMO

Este trabalho apresenta o desenvolvimento de uma tranca eletrônica para aplicações em controle de acesso, como portas residenciais e ambientes laboratoriais. O objetivo do projeto é aumentar a segurança e a praticidade em relação às fechaduras mecânicas convencionais, utilizando recursos de eletrônica digital e sistemas embarcados.

A tranca é composta por um microcontrolador stm32, módulo de interface com o usuário teclado matricial, driver de potência e atuador eletromecânico. O sistema permite o cadastro e a validação de senhas de acesso, acionando o mecanismo de travamento apenas quando a autenticação é bem-sucedida.

Palavras-chave: tranca eletrônica; controle de acesso; microcontrolador; segurança.

Capítulo 1

Introdução Teórica

Sistemas de controle de acesso vêm se tornando cada vez mais comuns em ambientes residenciais, comerciais e industriais, substituindo ou complementando as fechaduras mecânicas tradicionais. A tranca eletrônica insere-se nesse contexto como uma solução que combina praticidade, flexibilidade de configuração e maior segurança, permitindo a autenticação de usuários por meio de senhas ou outros métodos de identificação eletrônica. Diferentemente das fechaduras convencionais, que se baseiam apenas em mecanismos mecânicos, as trancas eletrônicas utilizam circuitos eletrônicos e dispositivos de comando para liberar ou bloquear o acesso de forma controlada.

Do ponto de vista da engenharia elétrica, um sistema de tranca eletrônica pode ser entendido como um conjunto de blocos funcionais: unidade de processamento, interface de entrada do usuário, circuito de acionamento de potência e atuador eletromecânico. A unidade de processamento é, em geral, implementada por um microcontrolador, responsável por ler os sinais de entrada, executar a lógica de decisão (como a comparação da senha digitada com um valor previamente armazenado) e comandar o acionamento da fechadura elétrica.

A interface de entrada do usuário é frequentemente realizada por meio de um teclado matricial. Esses dispositivos convertem ações físicas do usuário em sinais elétricos que podem ser interpretados pelo microcontrolador. A correta leitura dessas entradas exige conhecimento de eletrônica digital, técnicas de varredura de teclado (scanning) e, em alguns casos, tratamento de *bouncing* de contatos mecânicos, de forma a garantir que a informação capturada pelo sistema seja confiável.

Outro bloco fundamental é o circuito de acionamento da tranca propriamente dita. Como o microcontrolador normalmente não é capaz de fornecer corrente suficiente para acionar diretamente uma fechadura elétrica, faz-se necessário o uso de estágios de potência, compostos por transistores, MOSFETs, drivers ou relés, além de componentes de proteção como diodos de roda livre. Esse estágio atua como uma interface entre a lógica de controle (nível baixo de potência) e o atuador eletromecânico (nível mais alto de potência), garantindo o funcionamento seguro e a integridade dos componentes eletrônicos.

No contexto específico deste projeto, desenvolvido como parte de uma disciplina do

curso de Engenharia Elétrica, o foco principal esteve na concepção do circuito eletrônico e no desenvolvimento do layout de placa de circuito impresso (PCI) utilizando o software Altium Designer. Assim, foram elaborados o diagrama esquemático e o roteamento da placa, considerando boas práticas de projeto, como organização de trilhas, posicionamento de componentes, dimensionamento de trilhas de potência e cuidados com ruídos e aterramento. Devido ao tempo reduzido para a execução prática, o sistema foi implementado apenas em nível de protótipo em *protoboard*, não havendo a fabricação da placa final dentro do período letivo.

Apesar de o protótipo não ter evoluído para a versão final em PCI, o desenvolvimento teórico e o projeto no Altium permitiram consolidar conceitos importantes de eletrônica analógica e digital, sistemas embarcados e projeto de hardware. A partir da modelagem do sistema em blocos, da escolha dos componentes e da transposição do circuito esquemático para o layout de placa, o projeto de tranca eletrônica cumpriu seu papel pedagógico de integrar diferentes conteúdos da disciplina, aproximando o aluno da realidade de projetos profissionais em engenharia.

Capítulo 2

Desenvolvimento do Código

O código-fonte da tranca eletrônica foi desenvolvido em linguagem C, utilizando a biblioteca HAL fornecida pelo ambiente STM32CubeIDE para o microcontrolador STM32F103C8T6. A lógica de controle é responsável por fazer a leitura do teclado matricial 3×4, comparar a senha digitada com uma senha previamente cadastrada na memória do microcontrolador e, a partir disso, acionar o relé que libera a fechadura ou indicar o erro por meio de um LED.

Durante o desenvolvimento do firmware, algumas dificuldades se destacaram. A primeira foi a correta varredura (*scanning*) do teclado matricial, exigindo o mapeamento adequado das linhas e colunas, além do tratamento de *bouncing* das teclas, para evitar leituras múltiplas de um mesmo acionamento. Também foi necessário ajustar a temporização e a forma de detecção de teclas pressionadas, de maneira a garantir um funcionamento estável mesmo em um ambiente de prototipagem em *protoboard*, sujeito a ruídos e contatos imprecisos.

Outra dificuldade importante esteve na integração da lógica de senha com o tempo de acionamento da fechadura. Foi preciso definir um protocolo simples de interação com o usuário: a entrada da sequência numérica, a confirmação da senha por uma tecla específica (por exemplo, #), e o tratamento dos casos de erro com indicação visual. Além disso, o controle do relé através do transistor 2N2222 exigiu atenção à configuração correta dos pinos do microcontrolador (nível lógico, modo de saída, corrente máxima, etc.), garantindo que o acionamento da carga ocorresse sem danificar o microcontrolador.

Devido às limitações de tempo da disciplina, o código foi testado principalmente em nível de protótipo, com foco na funcionalidade básica da tranca: leitura de senha, validação e acionamento do relé/LED. Recursos adicionais, como contagem de tentativas, bloqueio temporário ou menus mais complexos no display, foram deixados como possibilidades para trabalhos futuros. Ainda assim, a implementação permitiu consolidar conceitos de programação embarcada, manipulação de GPIOs, tratamento de periféricos digitais e organização modular do código.

2.0.1 Implementação em Linguagem C

A seguir é apresentado um exemplo de código em C para o STM32F103C8T6, ilustrando a lógica de funcionamento da tranca eletrônica. O exemplo supõe o uso da biblioteca HAL, com os pinos configurados no STM32CubeMX da seguinte forma (podendo ser adaptado conforme o layout do projeto):

- Linhas do teclado (R0–R3): PB8, PB9, PB10, PB11 (saída digital).
- Colunas do teclado (C0–C2): PB12, PB13, PB14 (entrada com *pull-up* interno).
- Saída para o transistor do relé (base via R4): PA1 (saída digital).
- LED vermelho de erro: por exemplo PB5 (saída digital).
- Display I2C: ligado ao barramento I2C1 (não detalhado aqui, uso opcional).

O código abaixo implementa:

1. Leitura do teclado 3×4 por varredura.
2. Montagem da senha digitada em um *buffer*.
3. Comparação com uma senha fixa (`"1234"`).
4. Acionamento do relé em caso de senha correta.
5. Acendimento do LED vermelho em caso de senha incorreta.

Capítulo 3

Conclusão

3.1 Conclusão

O desenvolvimento da tranca eletrônica neste projeto permitiu aplicar, de forma integrada, diversos conceitos abordados ao longo das disciplinas de Engenharia Elétrica, em especial aqueles relacionados a sistemas embarcados, eletrônica digital e projeto de hardware. Embora o projeto não tenha sido concluído em sua forma final — com placa de circuito impresso confeccionada e sistema totalmente montado em campo — foi possível implementar e testar um protótipo em *protoboard* e elaborar o projeto da PCI no Altium Designer, que constituiu o foco principal da disciplina.

Do ponto de vista prático, a montagem em *protoboard* possibilitou validar parcialmente o funcionamento da lógica de controle: leitura do teclado matricial, comparação da senha, acionamento do relé por meio do transistor. Durante essa etapa, ficaram evidentes as dificuldades típicas de projetos embarcados, tais como o tratamento de ruídos, *bouncing* nas teclas, temporização adequada e correta configuração dos pinos do microcontrolador STM32F103C8T6. Essas dificuldades, apesar de terem consumido parte do tempo disponível, contribuíram para o amadurecimento na depuração de sistemas reais, que nem sempre se comportam exatamente como o previsto em simulações ou no papel.

No âmbito do projeto de hardware, a elaboração do esquemático e do layout da placa no Altium permitiu exercitar boas práticas de projeto, como a organização de blocos funcionais, o posicionamento criterioso dos componentes, o roteamento de trilhas de potência e sinais, e a preocupação com o aterramento e a proteção dos dispositivos semicondutores. Ainda que a placa não tenha sido fabricada dentro do prazo da disciplina, o arquivo de projeto gerado representa um estágio avançado de desenvolvimento e poderia servir como base para a continuidade do trabalho em semestres futuros ou em projetos de extensão.

Como possíveis desdobramentos futuros, destacam-se a confecção da PCI projetada, a montagem da tranca em um gabinete definitivo e a implementação de funcionalidades adicionais no firmware, como bloqueio temporário após múltiplas tentativas incorretas, registro

de eventos, uso mais intensivo do display I^2C e integração com outros métodos de autenticação (por exemplo, RFID ou biometria). Dessa forma, o projeto não apenas cumpriu o papel pedagógico proposto na disciplina, como também deixou uma base sólida para evoluções posteriores, aproximando as estudantes da realidade de desenvolvimento de sistemas de controle de acesso utilizados no mercado.

Anexos

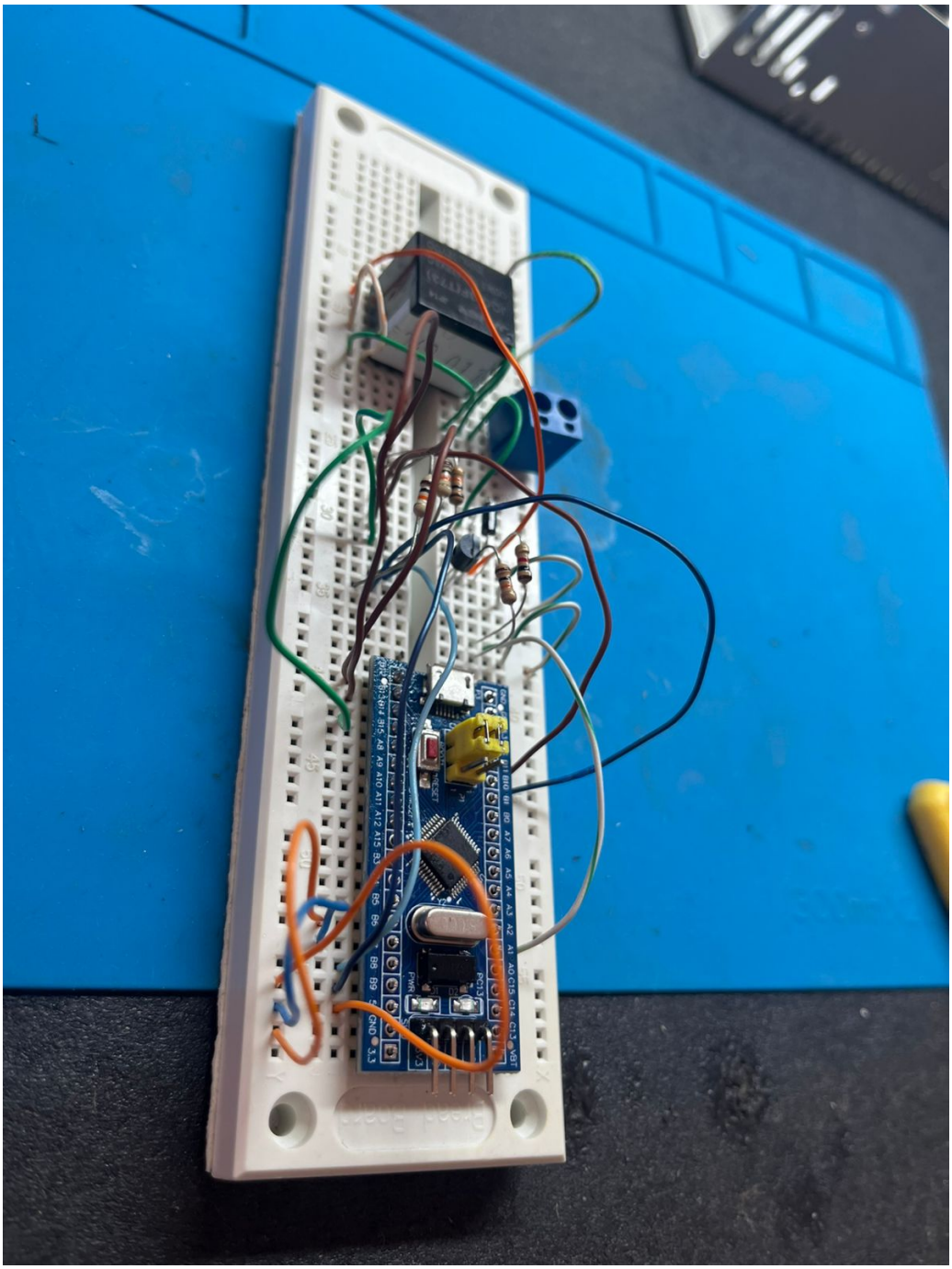
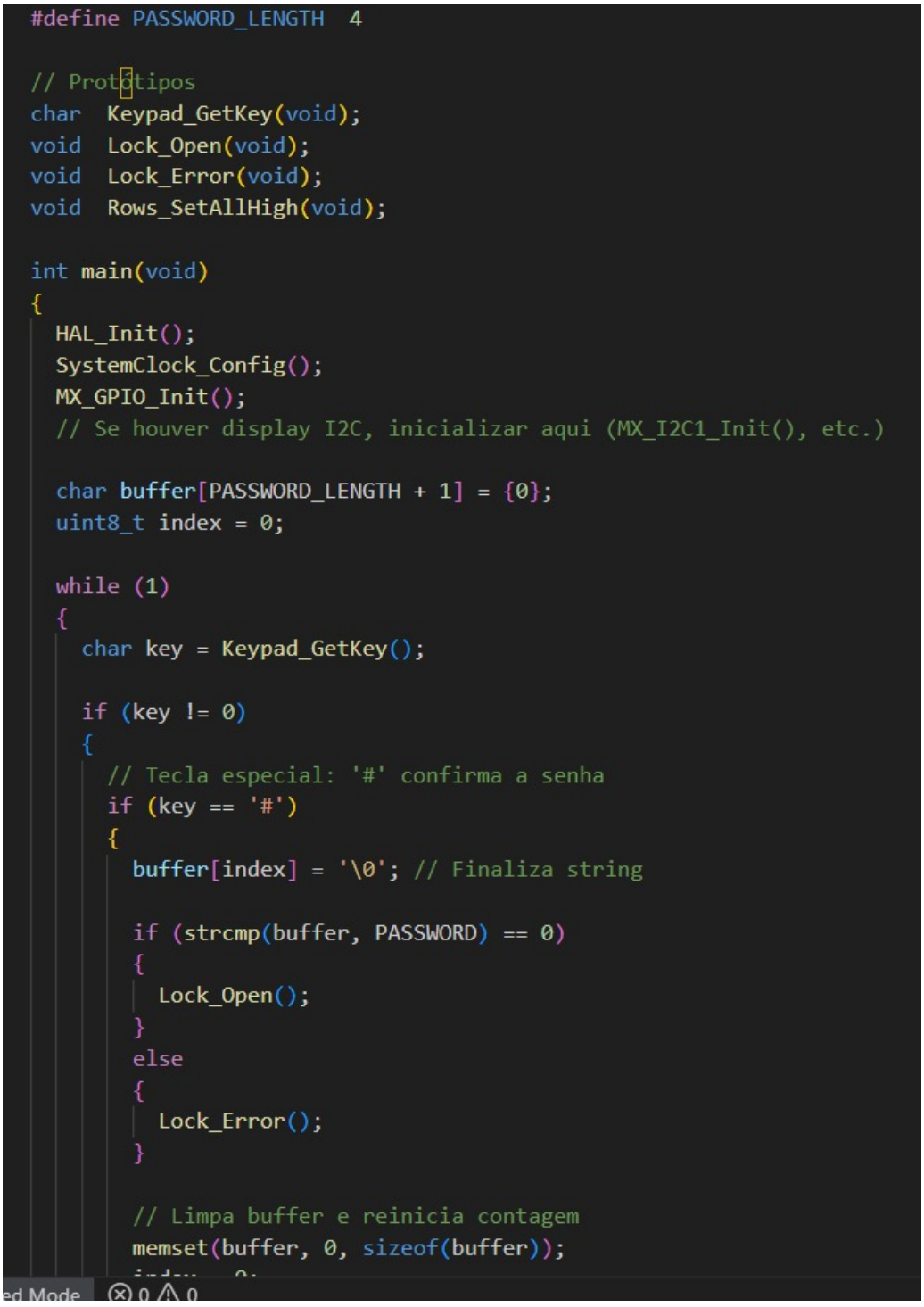


Figura 3.1:

```c
#include "main.h"
#include <string.h>

// ---- Definições de pinos (ajustar conforme CubeMX) ----
#define ROW0_GPIO_Port   GPIOB
#define ROW0_Pin         GPIO_PIN_8
#define ROW1_GPIO_Port   GPIOB
#define ROW1_Pin         GPIO_PIN_9
#define ROW2_GPIO_Port   GPIOB
#define ROW2_Pin         GPIO_PIN_10
#define ROW3_GPIO_Port   GPIOB
#define ROW3_Pin         GPIO_PIN_11

#define COL0_GPIO_Port   GPIOB
#define COL0_Pin         GPIO_PIN_12
#define COL1_GPIO_Port   GPIOB
#define COL1_Pin         GPIO_PIN_13
#define COL2_GPIO_Port   GPIOB
#define COL2_Pin         GPIO_PIN_14

#define RELAY_GPIO_Port  GPIOA
#define RELAY_Pin        GPIO_PIN_1   // Base do 2N2222 via R4

#define LED_ERR_GPIO_Port GPIOB
#define LED_ERR_Pin       GPIO_PIN_5   // LED vermelho

// ---- Configuração da senha ----
#define PASSWORD         "1234"
#define PASSWORD_LENGTH  4

// Protótipos
char  Keypad_GetKey(void);
void  Lock_Open(void);
void  Lock_Error(void);
void  Rows_SetAllHigh(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
```

Figura 3.2:

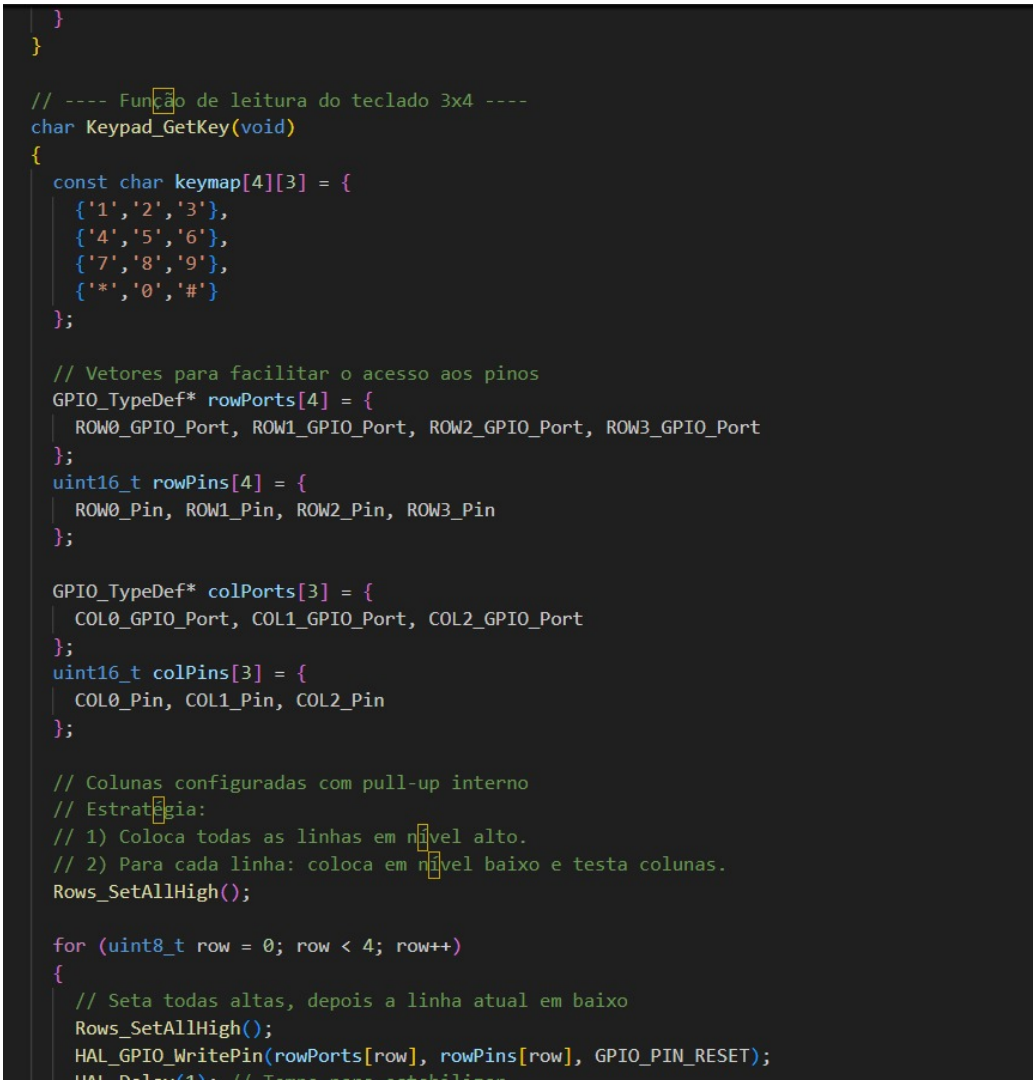


```c
#define PASSWORD_LENGTH  4

// Protótipos
char  Keypad_GetKey(void);
void  Lock_Open(void);
void  Lock_Error(void);
void  Rows_SetAllHigh(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  // Se houver display I2C, inicializar aqui (MX_I2C1_Init(), etc.)

  char buffer[PASSWORD_LENGTH + 1] = {0};
  uint8_t index = 0;

  while (1)
  {
    char key = Keypad_GetKey();

    if (key != 0)
    {
      // Tecla especial: '#' confirma a senha
      if (key == '#')
      {
        buffer[index] = '\0'; // Finaliza string

        if (strcmp(buffer, PASSWORD) == 0)
        {
          Lock_Open();
        }
        else
        {
          Lock_Error();
        }

        // Limpa buffer e reinicia contagem
        memset(buffer, 0, sizeof(buffer));
```

Figura 3.3:

```
{
  {
    {
      {
        }

        // Limpa buffer e reinicia contagem
        memset(buffer, 0, sizeof(buffer));
        index = 0;
      }
      // Tecla especial: '*' limpa a digitação
      else if (key == '*')
      {
        memset(buffer, 0, sizeof(buffer));
        index = 0;
      }
      // Dígitos normais '0'..'9'
      else
      {
        if (index < PASSWORD_LENGTH)
        {
          buffer[index++] = key;
          // Aqui poderia atualizar o display, se desejado
        }
      }

      // Aguarda soltar a tecla (anti-bouncing simples)
      HAL_Delay(150);
    }
  }
}

// ---- Função de leitura do teclado 3x4 ----
char Keypad_GetKey(void)
{
  const char keymap[4][3] = {
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'*','0','#'}
```

Figura 3.4:



```
  }
}

// ---- Função de leitura do teclado 3x4 ----
char Keypad_GetKey(void)
{
  const char keymap[4][3] = {
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'*','0','#'}
  };

  // Vetores para facilitar o acesso aos pinos
  GPIO_TypeDef* rowPorts[4] = {
    ROW0_GPIO_Port, ROW1_GPIO_Port, ROW2_GPIO_Port, ROW3_GPIO_Port
  };
  uint16_t rowPins[4] = {
    ROW0_Pin, ROW1_Pin, ROW2_Pin, ROW3_Pin
  };

  GPIO_TypeDef* colPorts[3] = {
    COL0_GPIO_Port, COL1_GPIO_Port, COL2_GPIO_Port
  };
  uint16_t colPins[3] = {
    COL0_Pin, COL1_Pin, COL2_Pin
  };

  // Colunas configuradas com pull-up interno
  // Estratégia:
  // 1) Coloca todas as linhas em nível alto.
  // 2) Para cada linha: coloca em nível baixo e testa colunas.
  Rows_SetAllHigh();

  for (uint8_t row = 0; row < 4; row++)
  {
    // Seta todas altas, depois a linha atual em baixo
    Rows_SetAllHigh();
    HAL_GPIO_WritePin(rowPorts[row], rowPins[row], GPIO_PIN_RESET);
```

Figura 3.5:

```
  }

  return 0; // Nenhuma tecla pressionada
}

void Rows_SetAllHigh(void)
{
  HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, GPIO_PIN_SET);
}

// ---- Acionamento da fechadura (relé) ----
void Lock_Open(void)
{
  // Desliga LED de erro (se estiver aceso)
  HAL_GPIO_WritePin(LED_ERR_GPIO_Port, LED_ERR_Pin, GPIO_PIN_RESET);

  // Aciona o relé (nível alto na base do transistor)
  HAL_GPIO_WritePin(RELAY_GPIO_Port, RELAY_Pin, GPIO_PIN_SET);

  // Mantém a trava aberta por alguns segundos
  HAL_Delay(3000);

  // Desaciona o relé
  HAL_GPIO_WritePin(RELAY_GPIO_Port, RELAY_Pin, GPIO_PIN_RESET);
}

// ---- Indicação de erro ----
void Lock_Error(void)
{
  // Acende LED vermelho por um tempo
  HAL_GPIO_WritePin(LED_ERR_GPIO_Port, LED_ERR_Pin, GPIO_PIN_SET);
  HAL_Delay(1000);
  HAL_GPIO_WritePin(LED_ERR_GPIO_Port, LED_ERR_Pin, GPIO_PIN_RESET);
}
```

Figura 3.6:

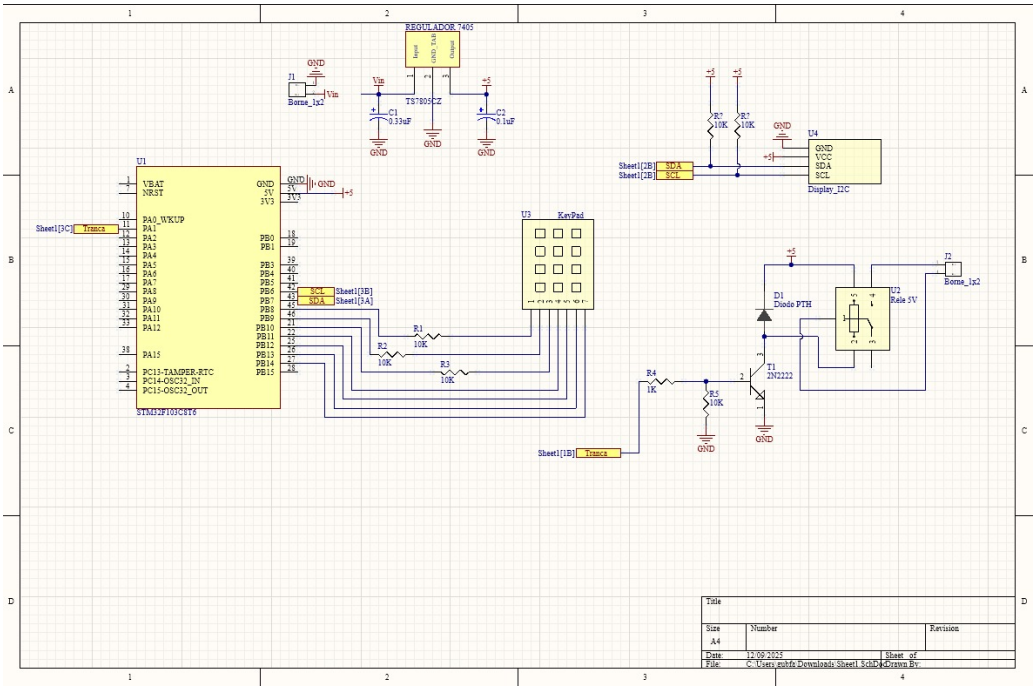

REGULADOR_7405
TS7805CZ
J1
Borne_1x2
Vin
C1
0.33uF
C2
0.1uF
GND
+5
U1
STM32F103C8T6
U3
KeyPad
U4
Display_I2C
R1
10K
R2
10K
R3
10K
R4
1K
R5
10K
R6
10K
R7
10K
D1
Diode PTH
U2
Rele 5V
T1
2N2222
J2
Borne_1x2
Sheet1[3C]
Trasca
Sheet1[3B]
Sheet1[3A]
Sheet1[2B]
SDA
SCL
Sheet1[1B]
Title
Size
A4
Number
Revision
Date:
12/09/2025
Sheet of
File:
C:\Users\subfz\Downloads\Sheet1.SchDoc
Drawn By:

Figura 3.7: