

Cofre biométrico reconfigurável.

um dispositivo capaz de guardar objetos com segurança e permitir acesso restrito às digitais cadastradas

Luiz Henrique Rocha Marinho

15/0041527

Engenharia eletrônica.

Universidade de Brasília

Brasília-DF, Brasil.

luizhenriquemarinhoFGA@gmail.com

Pedro Henrique Brito Checchia

15/0044488

Engenharia eletrônica.

Universidade de Brasília

Brasília-DF, Brasil

pedrobcb@gmail.com

Resumo —.

Este documento contém informações básicas sobre o projeto da disciplina de Microcontroladores e Microprocessadores. Este projeto consiste em um cofre com desbloqueio por impressão digital reconfigurável que tem como objetivo oferecer segurança de alta qualidade para que o usuário armazene objetos.

Palavras-chave — cofre; impressão digital;

I. REFERENCIAL TEÓRICO

O projeto se baseia na capacidade e precisão da leitura de um leitor de digitais (figura 1) que atua como sensor principal, a partir do reconhecimento da digital já cadastrada anteriormente, a trava elétrica (figura 2) é destravada automaticamente.



figura 1: Leitor biométrico

A partir de um teclado numérico de matriz 4x3(figura 6) e um display lcd (figura 7) , será possível reconfigurar os acessos. isso é possível graças a capacidade do leitor de armazenar e reconhecer até 200 digitas.[4]



Figura 2- trava elétrica solenóide, quando é sujeita a uma tensão de 12 V a trava é liberada.

A trava elétrica precisa de 12 V para funcionar, porém a MSP430 fornece uma saída de aproximadamente 3 V, portanto para o acionamento da trava será utilizada uma bateria de 12 V e um relé que será acionado a partir da MSP430 como mostra o esquema da figura 3.



Figura 7 -
mensagem de orientação 1.



Figura 8
- mensagem de orientação 2.



Figura 9 - mensagem de orientação 3

Esses são alguns exemplos de mensagens que serão mostradas

ao usuário, o Relé de 5V já foi implementado e o código do leitor já está em desenvolvimento a partir da comunicação serial UART half-duplex (figura 10) que utiliza as portas P1.1 e P1.2 como Tx (transmissor de dados) e Rx (Receptor de dados) [5]. Porém será preciso trabalhar com uma quantidade muito grande bytes (aproximadamente 16 bytes) [6]. Por isso será necessário mais tempo para que o leitor biométrico esteja funcionando por completo na msp430 a partir do Code Composer.

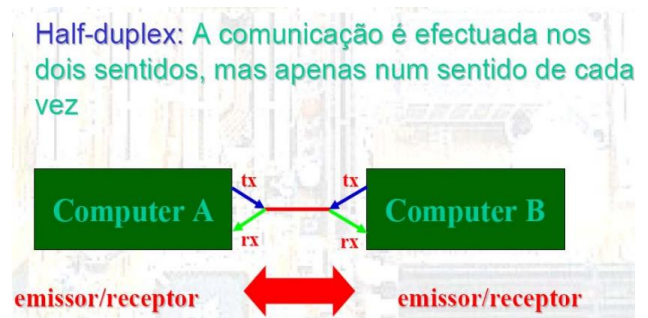


Figura 10 - comunicação serial fonte:
<http://www.eletrica.ufpr.br/>

III. OBJETIVOS

- Construir um cofre reconfigurável simples de se utilizar, com segurança, praticidade e confiabilidade para o usuário.

IV. BENEFÍCIOS

- Alta resistência a choques mecânicos
- Utilização do desbloqueio por impressão digital
- Interação com o usuário através de um display LCD
- Possibilidade de reconfiguração das digitais

V. REQUISITOS

Hardware:

- O projeto deverá contar com uma MSP430 G2553;
- leitor biométrico, teclado mecânico 3x4, display lcd, trava elétrica, baú de madeira, relé e alimentação de 12 V.
- Compreensão do funcionamento e características da comunicação serial para o leitor biométrico;
- A Dupla já possui todos os componentes e requisitos de hardware necessários.

Desempenho:

- o microcontrolador deve ter capacidade de armazenar um intervalo de 1 a 100 digitais, o número de digitais com acesso pode ser definido a partir do teclado e do display lcd;
- caso a digital esteja cadastrada, a trava deve receber 1 na saída (12V) para liberar o cofre. Tudo isso deve acontecer num intervalo de 1 a 5 segundos.[1]

Ambiente:

- o teclado, o leitor biométrico e o display devem ficar do lado de fora do cofre para o acesso do usuário, já o microprocessador, a trava elétrica, a alimentação e o relé devem ficar do lado de dentro para evitar violações.

Sistema:

a princípio, haverá uma digital cadastrada como a principal, para adicionar um novo usuário será preciso da autorização do dono da digital principal e todas as outras configurações também dependem disso. O teclado irá fornecer uma gama de opções que aparecerão no leitor lcd, porém as ordens só serão executadas quando o leitor verificar a digital do usuário principal, concedendo permissão.

VI REFERÊNCIAS

- [1]<https://www.filipeflop.com/blog/acionando-trava-eletrica-com-rfid/>
- [2]<http://blog.usinainfo.com.br/leitor-biometrico-arduino-sistema-de-cadastramento-e-leitura-de-digitais/>
- [3]<http://www.instructables.com/id/16x2-LCD-interfacing-in-4-bit-mode/>
- [4]<https://cdn-learn.adafruit.com/downloads/pdf/adafruit-optical-fingerprint-sensor.pdf>
- [5]<http://www.eletrica.ufpr.br/~rogerio/MSP430/00%20-%20CD%20DO%20ALUNO%20-%20FRAM/04%20-%20APOSTILAS/APOSTILA%20MSP430%20-%20C%20-%20PARTE%20IV.pdf>
- [6]ZFM-20_Fingerprint_Module.pdf

VII ANEXOS

1) CÓDIGO COMPLETO DISPLAY LCD E TECLADO NUMÉRICO

```

17#include <msp430g2553.h>
18
19
20#define lcd_port      P1OUT
21#define lcd_port_dir  P1DIR
22
23#define LCD_EN        BIT4
24#define LCD_RS        BIT5
25
26#define LED_DLY 100000    // VaLOR PARA SER DECREMENTADO, APROXIMADAMENTE 1 SEGUNDO
27#define LED1 BIT0
28#define SAIDA1 BIT1
29#define SAIDA2 BIT2
30#define SAIDA3 BIT3
31#define SAIDAS (SAIDA1 + SAIDA2 + SAIDA3)
32#define ENTRADA1 BIT4
33#define ENTRADA2 BIT5
34//#define ENTRADA3 BIT6
35#define ENTRRS (ENTRADA1 + ENTRADA2)
36
37void lcd_reset()
38{
39    lcd_port_dir = 0xff;
40    lcd_port = 0xff;
41    __delay_cycles(20000);
42    lcd_port = 0x03+LCD_EN;
43    lcd_port = 0x03;
44    __delay_cycles(10000);
45    lcd_port = 0x03+LCD_EN;

```

Updates Available

Updates are available for your software

```

46    lcd_port = 0x03;
47    __delay_cycles(1000);
48    lcd_port = 0x03+LCD_EN;
49    lcd_port = 0x03;
50    __delay_cycles(1000);
51    lcd_port = 0x02+LCD_EN;
52    lcd_port = 0x02;
53    __delay_cycles(1000);
54}
55
56void lcd_cmd (char cmd)
57{
58    // Send upper nibble
59    lcd_port = ((cmd >> 4) & 0x0F)|LCD_EN;
60    lcd_port = ((cmd >> 4) & 0x0F);
61
62    // Send lower nibble
63    lcd_port = (cmd & 0x0F)|LCD_EN;
64    lcd_port = (cmd & 0x0F);
65
66    __delay_cycles(4000);
67}
68
69void lcd_init ()
70{
71    lcd_reset();           // Call LCD reset
72    lcd_cmd(0x28);         // 4-bit mode - 2 line - 5x7 font.
73    lcd_cmd(0x0C);         // Display no cursor - no blink.
74    lcd_cmd(0x06);         // Automatic Increment - No Display shift.

```



```

75 lcd_cmd(0x80);    // Address DDRAM with 0 offset 80h.
76 lcd_cmd(0x01);    // Clear screen
77 }
78
79
80 void lcd_data (unsigned char dat)
81 {
82     // Send upper nibble
83     lcd_port = (((dat >> 4) & 0x0F) | LCD_EN | LCD_RS);
84     lcd_port = (((dat >> 4) & 0x0F) | LCD_RS);
85
86     // Send lower nibble
87     lcd_port = ((dat & 0x0F) | LCD_EN | LCD_RS);
88     lcd_port = ((dat & 0x0F) | LCD_RS);
89
90     _delay_cycles(4000); // a small delay may result in missing char display
91 }
92
93 void display_line(char *line)
94 {
95     while (*line)
96         lcd_data(*line++);
97 }
98
99
100 volatile unsigned int i, botao, linha;
101
102 void delay( volatile unsigned long int t)
103 {

```

```

102 void delay( volatile unsigned long int t)
103 {
104     while(t--);
105 }
106
107 void led_acende ()
108 {
109     while(1)
110     {
111         P2OUT |= LED1;
112         break;
113     }
114 }
115
116 void led_apaga()
117 {
118     while(1)
119     {
120         P2OUT &= ~LED1;
121         break;
122     }
123 }
124
125 void pisca(volatile unsigned int botao)
126 {
127
128     while(1)
129     {
130         for(i=botao; i>0; i--){

```

```

Getting Started  main.c  *main.c
130     for(i=botao; i>0; i--){
131         led_acende ();
132         //delay(1);
133
134         delay(LED_DLY);
135
136         led_apaga();
137
138         delay(LED_DLY);
139         //delay(1);
140     }
141
142     break;
143 }
144 }
145 }
146
147 void call_apaga ()
148 {
149     while (1)
150     {
151         // Stop Watch Dog Timer
152         //WDTCTL = WDTPW + WDTHOLD;
153
154         // Initialize LCD
155         lcd_init();
156         lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
157         display_line("");
158         lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40

```

```
Getting Started | main.c | *main.c
159     display_line("");
160     delay(LED_DLY);
161     break;
162 }
163 }
164
165 void call_lcd_principal ()
166 {
167     while (1)
168     {
169         // Stop Watch Dog Timer
170         // WDTCTL = WDTPW + WDTHOLD;
171
172         // Initialize LCD
173         lcd_init();
174         lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
175         display_line("1-mudar senha");
176         lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
177         display_line("2-novo usuario");
178         delay(LED_DLY);
179         // call_apaga();
180         break;
181     }
182 }
183
184 void call_lcd_1 ()
185 {
186     while (1)
187     {
188         // Stop Watch Dog Timer
189         // WDTCTL = WDTPW + WDTHOLD;
190
191         // Initialize LCD
192         lcd_init();
193         lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
194         display_line("coloque o dedo");
195         lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
196         display_line("cadastrado");
197         delay(LED_DLY);
198         // call_apaga();
199         break;
200     }
201 }
202
203 void call_lcd_2 ()
204 {
205     while (1)
206     {
207         // Stop Watch Dog Timer
208         // WDTCTL = WDTPW + WDTHOLD;
209
210         // Initialize LCD
211         lcd_init();
212         lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
213         display_line("quantos usuarios?");
214         lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
215         display_line("3- um , 4- dois");
216         delay(LED_DLY);
217         // call_apaga();
218         break;
219     }
220 }
```

```
189     lcd_init();
190     lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
191     display_line("coloque o dedo");
192     lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
193     display_line("cadastrado");
194     delay(LED_DLY);
195     // call_apaga();
196     break;
197 }
198 }
199
200 void call_lcd_2 ()
201 {
202     while (1)
203     {
204         // Stop Watch Dog Timer
205         // WDTCTL = WDTPW + WDTHOLD;
206
207         // Initialize LCD
208         lcd_init();
209         lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
210         display_line("quantos usuarios?");
211         lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
212         display_line("3- um , 4- dois");
213         delay(LED_DLY);
214         // call_apaga();
215         break;
216     }
217 }
```

```

220
221 int main(void){
222
223
224     int atual;
225     WDTCTL = WDTHOLD|WDTPW;
226     P2OUT |= ENTRS;
227     P2REN |= ENTRS;
228     P2OUT &= ~LED1;
229     P2OUT |= SAIDAS;
230     P2DIR |= LED1 + SAIDAS;
231     P2DIR &= ~(ENTRS);
232
233     call_lcd_principal();
234
235     while(1)
236     {
237         P2OUT |= SAIDAS;
238         P2OUT &= ~SAIDA1;
239         atual = (P2IN & ENTRS);
240         if(atual!= ENTRS)
241         {
242             if (atual == ENTRADA2)
243             {
244                 call_lcd_1();
245             }
246             else if (atual == ENTRADA1)
247             {
248                 pisca(4);
249             }

```

```

251     }
252     P2OUT |= SAIDAS;
253     P2OUT &= ~SAIDA2;
254     atual = (P2IN & ENTRS);
255     if(atual!= ENTRS)
256     {
257         if (atual == ENTRADA2 )
258             call_lcd_2();
259         else if (atual == ENTRADA1)
260             pisca(5);
261         //else if(atual==(ENTRADA1 + ENTRADA2))
262             // pisca(8);
263     }
264     P2OUT |= SAIDAS;
265     P2OUT &= ~SAIDA3;
266     atual = (P2IN & ENTRS);
267     if(atual!= ENTRS)
268     {
269         if (atual == ENTRADA2)
270             pisca(3);
271         else if (atual == ENTRADA1)
272             pisca(6);
273         //else if(atual==(ENTRADA1 + ENTRADA2))
274             // pisca(9);
275     }
276 }
277 return 0;
278 }
279

```