Cofre digital.

Um dispositivo capaz de guardar objetos com segurança e permitir acesso restrito ao uso de um ibutton

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
pedrobcbr@hotmail.com

Resumo —.

Este documento contém informações básicas sobre o projeto da disciplina de Microcontroladores e Microprocessadores. Este projeto consiste em um cofre com desbloqueio com ibutton 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 do ibutton (figura 1) que atua como sensor principal, a partir do reconhecimento de um código próprio garantido pelo fabricante. A trava elétrica (figura 2) é destravada automaticamente.

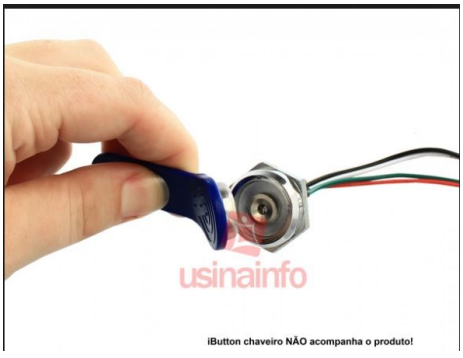

figura 1: Ibutton

A partir de um teclado numérico de matriz 4x3(figura 6) e um display lcd (figura 7).

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.

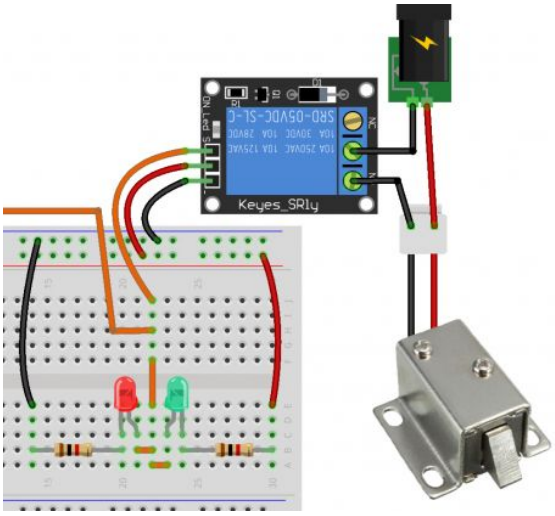

Figura 3 - Ilustração da alimentação da trava(fonte:https://www.filipeflop.com/blog/acionando-trava-eletrica-com-rfid/)

.

O cofre é feito de madeira de alta qualidade que garante a proteção dos bens que estaram do lado de dentro, como mostra as figuras 4 e 5.

Figura 4 - cofre aberto.

Figura 5 - cofre fechado.

II. JUSTIFICATIVA

- O motivo da realização deste projeto é oferecer a máxima segurança para armazenar objetos dos mais variados tipos, visto que o sistema de desbloqueio por ibutton é um meio bastante eficaz e seguro [4]. Para executar comandos como cadastrar, remover, entre outros, existe no produto um teclado numérico e um display LCD que exibe as informações:

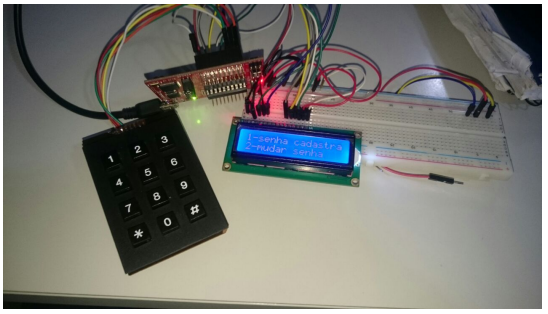

Figura 6 - funcionamento simultâneo do display e do teclado.

Foi utilizado o Software Code Composer Studio 7.4.0 para executar o código que se encontra nos anexos no final do documento. Basicamente foram criadas funções para printar no display a informação a partir do botão que foi pressionado, desenvolvendo assim a interface do projeto:

Figura 7 - mensagem de orientação 1.

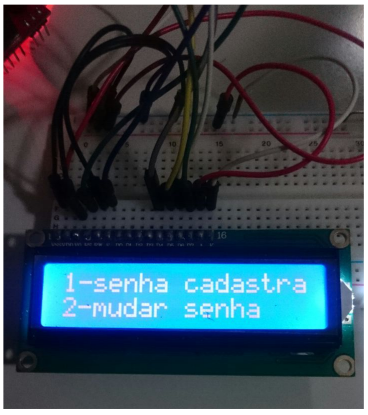

Figura 8 - mensagem de orientação 2.

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 ibutton está em desenvolvimento a partir da comunicação serial UART half-duplex (figura 10) que utiliza a porta P1.1 como o Rx (Receptor de dados) [1]. Será preciso trabalhar com uma quantidade grande bytes (aproximadamente 8 bytes) que estão agrupados em dígitos hexadecimais que definem a senha do ibutton[6].

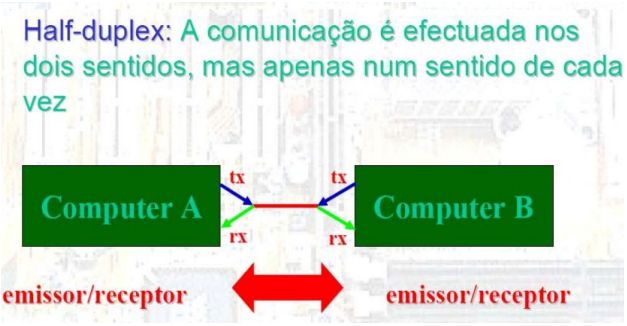

Figura 10 - comunicação serial fonte: http://www.eletrica.ufpr.br/

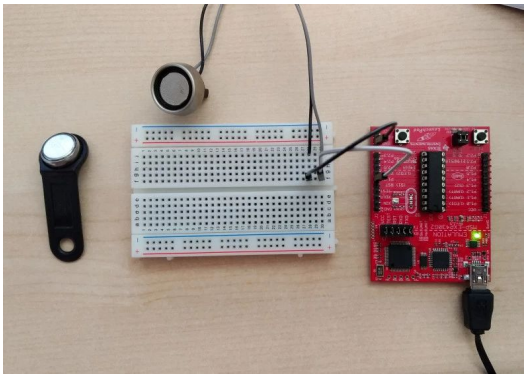

Figura 11 - ibutton e leitor conectados a msp430.

A partir do RXBUF, que é o comando utilizado para ler(recebe) a partir do pino P1_1 e armazenar bit por bit em uma comunicação serial UART, os bits que estão no ibutton são armazenados na variável char hs_i [ ] de 8 posições.

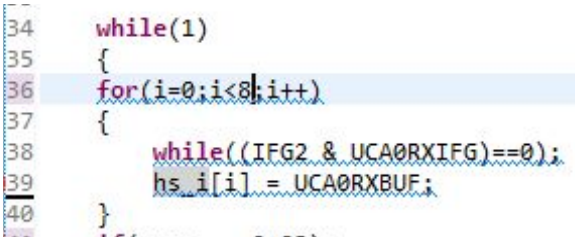

Figura 12 - armazenamento dos bits em hexa na variável hs_i [ ].

III. Objetivos

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

IV. Benefícios

- Alta resistência a choques mecânicos
- Utilização do desbloqueio por ibutton

- Interação com o usuário através de um display LCD
- Possibilidade de reconfiguração das chaves

V. Requisitos

Hardware:

-O projeto deverá contar com uma MSP430 G2553;
-ibutton, 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 Ibutton;

Desempenho:

-o microcontrolador deve ter capacidade de armazenar um intervalo de 64 bits que é a senha do ibutton, o número de acessos pode ser definido a partir da quantidade de ibuttons disponíveis com o auxílio do teclado e do display lcd;
-caso o ibutton esteja cadastrado, 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 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 fica 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 todos 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]http://www.cerne-tec.com.br/Artigo_07_ComunicacaoSerial.pdf

[2]https://www.up.edu.br/blogs/engenharia-da-computacao/wp-content/uploads/sites/6/2015/06/2003.5.pdf

[3]http://www.instructables.com/id/16x2-LCD-interfacing-in-4-bit-mode/

[4]https://datasheets.maximintegrated.com/en/ds/DS1990A.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

VII anexos

1) Código completo Display lcd e teclado numérico

```c
#include <msp430g2553.h>


#define lcd_port        P1OUT
#define lcd_port_dir    P1DIR

#define LCD_EN      BIT4
#define LCD_RS      BIT5

#define LED_DLY 100000      // VaLOR PARA SER DECREMENTADO, APROXIMADAMENTE 1 SEGUNDO
#define LED1 BIT0
#define SAIDA1 BIT1
#define SAIDA2 BIT2
#define SAIDA3 BIT3
#define SAIDAS (SAIDA1 + SAIDA2 + SAIDA3)
#define ENTRADA1 BIT4
#define ENTRADA2 BIT5
//#define ENTRADA3 BIT6
#define ENTRS (ENTRADA1 + ENTRADA2)

void lcd_reset()
{
    lcd_port_dir = 0xff;
    lcd_port = 0xff;
    __delay_cycles(20000);
    lcd_port = 0x03+LCD_EN;
    lcd_port = 0x03;
    __delay_cycles(10000);
    lcd_port = 0x03+LCD_EN;
```

Updates Available

Updates are available for your softwa

Getting Started | main.c | *main.c

```c
    lcd_port = 0x03;
    __delay_cycles(1000);
    lcd_port = 0x03+LCD_EN;
    lcd_port = 0x03;
    __delay_cycles(1000);
    lcd_port = 0x02+LCD_EN;
    lcd_port = 0x02;
    __delay_cycles(1000);
}

void lcd_cmd (char cmd)
{
    // Send upper nibble
    lcd_port = ((cmd >> 4) & 0x0F)|LCD_EN;
    lcd_port = ((cmd >> 4) & 0x0F);

    // Send lower nibble
    lcd_port = (cmd & 0x0F)|LCD_EN;
    lcd_port = (cmd & 0x0F);

    __delay_cycles(4000);
}

void lcd_init ()
{
    lcd_reset();         // Call LCD reset
    lcd_cmd(0x28);       // 4-bit mode - 2 line - 5x7 font.
    lcd_cmd(0x0C);       // Display no cursor - no blink.
    lcd_cmd(0x06);       // Automatic Increment - No Display shift.
```

```
    lcd_cmd(0x80);       // Address DDRAM with 0 offset 80h.
    lcd_cmd(0x01);       // Clear screen
}


void lcd_data (unsigned char dat)
{
    // Send upper nibble
    lcd_port = (((dat >> 4) & 0x0F)|LCD_EN|LCD_RS);
    lcd_port = (((dat >> 4) & 0x0F)|LCD_RS);

    // Send lower nibble
    lcd_port = ((dat & 0x0F)|LCD_EN|LCD_RS);
    lcd_port = ((dat & 0x0F)|LCD_RS);

    __delay_cycles(4000); // a small delay may result in missing char display
}

void display_line(char *line)
{
    while (*line)
        lcd_data(*line++);
}


volatile unsigned int i, botao, linha;

void delay( volatile unsigned long int t)
{
```

```
void delay( volatile unsigned long int t)
{
     while(t--);
}

void led_acende ()
{
     while(1)
     {
         P2OUT|= LED1;
         break;
     }
}

void led_apaga()
{
    while(1)
    {
        P2OUT&=~LED1;
        break;
    }
 }

void pisca(volatile unsigned int botao)
{

     while(1)
    {
    for(i=botao; i>0; i--){
```

Getting Started | main.c | *main.c

```
    for(i=botao; i>0; i--){
         led_acende ();
         //delay(1);

         delay(LED_DLY);

         led_apaga();

         delay(LED_DLY);
          //delay(1);
       }

     break;

     }
   }

void call_apaga ()
{
    while (1)
    {
    // Stop Watch Dog Timer
    //WDTCTL = WDTPW + WDTHOLD;

    // Initialize LCD
    lcd_init();
    lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
    display_line("");
    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     {
```

```
189     lcd_init();
190     lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
191     display_line("cologue 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
```