Tranca Eletrônica

Vivian Emanoelle Marinho Gomes
Igor da Silva Zagonel

IFMT – MT

igorzagonel@gmail.com
vivianemanoellegomes@gmail.com

Resumo **— Este relatório tem como objetivo familiarizar os alunos com as montagens e programações dos microcontroladores nas realizações de tarefas**.

I. Introdução

O objetivo do projeto é criar uma tranca eletrônica utilizando o PIC 16F877A. A fechadura receberá três tipos de dados o id, senha e status, mas será aberta apenas quando o usuário inserir a senha correta, o administrador também pode fazer alterações no banco de dados como inserir, deletar e buscar alguém dado. Realizamos também a criação de uma interface gráfica usando C#, para ficar visível para o usuário e facilitar na hora de inserir os dados necessários. Para realizar esse projeto utilizamos o PIC 16F877A, LCD, memória externa EEPROM.

II. A Teoria

- PIC 16F877A

O PIC 16F877A é um microcontrolador da família de 8 bits, ou seja, lê apenas palavras de no máximo 8 bits e possui núcleo de 14 bits. A letra F indica que a memória programada desse PIC é do tipo Flash, cada linha da memória é uma palavra de 14 bits. Os três números finais do PIC permitem identifica-lo precisamente. A frequência de operação (clock) vai até 20MHz, memória RAM com 368 bytes e a memória EEPROM com 256 bytes, pode funcionar em uma alimentação de 2V a 5,5V e sua pinagem possui 40 pinos.

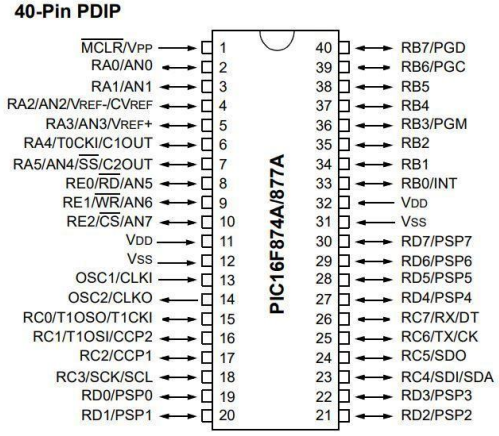

Fig. 1 – PIC 16F877A

- LCD

Um display de cristal líquido, é um painel utilizado para exibir informações por via eletrônica, como texto, imagens e vídeos. Seu uso inclui monitores, televisores, painéis e outros dispositivos.

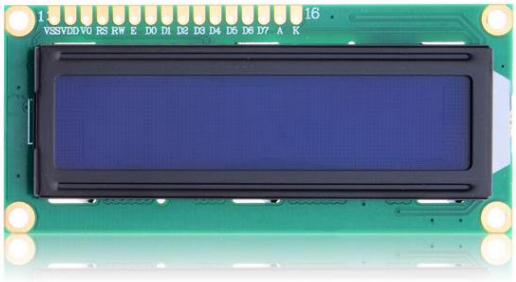

Fig. 2 – LCD 16x2

- EEPROM

A EEPROM significa memória somente de leitura programável apagável eletricamente. Ela pode ser apaga e reescrita via eletricidade e não por luz ultravioleta. Essa memória também pode ser parcialmente apagada, podendo ser alterar apenas alguns dados. Uma de suas vantagens é que não é necessário retira-lo do circuito para ser limpa ou reprogramada. A memória flash é um tipo de EEPROM.

Fig. 3 – EEPROM

- C#

O SharpDevelop é um ambiente de desenvolvimento de código. Essa ferramenta possui características interessantes, como ter uma IDE leve, autocomplete e chaves inteligentes.

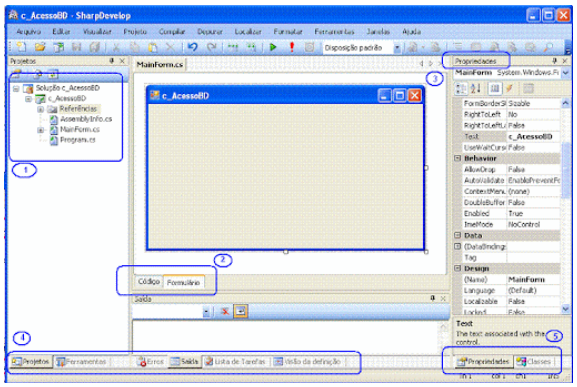

Fig. 4 – IDE SharpDevelop

- SQLite

É uma base de dados relacional de código aberto e que dispensa o uso de um servidor de sua atuação. Armazenando seus arquivos dentro de sua própria estrutura, funciona muito bem para diversas aplicações, principalmente websites e sistema mobile. O SQLite é uma biblioteca em linguagem C.

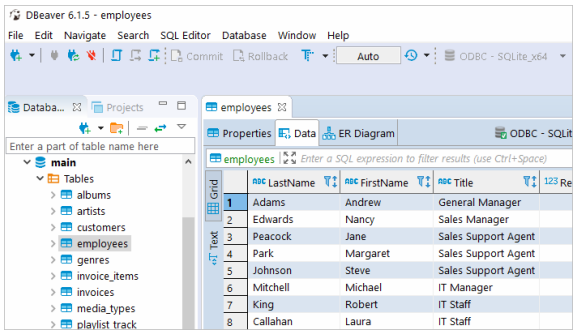

Fig. 5 – IDE SQLite

- Tranca Eletrônica

A fechadura eletrônica é uma ferramenta inteligente, evoluída para conseguir proporcionar mais segurança a um ambiente. Além de facilitar os transtornos de esquecer a chave ou perder. É uma ferramenta onde é possível abrir portas externas por meio de senhas, tag de proximidade, biometria.

Fig. 5 – Fechadura Eletrônica

III. Programação e Circuito

A montagem realizada nesse projeto consiste em um microcontrolador, o objetivo do mesmo era receber dois sinais diferentes enviados por potenciômetros e fazer a plotagem no gráfico.

- Programas
  - C#

Fazer o carregamento dos dados que estavam no banco de dados e inserindo na tabela para mostrar no Grid.

```
private void LoadData()
{
    SetConnection();
    sql_con.Open();

    sql_cmd = sql_con.CreateCommand();
    string CommandText = "select id, senha,status from  clientes";
    DB = new SQLiteDataAdapter(CommandText,sql_con);
    DS.Reset();
    DB.Fill(DS);
    DT= DS.Tables[0];
    Grid.DataSource = DT;
    sql_con.Close();

}
```

Fig. 5 – Código SharDevelop

A função para fazer a conexão com o banco de dados e a função para executar as instruções do banco.

```
private void SetConnection()
{
    sql_con = new SQLiteConnection("Data Source=clientes.db;Version=3;New=False;Compress=True;");
}
private void ExecuteQuery(string txtQuery)
{
    SetConnection();
    sql_con.Open();

    sql_cmd = sql_con.CreateCommand();
    sql_cmd.CommandText=txtQuery;

    sql_cmd.ExecuteNonQuery();
    sql_con.Close();
}
void MainFormLoad(object sender, EventArgs e)
{
    LoadData();
}
```

Fig. 6 – Código SharDevelop

Direcionando o botão adicionar para a função que foi criada logo abaixo e adiciona os dados no banco.

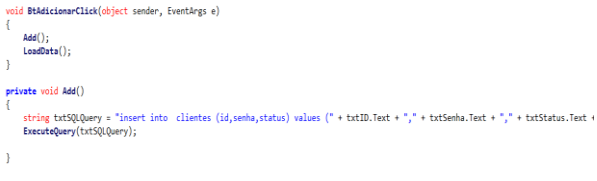

Fig. 7 – Código SharDevelop

Direcionando o botão excluir para a função que foi criada logo abaixo que exclui os dados no banco.

```
void BtExcluirClick(object sender, EventArgs e)
{
    Delete();
    LoadData();
}
private void Delete()
{
    string txtSQLQuery = "delete from  clientes where id ="+txtID.Text;
    ExecuteQuery(txtSQLQuery);
    txtID.Text = string.Empty;
    txtSenha.Text = string.Empty;
    txtStatus.Text = string.Empty;
    LoadData();
}
```

Fig. 8 – Código SharDevelop

Direcionando o botão de modificação para a função que foi criada logo abaixo que modifica os dados no banco.

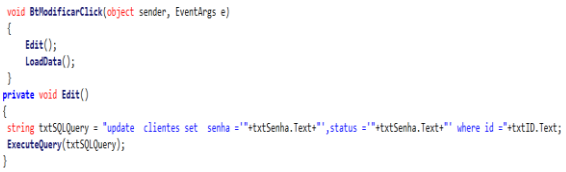

Fig. 9 – Código SharDevelop

Fazendo a comunicação Serial com o C#

```
private void atualizaListaCOMs()
{
   int i;
    bool quantDiferente;    //flag para sinalizar que a quantidade de portas mudou

    i = 0;
    quantDiferente = false;

    //se a quantidade de portas mudou
    if (comboBox1.Items.Count == SerialPort.GetPortNames().Length)
    {
        foreach (string s in SerialPort.GetPortNames())
        {
            if (comboBox1.Items[i++].Equals(s) == false)
            {
                quantDiferente = true;
            }
        }
    }
    else
    {
        quantDiferente = true;
    }

    //Se não foi detectado diferença
    if (quantDiferente == false)
```

Fig. 10 – Código SharDevelop

Adicionamos um timer para dar o clock conforme a atualização das portas COMs.

```
void Timer1Tick(object sender, EventArgs e)
{
      atualizaListaCOMs();
}
```

Fig. 11 – Código SharDevelop

Criamos uma função para o botão conectar que muda de estado quando a porta está acionada.

```
void BtnConectarClick(object sender, EventArgs e)
{
     if (serialPort1.IsOpen == false)
    {
        try
        {
            serialPort1.PortName = comboBox1.Items[comboBox1.SelectedIndex].ToString();
            serialPort1.Open();

        }
        catch
        {
            return;

        }
        if (serialPort1.IsOpen)
        {
            btnConectar.Text = "Desconectar";
            comboBox1.Enabled = false;

        }
    }
    else
    {
        try
```

Fig. 12 – Código SharDevelop

- Códigos PCW

A primeira parte está algumas funções que foram criadas para a execução do projeto.

```
void imprimeCliente();
void recebeClienteBanco();
void cadastraCliente();
void deletarCliente();
void menuPrincipalADM();
void menuPrincipalUser();
void apagarMemoria();
void manutencao();
void atualizarBancoSQL();
void atualizaBancoPIC();
```

Fig. 13 – Funções criadas

A segunda parte foi feita a estrutura das nossas configurações e a criação de algumas variáveis.

```
typedef struct {
  char id[2];
  char senha[4];
  char status;
}
Cliente;

Cliente pessoa;
Cliente Adminstrador;
int flagAdm = 1;
int flagMenuAdm = 0;
char tecla;
int ja = 0;
```

Fig. 14 – Código

No main estão sendo chamadas as funções que podem ser executadas. No menu do usuário é pedido para inserir o ID e Senha.

```
void menuPrincipalUser() {
  int aux = 0, k = 0, f = 0, y = 0;
  char aux1;
  char tc1;
  int local;
  int positionFindId = 0;
  Cliente cl;

  printf(lcd_escreve, "\fID:\n");
  while (aux < 2) {
```

Fig. 15 – Código

Assim que inserido, ocorre a busca na memória por esse dado. Quando reconhecido, a entrada é liberada. Se não, a entrada é negada.

```
while (k != 1) {
  local = positionFindId;
  if (read_ext_eeprom(local) == cl.id[0]) {
    if (read_ext_eeprom(local + 1) == cl.id[1]) {
      if (read_ext_eeprom(local + 2) == cl.senha[0]) {
        if (read_ext_eeprom(local + 3) == cl.senha[1]) {
          if (read_ext_eeprom(local + 4) == cl.senha[2]) {
            if (read_ext_eeprom(local + 5) == cl.senha[3]) {
              if (read_ext_eeprom(local + 6) == '1') {
                printf(lcd_escreve, "\fBEM VINDO\n");
                delay_ms(1500);
                k = 1; //aqui eu acendo o LED
              } else {
                printf(lcd_escreve, "\fINVALIDO\n");
                delay_ms(1500);
                k = 1;
```

Fig. 16 – Código

A função manutenção é apenas acessada pelo administrados. E pede para inserir a opção desejada.

```
void manutencao() {
  int f = 0;
  int y = 0;
  char aux1;
  printf(lcd_escreve, "\fDIGITE OPCAO: ");
```

Fig. 17 – Código

Entre as opções, estão:

```
case '1':
  imprimeCliente();
  break;
case '2':
  cadastraCliente();
  break;
case '3':
  deletarCliente(); //como deletar o cliente da memoria
  break;
case '4':
  flagMenuAdm = 0;
  break;
case '5':
  //recebe do banco de dados via serial.
  break;
case '6':
  atualizarBancoSQL();
  break;
```

Fig. 18 – Código

A função deletar cliente espera a entrada de um ID específico, busca na memória quando encontrado apaga as 7 posições sequenciais.

```
while (k != 1) {
  local = positionFindId;
  if (read_ext_eeprom(local) == teste.id[0]) {
    if (read_ext_eeprom(local + 1) == teste.id[1]) {
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      write_ext_eeprom(local, 0xff);
      local++;
      printf(lcd_escreve, "\fID APAGADO");
      while (f != 1) {
        tecla = tc_tecla(1500);
        if (tecla == '#') {
          f = 1;
```

Fig. 19 – Código

A função imprimir cliente espera a entrada de um ID específico, busca na memória quando encontrado imprime as 7 posições sequenciais e escreve no LCD.

```
while (k != 1) {
  local = positionFindId;
  if (read_ext_eeprom(local) == teste.id[0]) {
    if (read_ext_eeprom(local + 1) == teste.id[1]) {
      pessoa.id[0] = read_ext_eeprom(local);
      local++;
      pessoa.id[1] = read_ext_eeprom(local);
      local++;
      pessoa.senha[0] = read_ext_eeprom(local);
      local++;
      pessoa.senha[1] = read_ext_eeprom(local);
      local++;
      pessoa.senha[2] = read_ext_eeprom(local);
      local++;
      pessoa.senha[3] = read_ext_eeprom(local);
      local++;
      pessoa.status = read_ext_eeprom(local);
      local++;
      printf(lcd_escreve, "\fID: %c%c Stats: %c", pessoa.id[0], pessoa.id[1], pessoa.status);
      printf(lcd_escreve, "\nSenha: %c%c%c%c", pessoa.senha[0], pessoa.senha[1], pessoa.senha[2], pessoa.senha[3]);
      while (f != 1) {
        tecla = tc_tecla(1500);
        if (tecla == '#') {
          f = 1;
        }
```

Fig. 20 – Código

Na função cadastrar clientes, o primeiro cadastro realizado é do administrador. Somente depois que começa a cadastrar os clientes, pede para inserir ID, Senha e Status busca na memória um espaço vazio, se encontrar salva esse dado na memória. A memória tem um espaço de 512, porém como precisamos de um espaço de 7 posições para cada cadastro, é possível cadastrar apenas 73 pessoas na memória.

```
while (cont < 73) {
  if (read_ext_eeprom(lastMemoryPosition) == 0xff) {

    write_ext_eeprom(lastMemoryPosition, pessoa.id[0]);
    lastMemoryPosition++;
    write_ext_eeprom(lastMemoryPosition, pessoa.id[1]);
    lastMemoryPosition++;
    write_ext_eeprom(lastMemoryPosition, pessoa.senha[0]);
    lastMemoryPosition++;
    write_ext_eeprom(lastMemoryPosition, pessoa.senha[1]);
    lastMemoryPosition++;
    write_ext_eeprom(lastMemoryPosition, pessoa.senha[2]);
    lastMemoryPosition++;
    write_ext_eeprom(lastMemoryPosition, pessoa.senha[3]);
    lastMemoryPosition++;
    write_ext_eeprom(lastMemoryPosition, pessoa.status);
    lastMemoryPosition++;
    cont = 75;
  } else {
    lastMemoryPosition += 7;
  }
  cont++;
}
printf(lcd_escreve, "\fCadastrado");
delay_ms(1500);
```

Fig. 21 – Código

IV. COMENTÁRIOS E CONCLUSÕES

Com esse projeto concluímos que é possível construir uma tranca eletrónica utilizando o PIC em uma linguagem simples. Ainda foi realizado uma interface gráfica básica na qual se pode realizar os comandos de cadastrar, alterar, excluir e buscar, fazer a conexão com o banco de dados (SQLite). Fazendo também a comunicação serial, mostrando quais portas estariam disponíveis para ser utilizadas. A tranca apenas será aberta depois de passar pelo processo de busca das informações no banco de dados e obtendo a verificação que está correto conforme aquilo que foi solicitado.

V.REFERÊNCIAS

[1] Microcontrolandos, “Tranca Eletrônica”, Disponível em < httpshttp://microcontrolandos.blogspot.com/2013/01/tranca-eletronica.html>

[2] Márcio Althmann, “SharDevelop4”, Disponível em < httpshttps://tecnoblog.net/meiobit/79311/sharpdevelop-4-otima-ide-gratuita-para-desenvolvedores-net/>