Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

Engenharia Eletrotécnica e das Telecomunicações

Relatório do Miniprojecto

Sistema de Caixa Multibanco

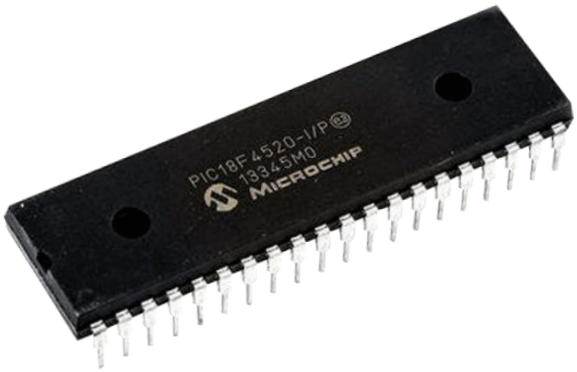
Microcontrolador e Instrumentação

3º Ano - 1º Semestre
Ano Letivo: 2024/2025

Docente: Prof. Dr. José Vieira

Ivandro Andrade Nº 20181669

Chaima Belgacem Nº 20241168



Janeiro 2025

Índice

Índice de figuras

Lista de Tabelas

INTRODUÇÃO

Perante a disciplina de Microcontroladores e Instrumentação, foi-nos posto um desafio de realizar um trabalho que tem como objetivo fazer a simulação de uma caixa multibanco.

O projeto foi feito de modo a simular um cenário mais próximo da realidade, dando ao utilizador a possibilidade de consultar o seu saldo, levantar dinheiro, depositar e fazer transferências. E não só, contamos também com uma conta de administrador, aonde o mesmo pode criar ou deletar contas de usuários para além de consultar todos existentes.

Durante a realização deste projeto, foi utilizado a PIC Millennium Board PCB143 REV2.0 juntamente com o microcontrolador PIC18F4520, um teclado matricial, um LCD 16x4 e ainda o leitor de cartões magnéticos KDE KT-2250.

Para simular o projeto utilizamos o software Proteus 8 Professional ao lado do PIC C Compiler que foi a estrutura aonde desenvolvemos o código do mesmo.

ESTRUTURA HARDWARE

Microcontrolador PIC18F4520

Para a realização deste projeto utilizamos o microcontrolador PIC18F4520 que é um microcontrolador de 8 bits fabricado pela Microchip Technology, pertencente à família **PIC18F**, que é conhecida por sua alta performance e eficiência. Ele utiliza a arquitetura **RISC**, que é baseado em uma arquitetura de conjunto reduzido de instruções **(RISC)**, que proporciona uma execução rápida e eficiente.

O **PIC18F4520** tem uma estrutura de memórias constituída por:

- **Memória Flash:** 32 KB de memória de programa (permite regravação).
- **Memória RAM:** 1,5 KB para dados temporários.
- **Memória EEPROM:** 256 bytes para armazenamento de dados não voláteis.

O **PIC18F4520** utiliza a arquitetura Harvard que consiste em separar a memória de programa da memória de dados, permitindo que o microcontrolador acesse instruções e dados de maneira simultânea. Isso proporciona maior eficiência e desempenho em comparação à arquitetura von Neumann, que usa um único barramento para instruções e dados.

Conta também com um clock de até 40 MHz (10 MIPS - instruções por segundo). Porém como a board aonde o microcontrolador se encontra embutida, suporta uma frequência de até 20Mhz (o que torna a execução do programa mais lenta), foi preciso trabalhar nesta faixa de frequência.

A estrutura deste **microcontrolador** conta com uma série de periféricos integrados:

- **Portas de I/O:** 36 pinos configuráveis como entrada ou saída.
- **Conversor A/D:** 10 bits com até 13 canais, ideal para leitura de sinais analógicos.
- **PWM:** Módulos para controle de motores e LEDs.
- **Timers:** Vários temporizadores para controle preciso de eventos.
- **USART:** Comunicação serial (RS232).
- **I²C e SPI:** Protocolos de comunicação para integração com sensores e outros dispositivos.

O **PIC18F4520** também oferece algumas capacidades avançadas que o melhor desempenho, tais como:

- **Watchdog Timer (WDT):** Recurso para reiniciar o sistema em caso de falha.
- **Brown-out Reset:** Proteção contra queda de tensão.
- **Sleep Mode:** Economia de energia em aplicações de baixo consumo.

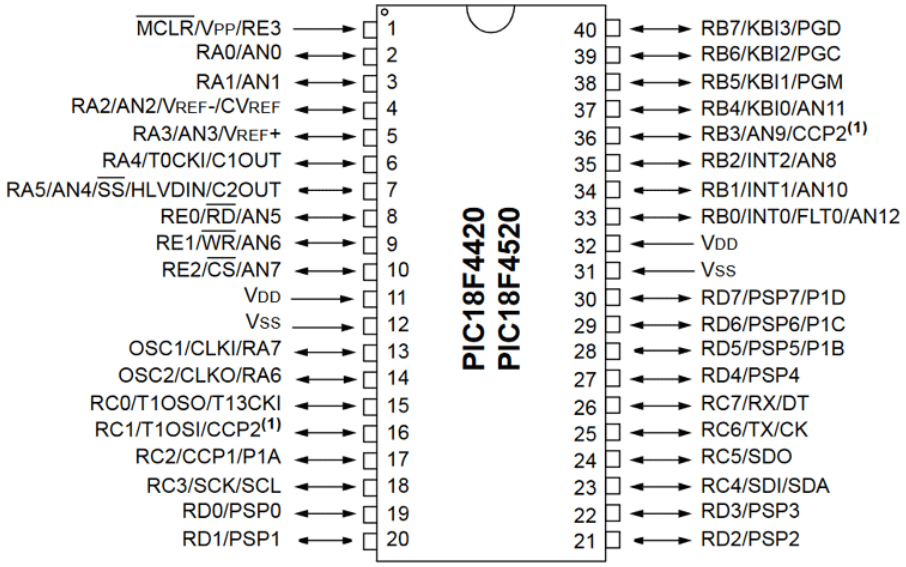


Figura 1 - PIC18F4520 Pinout

Acima temos um esquema dos pinos do PIC18F4520, o mesmo utilizado na resolução deste projeto. O teclado está ligado ao porto B e o LCD está ligado ao porto D. O ENABLE está ligado ao pino E1 e o RS está ligado ao pino E0. A leitura dos tramas recebidos através do leitor magnético são feitas através do pino C7.

LCD 16x4

Nós utilizamos um LCD de 16x4 HITACHI HD44780U já embutido na board, no qual possui 16 pinos ao todo. 4 pinos são dedicados para o controlo e 8 pinos são dedicados para os dados.

Número	Símbolo	Função
1	Vss	0v Power Supply (GND Level)
2	Vdd	Power Supply for Logic Circuit
3	Vo	Is for adjusting the contrast of the display. Usually, when this pin is grounded the pixels will be the darkest
4	RS	Data/Instruction select
5	R/W	Determines if we read from or write to the LCD
6	E	Enables or disables the LCD module
7-14	DB0-DB7	Bi-directional data bus

Tabela 1 - LCD pinout

Os pinos do porto D foram utilizados para os dados, os pinos E0 (RS) e E1(ENABLE) foram utilizados para o controlo.

Quando o pino E1 está “baixo”, o display se encontra desabilitado e os valores que se encontram nos pinos RS, R/W e no barramento de dados serão ignorados.

Quando o Pino E0 (RS) está “ativo” quer dizer que vamos enviar um caractere (data register), e quando ele está “baixo” quer dizer que vamos enviar uma instrução (instruction register).

Toda a escrita no LCD é feita segunda o código da tabela ASCII, tanto quanto a posição no LCD. Para determinarmos aonde queremos enviar esses caracteres (ASCII code), devemos levar em consideração os endereços DDRAM, que basicamente representa a posição do cursor no LCD. Abaixo são mostrados os endereços da DDRAM:

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Tabela 2 - Endereços DDRAM do LCD

Teclado

Para que o utilizador consiga inserir dados como password, montante a depositar e também interagir com multibanco, seria necessário um teclado. No nosso projeto aproveitamos o teclado matricial que já vinha inserido na board utilizada. Este teclado funciona a base de um sistema matricial que utiliza um conjunto de linhas e colunas para detetar qual tecla foi pressionada. Quando uma tecla é pressionada, ela conecta uma linha a uma coluna específica, criando um "curto-circuito" momentâneo entre elas.

Este teclado funciona a base de resistências de pull-up, que são basicamente usados para garantir que os pinos não fiquem "flutuando" (sem valor definido) quando nenhuma tecla está pressionada. Isso ocorre porque, na ausência de uma conexão direta com o Vcc (tensão de alimentação), a entrada pode captar ruídos do ambiente, levando a leituras incorretas.

No nosso projeto o teclado está conectado ao porto B que tem uma conexão com as resistências pull-ups da board. Um resistor de pull-up conecta o pino de entrada à **tensão de alimentação (Vcc)** através de um resistor. Quando não pressionado, todos as saídas estão no estado lógico **HIGH**, e a partir do momento que uma tecla é pressionada, o pino é conectado ao **GND** através da matriz do teclado, forçando o estado lógico para **LOW**. O programa faz o varrimento linha (16 posições, 4 pinos de entrada e 4 pinos de saída) por linha e assim deteta qual tecla foi pressionado através da mudança de estado.

Leitor magnético KDE KT-2250

No andamento do projeto, para fazer a leitura dos cartões bancários utilizamos o KDE KT-2250.

Ao passar o cartão pelo leitor magnético, ele captura um sinal analógico que varia conforme a polaridade dos dados no cartão. Após a conversão do sinal analógico para digital por meio de um **decoder** inserido no KDE KT-2250, esses dados são organizados no formato binário conforme o padrão da codificação da trama.

Os dados são organizados pelo firmware interno do leitor em pacotes ou strings legíveis para serem transferidos. O formato da informação é transcrito em códigos **ASCII**.

Após a leitura e formatação, o leitor transfere os dados para o **Microcontrolador PIC18F4520** através da comunicação serial **RS232**, que envia os dados em pacotes estruturados, geralmente iniciados e terminados por caracteres especiais para indicar o início e fim da leitura: **<STX>DATA<ETX>** (Onde **<STX>** indica o início e **<ETX>** o fim).

Na ligação serial **RS232**, o pino **RX** que é aonde vamos receber os dados, está ligado ao pino **C7** do porto **C**, e o pino **TX** está ligado ao ground que alimenta o leitor magnético com **12V**. Como apresentado na foto abaixo:

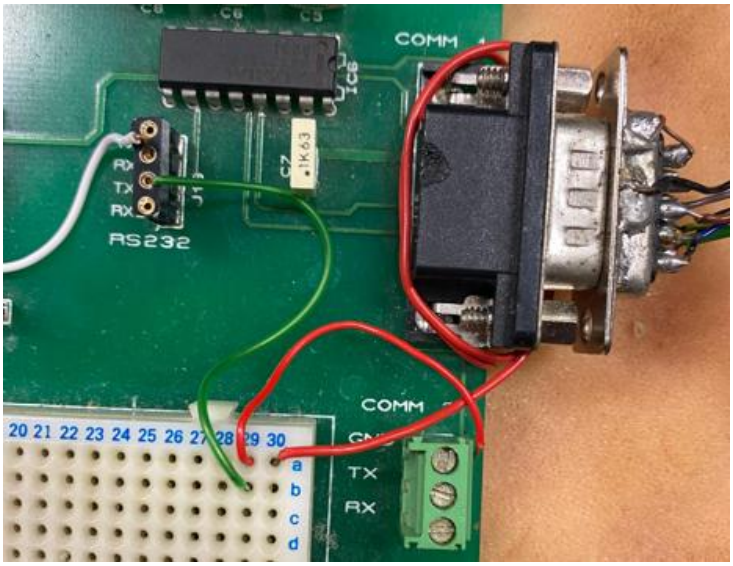


Figura 2 - Ligação serial RS232 do Leitor magnético KDE KT-2250

SOFTWARE

O software foi desenvolvido de forma a permitir a melhor interligação entre o LCD, PIC18F4520 e o leitor magnético KDE KT-2250.

Iniciamos o programa por importar todas a bibliotecas necessárias neste projeto e definindo os parâmetros da comunicação serial RS232 dando atenção especial ao pino C7 do porto C que é aonde a trama será recebida. Em seguida definimos os pinos de controlo RS para o pino E0 e Enable para o pino E1 (pinos mencionados anteriormente, que servem para o controlo do LCD).

Em seguida são declaradas as variáveis globais dando atenção especial ao **<char tecla='*'>,** que é a variável aonde será guardado a tecla premida. Igualamos a variável **tecla** a um **'*'**, o que indica que nenhuma tecla foi pressionada.

O programa todo conta com as seguintes funções que são chamadas durante toda a sua estrutura.

Funções destinados para o LCD

Foram criadas funções destinados ao funcionamento do LCD juntamente com o teclado. Das quais temos:

- **void activa(); -** Responsável por ativar o pino E1 que permite a ativação do módulo do LCD para a transferência de caracteres/instruções;

- **void transfere_carac(int carac); -** Responsável por colocar o pino **E0** a high, o que indica que vamos enviar um caracter para o porto **D;**
- **void transfere_inst (int inst); -** Responsável por colocar o pino **E0** a high, o que indica que vamos enviar uma instrução para o porto **D**;
- **void init_lcd(); -** Responsável por inicializar o LCD com uma série de instruções, dos quais temos:
    - **56 -** Especifica o modo de operação do LCD para **8 bits**, **2** linhas e **5x8 dots** para o tamanho dos caracteres;
    - **1 –** Limpa o LCD;
    - **12 –** Ativa o display, desativa o cursor visível e desativa o cursor piscante;
    - **2 –** Move o cursor para a posição inicial (início da primeira linha);

Função destinado ao teclado

Para fazer o varrimento das posições do nosso teclado que se encontra no porto B como mencionado anteriormente, usamos a função **char ler_tecla()** que é responsável por detetar a tecla premida, e guarda o caracter correspondente a essa tecla na variável anteriormente declarada como **char tecla.**

Void main (programa principal)

O programa principal começa por declarar a matriz **T** aonde é armazenado os dados (ID, Saldo, senha) dos 3 usuários. Em seguida é declarado todas as variáveis que serão utilizados durante o programa. Configura todos os pinos do porto **D** como **saídas** usando a instrução "**set_tris_d(0x00)**" e ativa os 4 bits **mais significativos** do porto **B** (RB4, RB5, RB6, RB7) e os bits **menos significativos** (RB0, RB1, RB2, RB3) como não ativos usando a instrução "**port_b_pullups(0XF0)**. Os pinos dos 4 bits **mais significativos** (RB4, RB5, RB6, RB7) são configurados como **entradas** e os pinos dos 4 bits **menos significativos** (RB0, RB1, RB2, RB3) são configurados como **saídas** usando a instrução "**set_tris_b(0XF0)**". É também chamado a função **void init_lcd()**, que explicado anteriormente, serve para inicializar o LCD. Por fim entramos no while aonde irá decorrer o resto do programa em um ciclo de loop.

Ciclo while (loop)

Ao entrarmos no ciclo while, o programa é inicializado com o envio dos caracteres utilizando a função **transfere_carac(caracter)**, aonde escreve "INTRODUCE CARD" no LCD. E fica a aguardar que o utilizador passe o cartão no leitor de cartões magnético.

A trama é lida utilizando a instrução **gets()**, e os dados são guardados na variável **char string[50].**

Após a leitura preenchemos 3 variáveis (str1, str2 e str3) com o ID referente aos usuários da matriz 'T'. Sendo assim: str1=ID do usuário 1, str2=ID do usuário 2 e str3=ID do usuário 3.

Em seguida preenchemos a variável **char readed[5]** com os últimos 4 dígitos do nº do cartão lido.

Comparação dos IDs

Após preencher as variáveis, comparamos o **readed** que é o ID do cartão lido com o **str1,2 e 3** utilizando a instrução **strcmp** no qual tem a função de comparar 2 strings e nos é retornado o valor 0 que é guardado nas variáveis **test1,2 e 3**.

Dentro de uma condição **if**, utilizamos a mesma instrução para comprar o **readed** com as variáveis **ID1, ID2** e **ID3** (contêm também o ID dos usuários 1,2 e 3). E com o auxílio da instrução strcpy que tem como função copiar uma string para uma outra, criamos as seguintes condições:

- Se readed=ID1, user_name = 'IVANDRO'.
- Se readed=ID2, user_name = 'CHAIMA'.
- Se readed=ID3, user_name = 'DAVID'.

Criamos também logo em seguida uma sequência de condições **'if'** com as seguintes condições:

- Se test1 for verdadeiro, será preenchido ambas as variáveis **"user=1" e "bal_po=8"**, sendo **user** o indicador do usuário e **"bal_pos"** a localização do seu saldo na matriz **"T"**;

- Se test2 for verdadeiro, será preenchido ambas as variáveis **“user=2” e “bal_po=20”**, sendo **user** o indicador do usuário e **“bal_pos”** a localização do seu saldo na matriz **“T”**;
- Se test3 for verdadeiro, será preenchido ambas as variáveis **“user=3” e “bal_po=32”**, sendo **user** o indicador do usuário e **“bal_pos”** a localização do seu saldo na matriz **“T”**;

Para a condição de negação criamos o seguinte caso **“if((test3!=0)&&(test1!=0)&&(test2!=0))”**, que para todos os valores de teste diferente de 0, ou seja, para todos os testes que o valor não for verdadeiro, utilizando as funções **transfere_inst(inst)** e **transfere_carac(caracter)** é escrito “INVALID” no LCD.

Caso um dos testes se confirmar, é escrito “WELCOME” + user_name que é a variável contendo o nome do usuário indicado.

Comparação do password

Após confirmar o usuário, é escrito no LCD “ENTER PASSWORD” (movendo “PASSWORD” para a segunda linha do LCD utilizando a instrução 192), aonde o programa entra em um ciclo **for** aguardando a leitura de **4 caracteres** e os armazenando na variável **“char pass[5]”.** E no mesmo método que os IDs dos usuários preenchidos através dos dados da matriz **“T”**, também foi guardado as passwords dos usuários nas variáveis **pass1,2 e 3** que correspondem aos usuários **1,2 e 3**.

Para comparar as passwords dos usuários preenchidos a partir da matriz, com a password inserida através do teclado, criamos uma cadeia **Swich,** utilizando a instrução **strcmp** (compara 2 strings) em 4 casos diferentes:

- Caso 1 – faz a comparação da **pass** (password inserido) com o **pass1** (password do usuário 1), e guarda o valor na variável **test4;**
- Caso 2 – faz a comparação da **pass** (password inserido) com o **pass2** (password do usuário 2), e guarda o valor na variável **test4;**
- Caso 3 – faz a comparação da **pass** (password inserido) com o **pass3** (password do usuário 3), e guarda o valor na variável **test4;**

Caso uma das condições for verdadeira, é escrita “VALID PASSWORD” no LCD e se as condições não forem verdadeiras é escrita “INCORRECT”.

Menu de operações

Após verificar o password, o programa passa para o menu de operações aonde inicialmente ele escreve no LCD todas 4 operações disponíveis para o usuário, sendo elas: 0 – Deposit, 1 – Withdraw, 2 – VIEW BALANCE e 3 – TRANSFER. O menu de operações é igual para todos os usuários comuns exceto para o usuário nº1 que corresponde ao número ligado a conta do administrador. Que igual aos usuários comuns, terá as mesmas operações, porém com a soma de mais 2 opções: 4 – ADD USER e 5 – DELETE USER.

Em seguida o programa com o auxílio da função **ler_tecla()**, irá aguardar a leitura de uma tecla correspondente a uma das operações. Caso a operação escolhida for diferente das apresentadas, é escrito “INVALID OPERATION” no LCD.

Operações

Para o menu de operações criamos uma cadeia **Swich** que seleciona as condições com base na **operação** escolhida. Entre as operações, temos:

Deposit (tecla 0)

Ao premir a tecla ‘0’, será apresentado no LCD o valor atual do seu saldo e em seguida lhe será perguntado a montante desejada a depositar (sendo ‘999’ o valor máximo permitido). Esse valor será guardado na variável **“char amount [ ]”**. Na variável **“old_bal [5]”** é guardado o saldo atual (guardado na matriz **T**) do usuário.

O processo de depósito consiste em somar o valor depositado com o valor atual sendo para isso necessário fazer a conversão de uma **string** (conjunto de char) para inteiro. Após a conversão guardamos o resultado o valor atual (**old_balance**) no variável **x**, e o montante depositado na variável **y**. Com a conversão realizada, passamos para a operação de soma (**x+y**) e o resultado foi guardado na variável **int balance**. Para salvar esse valor na posição do saldo atual no nosso base de dados

(**T**), foi preciso fazer a conversão desse valor de **int** para uma **string** (conjunto de **char**). Explicando os 2 processos de conversão de uma forma direta, temos:

Conversão de uma String em um valor inteiro

Para converter uma **string** em um valor inteiro (**int**), primeiro convertemos um **caractere** (‘0’,’1’,etc) para o seu valor numérico correspondente (0,1,etc). Esse processo é feito usando a operação **“old_bal[i] – ‘0’ “**, e após essa conversão, o resultado é guardado na variável correspondente (**y** ou **x**), aonde esse mesmo valor será **multiplicado** por **10** para o deslocar os dígitos criando assim um espaço para o próximo dígito. Todo esse processo é feito dentro de um ciclo **While** com a seguinte condição: **“while(old_bal[i] != ‘\0’)“**, que tem como função percorrer cada caractere da string até encontrar o caractere **nulo** (‘\0’), que marca o fim da string.

Conversão de um valor inteiro em uma String

Para converter um **valor inteiro** em uma **string**, primeiro pegamos no valor da soma que foi guardado na variável **balance**, e dividimos em milhares, centenas, dezenas e unidades e cada uma desses valores foram guardados nas variáveis correspondentes: th = milhares, h = centenas, d = dezenas e u=unidades. Todo esse processo foi feito dividindo o valor pela unidade correspondente (milhares = /1000, etc) e depois multiplicando pela percentagem correspondente a unidade que queremos descartar. Esse processo foi aplicado para todos os valores que fossem necessários converter:

1. th = balance / 1000:
    - extrai o dígito dos **milhares**.
2. h = (balance % 1000) / 100:
    - Primeiro, obtém o resto da divisão por 1000 (balance % 1000), removendo os milhares.
    - Depois, divide por 100 para obter as **centenas**;

3. d = ((balance % 1000) % 100) / 10:
    - Obtém o resto das centenas ((balance % 1000) % 100), removendo as milhares e centenas.
    - Divide por 10 para obter as **dezenas**;
4. u = ((balance % 1000) % 100) % 10;
    - Obtém o resto das dezenas (((balance % 1000) % 100) % 10), deixando apenas as **unidades**.

Após a conversão, todos os valores (**th**, **h**, **d** e **u**) são guardados na variável **char new_bal [5]** e esse valor será reescrita na base de dados substituindo o valor de saldo antigo pelo novo saldo. Por fim é apresentado o novo saldo do usuário no LCD.

Withdraw (tecla 1)

Ao premir a tecla '1', será apresentado no LCD o valor atual do seu saldo e em seguida lhe será perguntado a montante desejada a ser levantada. O processo é basicamente todo igual ao **Deposit** com apenas a exceção de:

- Ao invés de somar os valores convertidos, será feito a subtração (balance = x – y);
- Quando o montante inserido for maior do que o saldo disponível na base de dados, será impresso "INSUFFICIENT FUNDS!" no LCD, com que o usuário não tem o valor suficiente para efetuar a operação.

View Balance (tecla 2)

Ao premir a tecla '2' é apresentado o saldo correspondente ao usuário do cartão lido no leitor de cartões magnético. Esse processo é feito usando as funções **transfere_inst e transfere_carac** juntamente com um ciclo **for** que através do **bal_pos**, podemos saber a posição do saldo correspondente ao usuário na base de dados **T**. E por fim esse saldo será escrita no LCD.

Transfer (tecla 3)

Ao premir a tecla “3”, da mesma forma que as operações **Deposit** e **Withdraw**, é pedido o montante que no caso vai ser transferido. Tendo a mesma condição que o **Withdraw** aonde se o montante pretendido for maior que o saldo atual, será impresso “INSUFFICIENT FUNDS!” no LCD.

O processo de conversão e armazenamento dos valores é exatamente ao **Deposit** e **Withdraw**, aonde os valores são convertidos e armazenados nas variáveis **x** e **y** (x = saldo atual, y = montante inserido). A função dessa operação consiste em tirar o valor da conta do usuário que está a fazer a operação (processo similar ao **Withdraw**) e transferir esse dinheiro para a ID do utilizador pretendido (processo similar ao **Deposit,** porém será no ID do usuário que vai receber o dinheiro).

Após inserir o montante desejado, o programa irá pedir ao utilizador o ID correspondente a conta para qual será transferido o dinheiro. Quando inserido, o ID será guardado na variável **char ID_dest [5]** e essa variável será comparada com os IDs dos usuários guardados na base de dados (**str1** = usuário 1, **str2** = usuário 2 e **str3** = usuário 3) usando a instrução **strcmp** (compara 2 strings) e irá guardar o resultado dessa comparação nas variáveis **int1 transfer_test1**, **transfer_test2** e **transfer_test3** (**int1** por ser uma variável que pode armazenar apenas um bit único (0 ou 1). Usamos 3 diferentes condições **if**, todas com diferentes casos de acordo com o resultado das comparações:

- If (transfer_test1==0) – Se **o ID_dest** (ID de destino) for igual a **str1** (ID do usuário 1), o transfer_user = 1, o que indica que o ID de destino é pertencente ao **usuário 1** e des_bal_pos = 8, indicando a posição do saldo do usuário de destino na base de dados **T**;
- If (transfer_test2==0) – Se **o ID_dest** (ID de destino) for igual a **str2** (ID do usuário 2), o transfer_user = 2, o que indica que o ID de destino é pertencente ao **usuário 2** e des_bal_pos = 20, indicando a posição do saldo do usuário de destino na base de dados **T**;
- If (transfer_test3==0) – Se **o ID_dest** (ID de destino) for igual a **str3** (ID do usuário 3), o transfer_user = 3, o que indica que o ID de destino é pertencente ao **usuário 3** e des_bal_pos = 32, indicando a posição do saldo do usuário de destino na base de dados **T**;

Para verificarmos se o ID de destino fosse inválido fizemos a seguinte condição **“if((transfer_user==0)||(transfer_user==user))**, aonde indica que se nenhum usuário válido fosse identificado após as comparações de ID e também que se o usuário de destino fosse igual ao usuário atual, a operação seria inválida. Escrevendo assim “INVALID ID” no LCD.

Por fim, ao terminar todos os processos, é escrita “TRANSACTION SUCESSFUL” no LCD indicando assim o sucesso da operação.

ADD USER (tecla 4)

Caso o usuário atual seja o administrador, o seu menu de operações terá acesso a mais 2 operações, sendo uma delas **ADD USER**. Esta operação tem como função adicionar um novo usuário a base de dados.

A operação começa por apresentar o nº de usuários existentes e toda a informação presente na base de dados no LCD (ID, salvo e password dos usuários). Em seguida é perguntado ao administrador qual o ID do usuário novo, e essa informação é guardada na variável **char usernew_ID** que será em seguida escrita na base de dados **T** utilizando o ciclo **for**. A cada informação nova o **length** (destinado a aumentar o tamanho da base de dados) aumenta o seu valor somando com 4 (**length = lenght + 4**). Visto que cada uma das informações (com exceção do **ID**), tem 4 digitos.

Após inserir o ID novo, é perguntado ao administrador qual a password pretendida e essa informação será guardada na variável **char usernew_pass [ ]**, que igual ao ID, também será escrita na base de dados **T**.

Posteriormente é perguntado ao administrador o montante de saldo inicial do novo utilizador. Essa informação será guardada na variável **char usernew_bal [ ]** que também será escrita na base de dados **T**.

Por fim é apresentado o base de dados atualizado no LCD contendo o ID, saldo e password de todos os utilizadores incluindo os novos.

DELETE USER (tecla 5)

Diferente da operação **ADD USER**, a operação **DELETE USER** tem como função deletar um usuário da base de dados. Começa por apresentar o nº de usuários existentes e toda a informação presente na base de dados no LCD (ID, salvo e password dos usuários). Em seguida é perguntado ao administrador qual o ID do usuário que pretende deletar, e essa informação é guardada na variável **char delete_nbr**.

O **DELETE USER** terá algumas etapas durante todo o seu processo:

1. O ID do usuário a ser deletado será convertido de um caractere para um valor inteiro correspondente.

2. Para determinar a posição inicial do usuário a ser deletado foi preciso fazer a seguinte operação:
   - Del_pos = (delete_nbr – 1) * 12 : Como cada usuário ocupa **12 posições** na base de dados (usuário **1** = posição **0**, usuário **2** = posição **12** e usuário **3** = posição **23**), subtraindo 1 de **delete_nbr** e multiplicando por 12, foi possível obter a posição inicial do usuário que será removido na base de dados.
3. Para remover o usuário, foi implementado o seguinte for ciclo:

   for (m = del_pos; m < (del_pos + 12) – 1; m++) {
   T [m] = T [m + 12]; }
   - Este ciclo começa na posição inicial do usuário a ser deletado (del_pos) e percorre até a última posição do próximo usuário.
   - Em seguida ele sobrescreve os dados do usuário atual com os dados do próximo usuário (deslocando os dados para a esquerda em 12 posições)
   - Essencialmente, cada usuário após o usuário deletado é movido uma posição para “tapar p buraco” deixado pelo usuário removido. Por exemplo, temos o seguinte base de dados [Usuário1 | Usuário2 | Usuário3 | Usuário4]: se o **usuário 2** for deletado, os dados de **usuário 3** substituem os de **usuário 2** e os dados do **usuário 4** substituem os de **usuário 3**.
4. Após a remoção, o tamanho total da base de dados (**Length**) é reduzido em **12** (length = length – 12), pois o espaço correspondente ao usuário deletado não é mais considerado válido.

5. O valor do user_nb que corresponde ao número total de usuários no banco de dado é atualizado subtraindo o seu valor total por 1 (**user_nb = user_nb-1**).

Por fim são apresentados o base de dados e o número de utilizadores atualizados no LCD.

CONCLUSÃO

A realização deste projeto na disciplina proporcionou uma oportunidade valiosa para aplicar conhecimentos teóricos e práticos na simulação de uma caixa multibanco. Utilizando a Millennium Board PCB143 REV2.0 com o microcontrolador PIC18F4520, um teclado matricial, um LCD 16x4 e um leitor de cartões magnéticos KDE KT-2250, foi possível integrar diferentes componentes eletrónicos e desenvolver uma solução funcional.

Entretanto, começamos por realizar o projeto com o microcontrolador PIC16F874A, porém pela falta de memória visto que o nosso programa seria extenso em termos de linhas de códigos, foi necessário trocar para o PIC18F4520.

O programa foi desenvolvido para 3 utilizadores, sendo 2 comuns e 1 administrador, por isso a base de dados foi configurado com um tamanho limite. Porém para um projeto futuro, ficou a ideia de implementar uma base de dados adaptável para que fosse possível adicionar mais de 3 utilizadores e possivelmente mais administradores.

Também foi adicionado a lista de ideias para a melhoria futura do projeto, darmos mais opções de funcionalidades extras para o menu do administrador.

Contudo o projeto alcançou com sucesso os objetivos propostos, demonstrando as funcionalidades essenciais de uma caixa multibanco, como a leitura de cartões magnéticos, a interação com o usuário por meio do teclado matricial, a exibição de informações no LCD e as funcionalidades propostas como levantamento de dinheiro, transferência de dinheiro, consulta de saldo e depósito. Também para o menu do administrador foi alcançado com sucesso as funcionalidades de adicionar conta e remover. Esta experiência permitiu consolidar conceitos importantes sobre microcontroladores, protocolos de comunicação e sistemas integrados.

DIAGRAMA DE BLOCOS

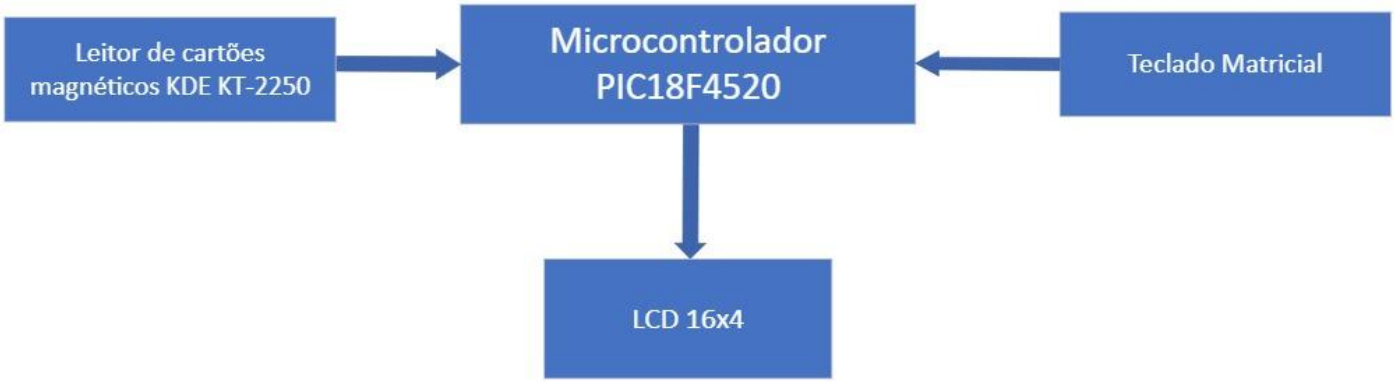


Figura 3 - Diagrama de blocos do Sistema de caixa Multibanco

FLUXOGRAMA DO PROGRAMA PRINCIPAL

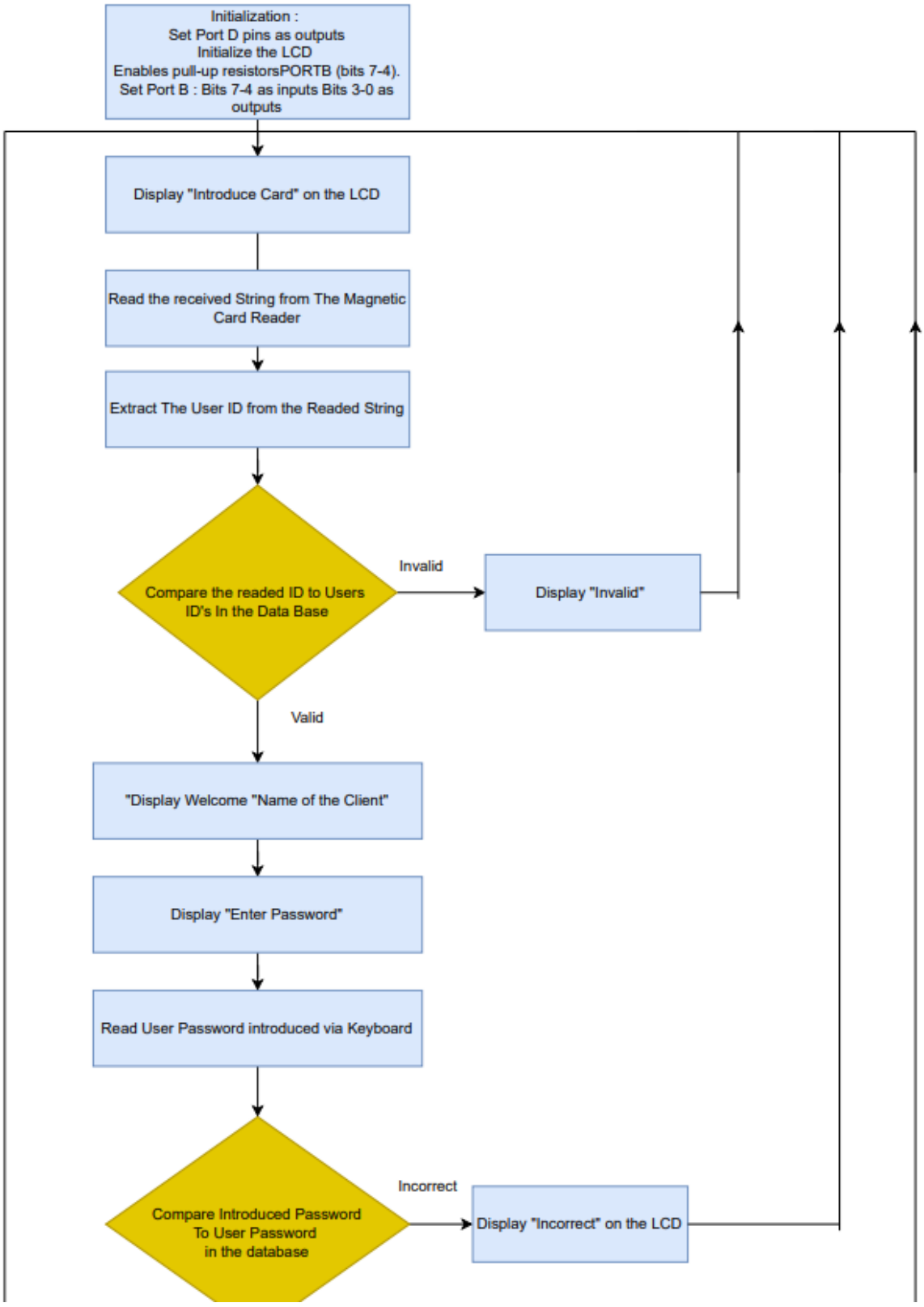


Figura 4 - Fluxograma do programa principal

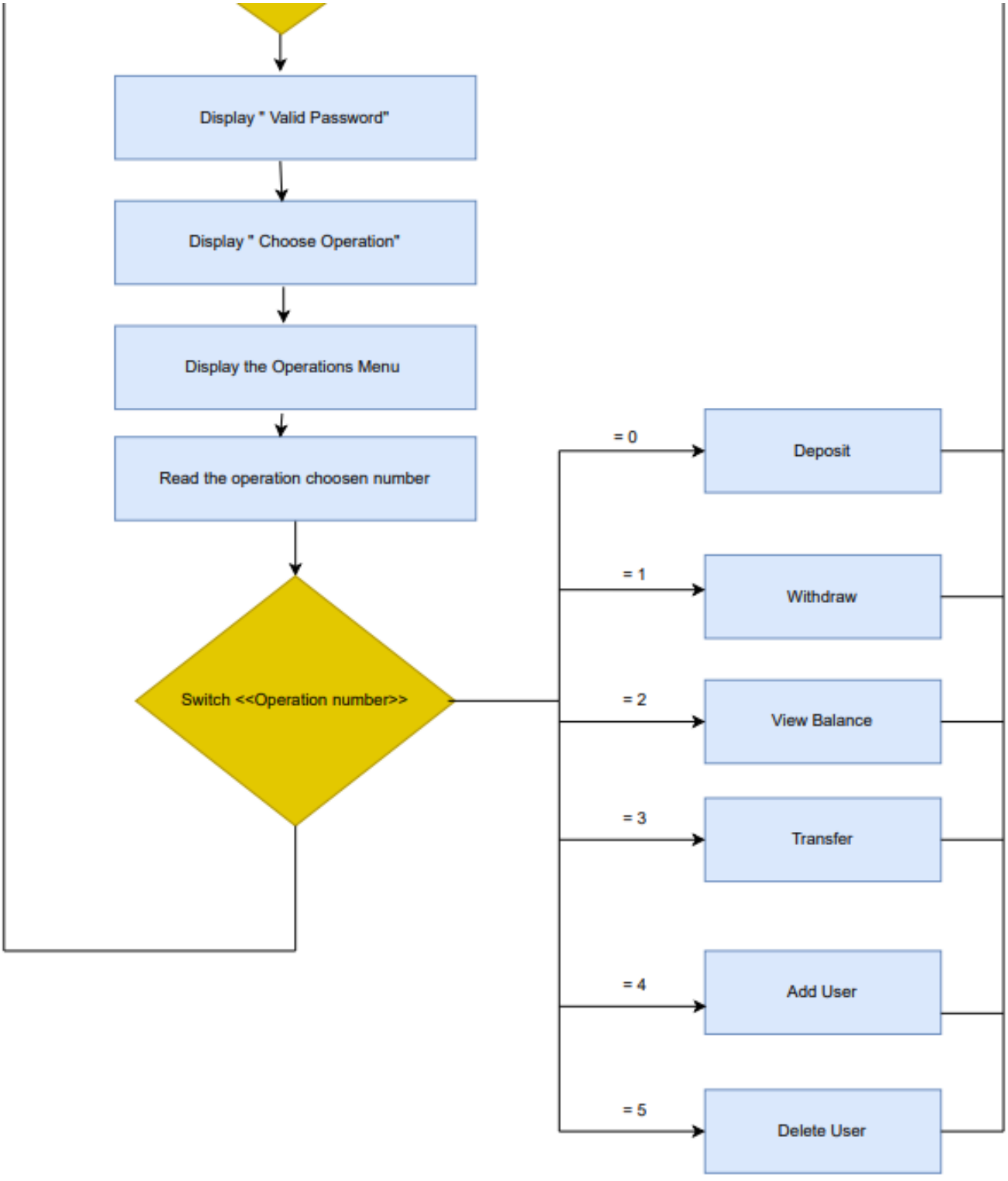


Figura 5 - Fluxograma do programa principal

FLUXOGRAMA DO MENU DE OPERAÇÕES

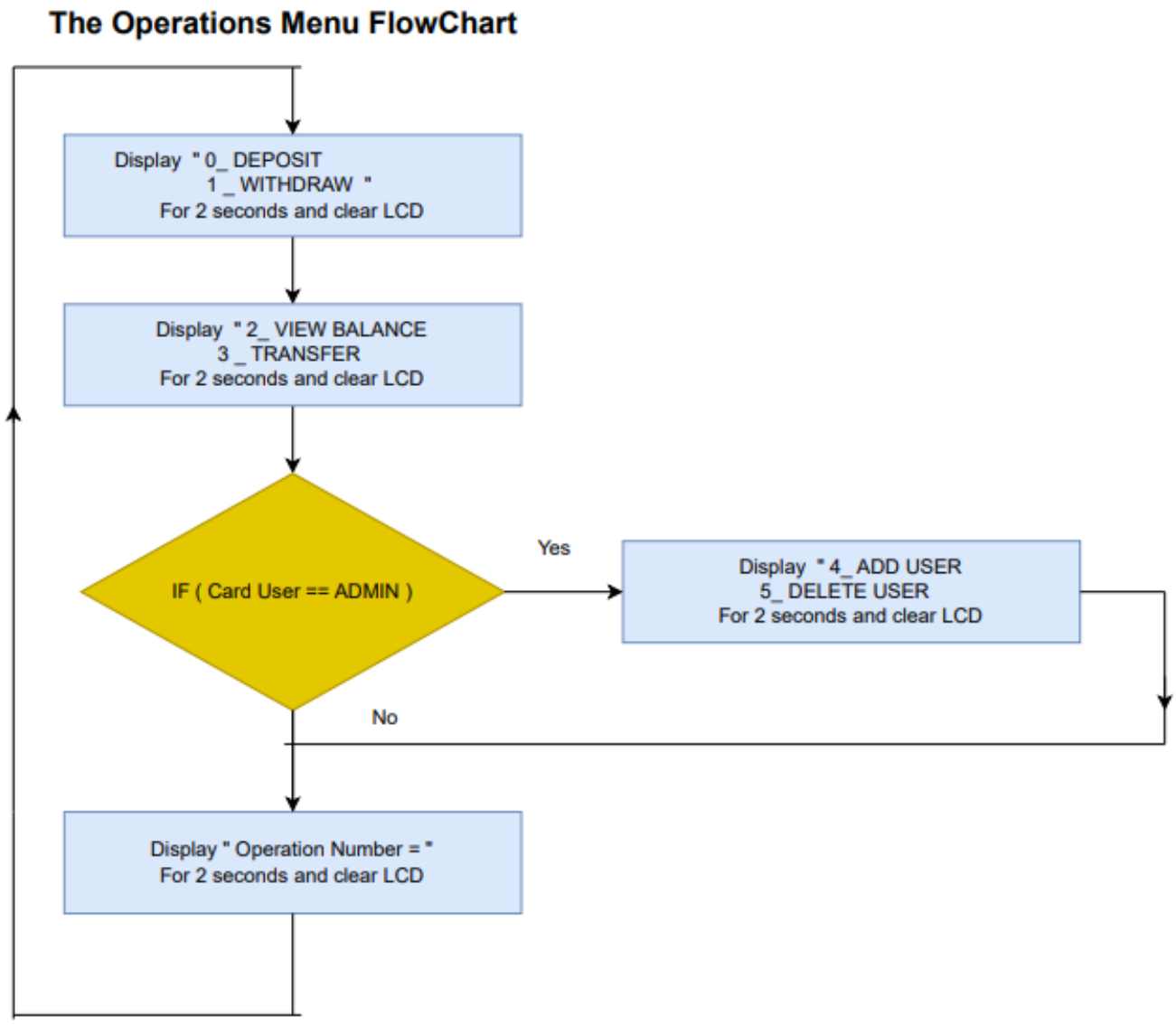


Figura 6 - Fluxograma do menu de operações

CÓDIGOS DO PROGRAMA

```
#include <18F4520.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#fuses NOWDT,HS, NOPUT, NOPROTECT, NODEBUG, NOLVP, NOCPD,
NOWRT, BROWNOUT

#use delay(clock=20000000)
#use rs232 (baud= 9600, xmit=PIN_C7, rcv=PIN_C6, parity= N, bits=8)
#define RS     PIN_E0
#define ENABLE PIN_E1
//#define LSB PIN_A2

int instrucao;
char caracter;
int16 cont;
char tecla='*';
char   tecla1;
void activa()
{
  output_high(ENABLE);
  delay_ms(20);
  output_low(ENABLE);
}

void transfere_carac(int carac)
{
  output_high(RS);
  output_d(carac);
  activa();
}

void transfere_inst(int inst)
{
  output_low(RS);
  output_d(inst);
  activa();
```

```
}

void init_lcd()
 {
 instrucao=56;
  transfere_inst(instrucao);
  instrucao=1;
  transfere_inst(instrucao);
  instrucao=12;
  transfere_inst(instrucao);
  instrucao=2;              // colocação do cursor na 1º linha
  transfere_inst(instrucao);
 }

char ler_tecla()
{

  tecla='*';

  output_low(pin_b0);
  output_high(pin_b1);
  output_high(pin_b2);
  output_high(pin_b3);

  while((input(pin_b4)==0) && (input(pin_b4)==0) && (input(pin_b4)==0)){
   tecla='1';
return(tecla);
   }

  while((input(pin_b5)==0) && (input(pin_b5)==0) && (input(pin_b5)==0)){
   tecla='4';
return(tecla);
   }

  while((input(pin_b6)==0) && (input(pin_b6)==0) && (input(pin_b6)==0)){
   tecla='7';
return(tecla);
   }
```

```
  while((input(pin_b7)==0) && (input(pin_b7)==0) && (input(pin_b7)==0)){
   tecla='A';
return(tecla);
   }

  output_high(pin_b0);
  output_low(pin_b1);
  output_high(pin_b2);
  output_high(pin_b3);

  while((input(pin_b4)==0) && (input(pin_b4)==0) && (input(pin_b4)==0)){
   tecla='2';
return(tecla);
   }

  while((input(pin_b5)==0) && (input(pin_b5)==0) && (input(pin_b5)==0)){
   tecla='5';
return(tecla);
   }

  while((input(pin_b6)==0) && (input(pin_b6)==0) && (input(pin_b6)==0)){
   tecla='8';
return(tecla);
    }

  while((input(pin_b7)==0) && (input(pin_b7)==0) && (input(pin_b7)==0)){
   tecla='0';
return(tecla);
    }


  output_high(pin_b0);
  output_high(pin_b1);
  output_low(pin_b2);
  output_high(pin_b3);


  while((input(pin_b4)==0) && (input(pin_b4)==0) && (input(pin_b4)==0)){
   tecla='3';
```

```
return(tecla);
   }

   while((input(pin_b5)==0) && (input(pin_b5)==0) && (input(pin_b5)==0)){
   tecla='6';
return(tecla);
   }

   while((input(pin_b6)==0) && (input(pin_b6)==0) && (input(pin_b6)==0)){
   tecla='9';
return(tecla);
   }

   while((input(pin_b7)==0) && (input(pin_b7)==0) && (input(pin_b7)==0)){
   tecla='B';
return(tecla);
   }


  output_high(pin_b0);
  output_high(pin_b1);
  output_high(pin_b2);
  output_low(pin_b3);

  while((input(pin_b4)==0) && (input(pin_b4)==0) && (input(pin_b4)==0)){
   tecla='F';
return(tecla);
   }

  while((input(pin_b5)==0) && (input(pin_b5)==0) && (input(pin_b5)==0)){
   tecla='E';
return(tecla);
   }

  while((input(pin_b6)==0) && (input(pin_b6)==0) && (input(pin_b6)==0)){
   tecla='D';
return(tecla);
   }
```

```
  while((input(pin_b7)==0) && (input(pin_b7)==0) && (input(pin_b7)==0)){
   tecla='C';
return(tecla);
   }

}



void main()
{

char T[60]="06910000000045891111000025081234150O";
 char A[60]="";
   int i;
   char k;
char ID1[5]="0691";
char ID2[5]="4589";
char ID3[5]="2508";
char string[50];
char str1[5]="";
char str2[5]="";
char str3[5]="";
char user_name[10]="";
char readed[5]="";
char pass1[5]="";
char pass2[5]="";
char pass3[5]="";
char old_bal[5]="";
char pass[5]="";
int1 test1=1 , test2=1 , test3=1;
int user=0;
int1 test4=1;
char operation;
char amount[5]="";
char new_bal[5]="";
char ID_dest[5]="";
char bal_dest_old[5]="";
char bal_dest_new[5]="";
```

```
int16 balance=0;
int16 x=0;// old bal of card user
int16 y=0;// amount
int16 z=0;// dest old balance
int bal_pos;
int des_bal_pos;
int th,h,d,u;
int1 transfer_test1=1 , transfer_test2=1 ,transfer_test3=1;
int transfer_user=0;
int user_nb=3;
int j;
char delete_nb;
int delete_nbr;
int length=36;
int del_pos;
int m;
char usernew_ID[5];
char usernew_pass[5];
char usernew_bal[5];
int MAX_USER_NB=5;
char o;
int8 v;
int w=0;



set_tris_d(0x00);
init_lcd();
port_b_pullups(0xF0); //1-Outputs | 2-Inputs
set_tris_b(0xF0);


  while (1)
   {
   transfere_inst(1);
 caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
```

```
caracter ='T';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
transfere_inst(192);
caracter ='C';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);

delay_ms(2000);




user=0;

gets(string);
transfere_inst(1);
transfere_inst(128);
for (i=14;i<=17;i++)
{
caracter =string[i]; // display the ID readed
transfere_carac(caracter);
}
```

```
delay_ms(1000);

for(i=0;i<=3;i++)
{
str1[i]=T[i];
}
str1[4]='\0';

for(i=0;i<=3;i++)
{
str2[i]=T[i+12];
}
str2[4]='\0';

for(i=0;i<=3;i++)
{
str3[i]=T[i+24];
}
str3[4]='\0';

readed[0]=string[14];
readed[1]=string[15];
readed[2]=string[16];
readed[3]=string[17];
readed[4]='\0';

test1=strcmp(readed, str1);
test2=strcmp(readed, str2);
test3=strcmp(readed, str3);
if(0==strcmp(readed, ID1)){
strcpy (user_name, "IVANDRO");
}
if(0==strcmp(readed, ID2)){
strcpy (user_name, "CHAIMA");
}
if(0==strcmp(readed, ID3)){
strcpy (user_name, "DAVID");
}
```

```
if(test1==0)
{
user=1;
bal_pos=8;

}
if(test2==0)
{
user=2;
bal_pos=20;

}
if(test3==0)
{
user=3;
bal_pos=32;
}

if((test3!=0)&&(test1!=0)&&(test2!=0))
{
transfere_inst(192);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='V';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
delay_ms(1000);
test4=1;
}
```

```
else
{
instrucao=1;
transfere_inst(instrucao);
instrucao=128;
transfere_inst(instrucao);
caracter ='W';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
transfere_inst(192);
i=0;
while (user_name[i] != '\0') {
      transfere_carac(user_name[i]);
       i++;
   }
delay_ms(1000);

instrucao=1;
transfere_inst(instrucao);
instrucao=128;
transfere_inst(instrucao);
caracter ='E';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='E';
```

```
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
instrucao=192;
transfere_inst(instrucao);
caracter ='P';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
delay_ms(1000);

transfere_inst(1);
transfere_inst(128);

//reading password from keyboard

transfere_carac('P');
transfere_carac('A');
transfere_carac('S');
transfere_carac('S');
transfere_carac('W');
transfere_carac('O');
transfere_carac('R');
transfere_carac('D');
transfere_carac('=');

  for(i=0;i<4;i++){
```

```
    ler_tecla();
  while (tecla=='*')
  {
     ler_tecla();
 }
 k=tecla;

 pass[i]=k;
transfere_carac('*');
delay_ms(1000);

   }
pass[4]='\0';



//fill pass of user1
for(i=0;i<=3;i++)
{
pass1[i]=T[i+4];
}
pass1[4]='\0';

//fill pass of user2
for(i=0;i<=3;i++)
{
pass2[i]=T[i+16];
}
pass2[4]='\0';

//fill pass of user3
for(i=0;i<=3;i++)
{
pass3[i]=T[i+28];
}
pass3[4]='\0';

//comparing password readed and user password in database
// switch user which balance to adjust or work with
```

```
switch (user) {

    case 1:
         test4=strcmp(pass, pass1);

          break;

    case 2:test4=strcmp(pass, pass2);

          break;

    case 3:test4=strcmp(pass, pass3);

          break; }
}
         if(test4==0){ // if right user
instrucao=1;
transfere_inst(instrucao);
instrucao=128;
transfere_inst(instrucao);
caracter ='V';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
transfere_carac(' ');
transfere_inst(192);
transfere_carac('P');
transfere_carac('A');
transfere_carac('S');
transfere_carac('S');
transfere_carac('W');
transfere_carac('O');
```

```
transfere_carac('R');
transfere_carac('D');
delay_ms(1000);



//SELECT OPERATION
transfere_inst(1);
transfere_inst(128);
caracter ='C';
transfere_carac(caracter);
caracter ='H';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
transfere_inst(192);
caracter ='O';
transfere_carac(caracter);
caracter ='P';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='N';
```

```
transfere_carac(caracter);
delay_ms(1000);

transfere_inst(1);
transfere_inst(128);
caracter ='0';
transfere_carac(caracter);
caracter ='-';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='P';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);

instrucao=192;
transfere_inst(instrucao);
caracter ='1';
transfere_carac(caracter);
caracter ='-';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='H';
transfere_carac(caracter);
caracter ='D';
```

```
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
delay_ms(2000);
instrucao=1;
transfere_inst(instrucao);
instrucao=128;
transfere_inst(instrucao);
caracter ='2';
transfere_carac(caracter);
caracter ='-';
transfere_carac(caracter);
caracter ='V';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='E';
```

```
        transfere_carac(caracter);
        transfere_inst(192);
        caracter ='3';
        transfere_carac(caracter);
        caracter ='-';
        transfere_carac(caracter);
        caracter ='T';
        transfere_carac(caracter);
        caracter ='R';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
        caracter ='N';
        transfere_carac(caracter);
        caracter ='S';
        transfere_carac(caracter);
        caracter ='F';
        transfere_carac(caracter);
        caracter ='E';
        transfere_carac(caracter);
        caracter ='R';
        transfere_carac(caracter);
        delay_ms(2000);

        if(user==3) {  //**************************** if it's the admin display 2 extra
operations to add and delete users
        instrucao=1;
        transfere_inst(instrucao);
        instrucao=128;
        transfere_inst(instrucao);
        caracter ='4';
        transfere_carac(caracter);
        caracter ='-';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
        caracter ='D';
        transfere_carac(caracter);
        caracter ='D';
```

```
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_inst(192);
caracter ='5';
transfere_carac(caracter);
caracter ='-';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
delay_ms(2000);
```

```
        }//**************************************end admin
code*********************************************
        transfere_inst(1);
        transfere_inst(128);
        caracter ='O';
        transfere_carac(caracter);
        caracter ='P';
        transfere_carac(caracter);
        caracter ='E';
        transfere_carac(caracter);
        caracter ='R';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
        caracter ='T';
        transfere_carac(caracter);
        caracter ='I';
        transfere_carac(caracter);
        caracter ='O';
        transfere_carac(caracter);
        caracter ='N';
        transfere_carac(caracter);
        caracter ='=';
        transfere_carac(caracter);
        delay_ms(1000);

        // read operation

            ler_tecla();
          while (tecla=='*')
          {
              ler_tecla();
         }
        operation=tecla;
        transfere_carac(operation);
        delay_ms(1000);
        transfere_inst(1);
        transfere_inst(128);
```

```
          if( (operation!='0') && (operation!='1' ) && (operation!='2') &&(operation!='3'
)&&(operation!='4' )&&(operation!='5' ))
          {
        transfere_inst(1);
        transfere_inst(128);
        caracter ='I';
        transfere_carac(caracter);
        caracter ='N';
        transfere_carac(caracter);
        caracter ='V';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
        caracter ='L';
        transfere_carac(caracter);
        caracter ='I';
        transfere_carac(caracter);
        caracter ='D';
        transfere_carac(caracter);
        transfere_inst(192);
        caracter ='O';
        transfere_carac(caracter);
        caracter ='P';
        transfere_carac(caracter);
        caracter ='E';
        transfere_carac(caracter);
        caracter ='R';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
        caracter ='T';
        transfere_carac(caracter);
        caracter ='I';
        transfere_carac(caracter);
        caracter ='O';
        transfere_carac(caracter);
        caracter ='N';
        transfere_carac(caracter);
```

```
delay_ms(1000);
  }

   // switch operation
switch (operation) {

    case '0': //************ Deposit
    {

     transfere_inst(1);
transfere_inst(128);
caracter ='A';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);


delay_ms(1000);
 for(i=0;i<4;i++){
     ler_tecla();
  while (tecla=='*')
  {
      ler_tecla();
 }
 k=tecla;

 amount[i]=k;
transfere_carac(amount[i]);
```

```
delay_ms(1000);

  }

   amount[4]='\0';
   // put old balance in  old_bal (copy)
   for(i=0;i<4;i++)
   {
   old_bal[i]=T[i+bal_pos];//old balance
   }
   old_bal[4]='\0';
    transfere_inst(1);
transfere_inst(128);
caracter ='O';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='d';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
// new_bal initialization
new_bal="0000";
new_bal[4]='\0';
balance=0;
//display old balance
   for(i=0;i<4;i++)
   {
   transfere_carac(old_bal[i]);
   }
   delay_ms(1000);
   // converting string amount to int y
   i=0;
   y=0;
while (amount[i] != '\0') {
```

```
            y = y * 10 + (amount[i] - '0'); // Accumulate digits
            i++;
        }

        // converting string old_bal to int X
        i=0;
        x=0;
    while (old_bal[i] != '\0') {
            x = x * 10 + (old_bal[i] - '0'); // Accumulate digits
            i++;
        }

        // sum and put into balance
        balance= x+y;
        //convert balance to a string and save it in new_bal and affect it in data
base
        th=balance/1000;

    h=(balance%1000)/100;

    d=((balance%1000)%100)/10;

    u=((balance%1000)%100)%10;
    new_bal[0]=th+48;
    new_bal[1]=h+48;
    new_bal[2]=d+48;
    new_bal[3]=u+48;
    new_bal[4]='\0';
    for(i=0;i<4;i++)
        {
        T[i+bal_pos]=new_bal[i];
        }

        //affect balance to vector according to bal_pos

        // display the new balance
         transfere_inst(1);
    transfere_inst(128);
    caracter ='N';
```

```
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
   delay_ms(1000);


   for(i=0;i<4;i++)
   {
   transfere_carac(new_bal[i]);
   }
   delay_ms(1000);
    transfere_inst(1);

         break;
}

   case '1':// withdraw
{

    transfere_inst(1);
transfere_inst(128);
caracter ='A';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
```

```
caracter ='U';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);


delay_ms(1000);
 for(i=0;i<4;i++){
     ler_tecla();
  while (tecla=='*')
  {
      ler_tecla();
 }
 k=tecla;

 amount[i]=k;
transfere_carac(amount[i]);
delay_ms(1000);

   }

    amount[4]='\0';
    // put old balance in  old_bal (copy)
    for(i=0;i<4;i++)
    {
    old_bal[i]=T[i+bal_pos];//old balance
    }
    old_bal[4]='\0';
     transfere_inst(1);
transfere_inst(128);
caracter ='O';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='d';
```

```
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
// new_bal initialization
new_bal="0000";
new_bal[4]='\0';
balance=0;
//display old balance
   for(i=0;i<4;i++)
   {
   transfere_carac(old_bal[i]);
   }
   delay_ms(1000);
   // converting string amount to int y
   i=0;
   y=0;
while (amount[i] != '\0') {
       y = y * 10 + (amount[i] - '0'); // Accumulate digits
       i++;
   }

   // converting string old_bal to int X
   i=0;
   x=0;
while (old_bal[i] != '\0') {
       x = x * 10 + (old_bal[i] - '0'); // Accumulate digits
       i++;
   }


   if(y>x){// not enough balance Insufficient funds
   transfere_inst(1);
transfere_inst(128);
caracter ='I';
transfere_carac(caracter);
```

```
caracter ='N';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='F';
transfere_carac(caracter);
caracter ='F';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);

transfere_inst(192);
caracter ='F';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='!';
transfere_carac(caracter);
  delay_ms(1000);

   transfere_inst(1);
```

```
          }
        else { // enough funds
           balance= x-y;
           //convert balance to a string and save it in new_bal and affect it in data
base
           th=balance/1000;

        h=(balance%1000)/100;

        d=((balance%1000)%100)/10;

        u=((balance%1000)%100)%10;
        new_bal[0]=th+48;
        new_bal[1]=h+48;
        new_bal[2]=d+48;
        new_bal[3]=u+48;
        new_bal[4]='\0';
        for(i=0;i<4;i++)
           {
           T[i+bal_pos]=new_bal[i];
           }

           //affect balance to vector according to bal_pos

           // display the new balance
            transfere_inst(1);
        transfere_inst(128);
        caracter ='N';
        transfere_carac(caracter);
        caracter ='E';
        transfere_carac(caracter);
        caracter ='W';
        transfere_carac(caracter);
        caracter =' ';
        transfere_carac(caracter);
        caracter ='B';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
```

```
caracter ='L';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
   delay_ms(1000);
   for(i=0;i<4;i++)
   {
   transfere_carac(new_bal[i]);
   }
   delay_ms(1000);

   transfere_inst(1);
} // enough funds ends

         break;
}


   case '2':// view balance
transfere_inst(1);
transfere_inst(128);

caracter ='B';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
   delay_ms(1000);
```

```
    for(i=0;i<4;i++)
    {
    transfere_carac(T[i+bal_pos]);
    }
    delay_ms(2000);
    transfere_inst(1);

          break;
    case '3':// transfer--------------------------------------------
{ // code transfer begin
transfere_inst(1);
transfere_inst(128);
caracter ='A';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);


delay_ms(1000);
 for(i=0;i<4;i++){
     ler_tecla();
  while (tecla=='*')
  {
      ler_tecla();
 }
 k=tecla;

 amount[i]=k;
transfere_carac(amount[i]);
```

```
delay_ms(1000);

  }

   amount[4]='\0';
// put old balance in  old_bal (copy)
   for(i=0;i<4;i++)
   {
   old_bal[i]=T[i+bal_pos];//old balance
   }
   old_bal[4]='\0';
    transfere_inst(1);
transfere_inst(128);
caracter ='O';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='d';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
// new_bal initialization
new_bal="0000";
new_bal[4]='\0';
balance=0;
//display old balance
   for(i=0;i<4;i++)
   {
   transfere_carac(old_bal[i]);
   }
   delay_ms(1000);
// converting string amount to int y
   i=0;
   y=0;
while (amount[i] != '\0') {
```

```
        y = y * 10 + (amount[i] - '0'); // Accumulate digits
        i++;
    }

    // converting string old_bal to int X
    i=0;
    x=0;
while (old_bal[i] != '\0') {
        x = x * 10 + (old_bal[i] - '0'); // Accumulate digits
        i++;
    }


    if(y>x){// not enough balance Insufficient funds
    transfere_inst(1);
transfere_inst(128);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='F';
transfere_carac(caracter);
caracter ='F';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='T';
```

```
transfere_carac(caracter);

transfere_inst(192);
caracter ='F';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='!';
transfere_carac(caracter);
  delay_ms(1000);
  }
else { // enough funds
   // Ask Id
   transfere_inst(1);
transfere_inst(128);
caracter ='D';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='O';
```

```
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
transfere_inst(192);
caracter ='I';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);

delay_ms(1000);
 for(i=0;i<4;i++){
     ler_tecla();
  while (tecla=='*')
  {
      ler_tecla();
 }
 k=tecla;

 ID_dest[i]=k;
transfere_carac(ID_dest[i]);
delay_ms(1000);
 }
ID_dest[4]='\0';
// compare ID with users ID in data base
transfer_test1=strcmp(ID_dest, str1);
transfer_test2=strcmp(ID_dest, str2);
transfer_test3=strcmp(ID_dest, str3);


if(transfer_test1==0)
{
transfer_user=1;
des_bal_pos=8;

}
if(transfer_test2==0)
{
```

```
transfer_user=2;
des_bal_pos=20;
}
if(transfer_test3==0)
{
transfer_user=3;
des_bal_pos=32;

}

if((transfer_user==0)||(transfer_user==user)) //if invalid id dest
{
transfere_inst(1);
transfere_inst(128);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='V';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);

delay_ms(1000);
} // if invalid id destination ends
else { //************** if valid id
destination*****************************************************************************
```

```
        // add balance to the destination
##################################################################
        // put old balance of destination bal_dest_old
           for(i=0;i<4;i++)
           {
           bal_dest_old[i]=T[i+des_bal_pos];//old balance
           }
           bal_dest_old[4]='\0';
         // converting string ID_dest_old to int z
           i=0;
           z=0;
        while (bal_dest_old[i] != '\0') {
              z = z * 10 + (bal_dest_old[i] - '0'); // Accumulate digits
              i++;
           }
        balance= y+z;
        //convert balance to a string and save it in bal_dest_new and affect it in data
base
           th=balance/1000;

        h=(balance%1000)/100;

        d=((balance%1000)%100)/10;

        u=((balance%1000)%100)%10;
        bal_dest_new[0]=th+48;
        bal_dest_new[1]=h+48;
        bal_dest_new[2]=d+48;
        bal_dest_new[3]=u+48;
        bal_dest_new[4]='\0';
        //affect new destination balance to vector according to des_bal_pos
        for(i=0;i<4;i++)
           {
           T[i+des_bal_pos]=bal_dest_new[i];
           }
```

```
        // sustract balance from user
#####################################################################
        balance= x-y;
           //convert balance to a string and save it in new_bal and affect it in data
base
           th=balance/1000;

        h=(balance%1000)/100;

        d=((balance%1000)%100)/10;

        u=((balance%1000)%100)%10;
        new_bal[0]=th+48;
        new_bal[1]=h+48;
        new_bal[2]=d+48;
        new_bal[3]=u+48;
        new_bal[4]='\0';
        for(i=0;i<4;i++)
           {
           T[i+bal_pos]=new_bal[i];
           }

           //affect balance to vector according to bal_pos

           // display the new balance
        transfere_inst(1);
        transfere_inst(128);
        caracter ='N';
        transfere_carac(caracter);
        caracter ='E';
        transfere_carac(caracter);
        caracter ='W';
        transfere_carac(caracter);
        caracter =' ';
        transfere_carac(caracter);
        caracter ='B';
        transfere_carac(caracter);
        caracter ='A';
        transfere_carac(caracter);
```

```
caracter ='L';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
delay_ms(1000);
for(i=0;i<4;i++)
{
transfere_carac(new_bal[i]);
}
delay_ms(1000);

transfere_inst(1);
transfere_inst(128);
caracter ='T';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
transfere_inst(192);

caracter ='S';
transfere_carac(caracter);
```

```
caracter ='U';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='F';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);

delay_ms(2000);
   transfere_inst(1);



 }//********** if vald id dest ends**************************************


}// enough funds ends

break;
}// code transfer ends = case 3 ends
case '4':// ADD USER
{//case 4 code begins
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
```

```
caracter ='R';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);

transfere_carac(user_nb+48);

delay_ms(1500);
j=0;
// display current data base
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='1';
transfere_carac(caracter);
caracter ='=';
```

```
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
transfere_inst(192);
for(i=0;i<12;i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j=2;
while(j<=user_nb)
{
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_carac(j+48);
transfere_carac('=');
transfere_inst(192);

for(i=(j-1)*12;i<(j*12) ;i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j++;
}
```

```
// ask admin for usernew_ID
transfere_inst(1);
transfere_inst(128);
caracter ='N';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_inst(192);
caracter ='I';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);

delay_ms(1000);
 for(i=0;i<4;i++){
    ler_tecla();
  while (tecla=='*')
  {
     ler_tecla();
 }
 k=tecla;

 usernew_ID[i]=k;
transfere_carac(usernew_ID[i]);
delay_ms(1000);
```

```
  }

   usernew_ID[4]='\0';


//end usernew_ID/ write it in T
for(i=0;i<4;i++){
  T[i+length]=usernew_ID[i];

  }
  length=length+4;
//end
// ask admin for usernew_pass
transfere_inst(1);
transfere_inst(128);
caracter ='N';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_inst(192);
caracter ='P';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
```

```
caracter ='S';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
delay_ms(1000);

 for(i=0;i<4;i++){
     ler_tecla();
  while (tecla=='*')
  {
      ler_tecla();
 }
 k=tecla;

 usernew_pass[i]=k;
transfere_carac(usernew_pass[i]);
delay_ms(1000);

   }

    usernew_pass[4]='\0';


//end usernew_pass/ write new pass in T
for(i=0;i<4;i++){
   T[i+length]=usernew_pass[i];

   }
   length=length+4;
//end
// ask admin for usernew_balance
transfere_inst(1);
transfere_inst(128);
caracter ='N';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
```

```
caracter =' ';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_inst(192);
caracter ='B';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
delay_ms(1000);

 for(i=0;i<4;i++){
    ler_tecla();
  while (tecla=='*')
  {
     ler_tecla();
 }
 k=tecla;

 usernew_bal[i]=k;
transfere_carac(usernew_bal[i]);
```

```
delay_ms(1000);

  }

   usernew_bal[4]='\0';


//end usernew_pass/ write new bal in T
for(i=0;i<4;i++){
  T[i+length]=usernew_bal[i];

  }
  length=length+4;
//end

user_nb=user_nb+1;

// display the new database
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='1';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='I';
```

```
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
transfere_inst(192);
for(i=0;i<12;i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j=2;
while(j<=user_nb)
{
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_carac(j+48);
transfere_carac('=');
transfere_inst(192);

for(i=(j-1)*12;i<(j*12) ;i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j++;
}

// end displaying the new database

        break;
}//case 4 code ends
case '5':// DELETE USER USER
{//case 5 code begins
// display current users nbr
```

```
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);

transfere_carac(user_nb+48);
delay_ms(1500);
j=0;
// display current data base
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
```

```
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='1';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
transfere_inst(192);
for(i=0;i<12;i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j=2;
while(j<=user_nb)
{
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_carac(j+48);
transfere_carac('=');
transfere_inst(192);
```

```
for(i=(j-1)*12;i<(j*12);i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j++;
}

// read user NUMBER to delete
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='L';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
```

```
transfere_inst(192);
caracter ='N';
transfere_carac(caracter);
caracter ='U';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
delay_ms(1500);
 ler_tecla();
  while (tecla=='*')
  {
      ler_tecla();
 }
delete_nb=tecla;
transfere_carac(delete_nb);
delay_ms(1000);
delete_nbr=delete_nb-'0';
del_pos=(delete_nbr-1)*12;
// delete the user 12 int = account (translate it) in user before
i=0;
while(i<12){
        for ( m = del_pos; m < (del_pos+12) - 1; m++) {
            T[m] = T[m + 12];
        }


        i++;
    }
length = length -12;
```

```
//

user_nb=user_nb-1;


// display the new data base
j=0;
// display current data base
transfere_inst(1);
transfere_inst(128);
caracter ='N';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='W';
transfere_carac(caracter);
caracter =' ';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='B';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
delay_ms(1500);
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
```

```
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='1';
transfere_carac(caracter);
caracter ='=';
transfere_carac(caracter);
caracter ='A';
transfere_carac(caracter);
caracter ='D';
transfere_carac(caracter);
caracter ='M';
transfere_carac(caracter);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
transfere_inst(192);
for(i=0;i<12;i++)
{ transfere_carac(T[i]);
}
delay_ms(1500);
j=2;
while(j<=user_nb)
{
transfere_inst(1);
transfere_inst(128);
caracter ='U';
transfere_carac(caracter);
caracter ='S';
transfere_carac(caracter);
caracter ='E';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
transfere_carac(j+48);
```

```
transfere_carac('=');
transfere_inst(192);

for(i=(j-1)*12;i<(j*12);i++)
{ transfere_carac(T[i]);
}
delay_ms(2000);
j++;
}
instrucao=1;
transfere_inst(instrucao);

          break;
}//case 5 code ends
           }//switch ends

         } //end if right user
// valid user card
//wrong entered pass

else
{
instrucao=1;
transfere_inst(instrucao);
instrucao=128;
transfere_inst(instrucao);
caracter ='I';
transfere_carac(caracter);
caracter ='N';
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='O';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='R';
transfere_carac(caracter);
caracter ='E';
```

```
transfere_carac(caracter);
caracter ='C';
transfere_carac(caracter);
caracter ='T';
transfere_carac(caracter);
delay_ms(1000);
instrucao=1;
transfere_inst(instrucao);

}//ends else


  }//END WHILE

}//END MAIN
```

ESQUEMA ELÉTRICO

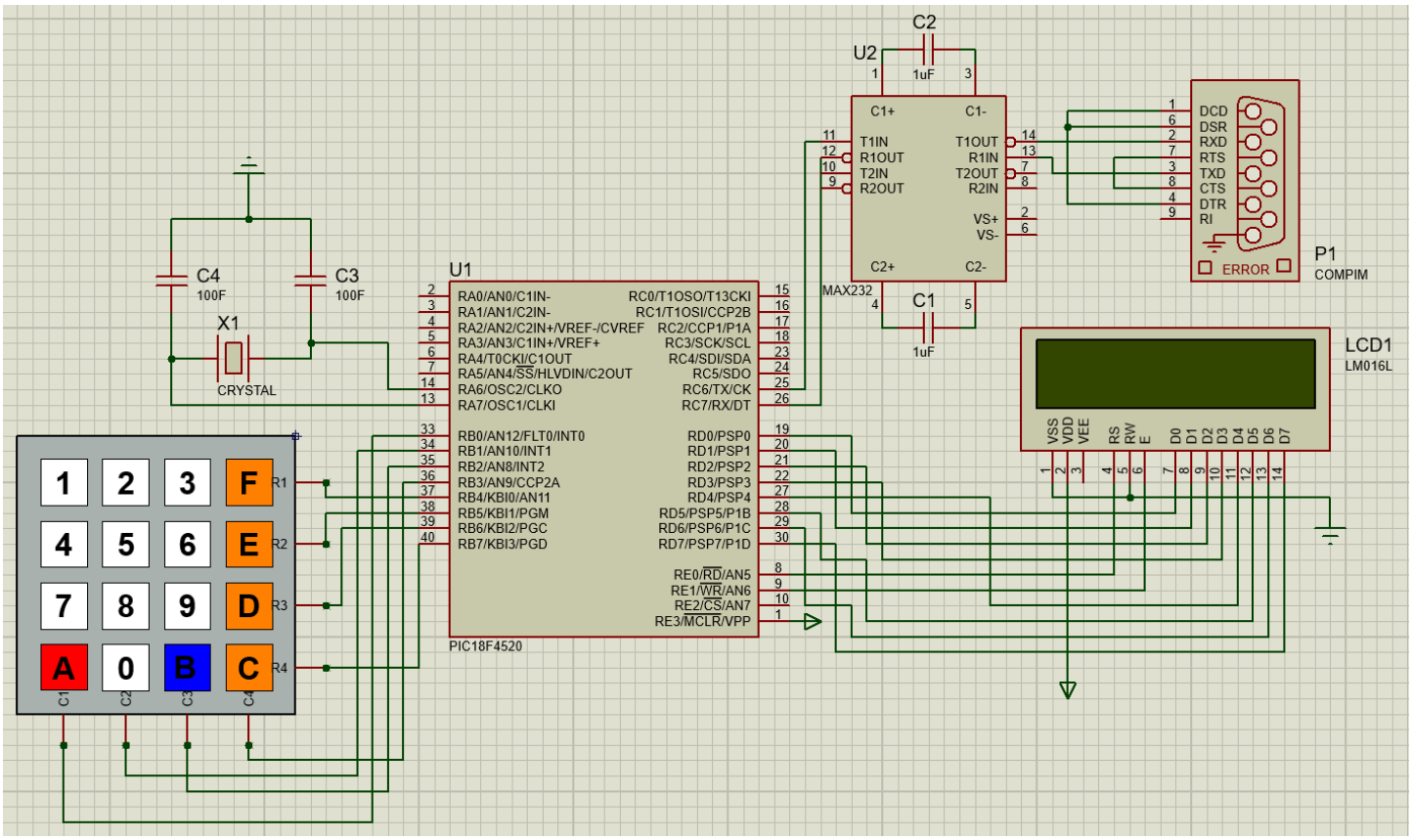


Figura 7 - Esquema Elétrico feito no Proteus 8 Professional