Projeto de Aplicação no Microcontrolador PIC16F877A Utilizando a Linguagem C "Bomba D'Água"

José Arthur Pereira Alves
Faculdade de Computação e Engenharia Elétrica - FACEEL
Universidade Federal do Sul e Sudeste do Pará - UNIFESSPA
Marabá, Brasil
arthurj167@unifesspa.edu.br

Luana Batista Araújo
Faculdade de Computação e Engenharia Elétrica - FACEEL
Universidade Federal do Sul e Sudeste do Pará - UNIFESSPA
Marabá, Brasil
lluanabatist@unifesspa.edu.br

Manoel Malon Costa de Moura
Faculdade de Computação e Engenharia Elétrica - FACEEL
Universidade Federal do Sul e Sudeste do Pará - UNIFESSPA
Marabá, Brasil
malloncosta@unifesspa.edu.br

***Resumo* —Um microcontrolador é pequeno dispositivo que apresenta um processador contido em um circuito integrado capaz de realizar operações lógicas e aritméticas, sendo amplamente utilizados em sistemas embarcados. Tendo em vista a importância da compreensão do seu funcionamento para o engenheiro da computação, através dos softwares MPLAB e PROTEUS, desenvolveu-se um projeto para simulação de um sensor de nível de água visando aplicação do microcontrolador PIC utilizando a linguagem de programação C.**

Palavras-chaves — microcontrolador, compreensão, projeto, programação.

I. INTRODUÇÃO

Microcontroladores são pequenos dispositivos (circuitos integrados) responsáveis por controlar e coordenar, circuitos eletrônicos por meio de algoritmos programáveis. Eles são utilizados, sobretudo, em sistemas embarcados, que produzem uma sequência de tarefas pré-estabelecidas, controladas pelos dispositivos em questão. Deste modo, é possível aplicá-los em diversos tipos de projetos, como controles de carros, controles remotos, funcionamento de eletrodomésticos, funcionamento de brinquedos, entre outros dispositivos automatizados.

A compreensão do funcionamento de microprocessadores e microcontroladores é essencial para preparar o aluno de engenharia da computação para a entendimento e execução de projetos de automação. Assim, com o objetivo de reforçar os conhecimentos acerca do seu funcionamento, bem como, de sua programação, um projeto utilizando o microcontrolador PIC16F877A foi elaborado.

O PIC16F877A é um microcontrolador desenvolvido pela empresa Microchip da família de 8 bits e núcleo de 14 bits, internamente possui encapsulado um microprocessador, a memória de programa e dados, a comunicação serial, o conversor analógico/digital, os geradores PWM, além de vários periféricos. Suas principais características são: frequência de operação (clock) até 20MHz, memória flash de programa (8K de palavras de 14 bits), memória RAM de 368 bytes, conjunto de instruções RISC, pinagem DIP de 40 pinos e alimentação de 2V a 5,5V.

II. METODOLOGIA

O objetivo do desenvolvimento deste projeto consistiu em empregar os conhecimentos adquiridos em sala de aula e nos experimentos realizados, utilizando a linguagem de programação C aplicada ao microcontrolador PIC16F877A. Logo, com o intuito de avaliar o aprendizado destes conhecimentos, foi solicitado aos alunos o desenvolvimento de uma aplicação prática, assim, os alunos referentes a este trabalho optaram pela elaboração de um sensor de nível de água.

O projeto teve como elementos centrais de apreciação a aplicação do microcontrolador PIC16F877A e a utilização dos softwares Proteus 8, programa que permite a criação de projetos eletrônicos, e MPLAB versão 5.15, ambiente de programação para desenvolvimento de aplicações e sistemas embarcados. Para melhor compreensão do programa desenvolvido, o trabalho foi dividido em duas partes principais detalhadas nos itens A e B abaixo.

A. Esquemático – Proteus

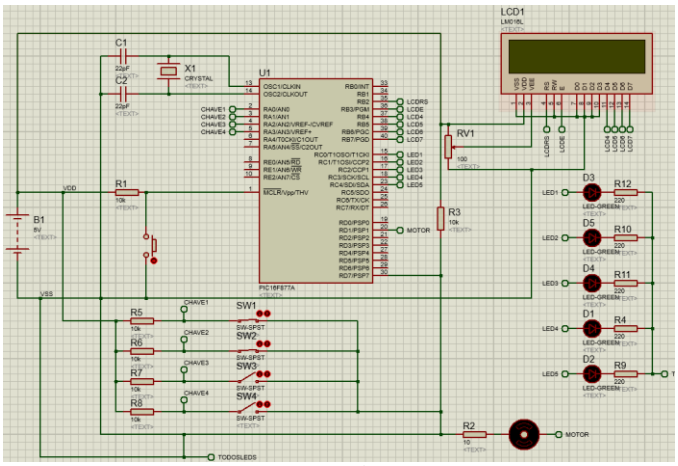


Figura 1 – Circuito feito no Proteus

O esquemático acima, apresenta o circuito do projeto em questão. O PIC16F877A foi o microcontrolador utilizado, este foi aplicado de tal forma que apresenta: um cristal oscilador conectado aos pinos OSC1 E OSC2; 5 leds com resistores de 220 Ohms, representando cada nível d'água, ligados nos pinos de RC0 a RC4; um motor conectado ao pino RD1, que é acionado sempre que a água atinge o nível mais baixo; 4 chaves ligadas aos pinos RA0 a RA3, que verificam o nível d'água; e por fim o display 16x2 ligado as portas RB2 a RB7, que exibe de forma intuitiva ao usuário os dados obtidos pelo microcontrolador.

B. *Código em C*

Código escrito no MPLAB, com o uso da linguagem C. Primeiramente define-se as seguintes configurações do programa:

```
1
2   /*
3    * File: projeto3.c
4    */
5
6   #define _XTAL_FREQ 8000000
7
8   #pragma config FOSC = HS
9   #pragma config WDTE = OFF
10  #pragma config PWRTE = OFF
11  #pragma config BOREN = ON
12  #pragma config LVP = OFF
13  #pragma config CPD = OFF
14  #pragma config WRT = OFF
15  #pragma config CP = OFF
16
17  #include <stdio.h>      //b
18  #include <stdlib.h>     //f
19  #include <stdbool.h>    //
20  #include <xc.h>     //Bibli
21
22  #include "lcd.h"    //bibli
23
```

Figura 2 – Configurações do programa

Nota-se que a única configuração que está ativada é a BOREN. Em seguida têm-se as bibliotecas que foram utilizadas no código, a biblioteca **xc.h** em especifico é usada para microcontroladores PIC, vale salientar também a biblioteca **lcd.h**, que é para utilização do LCD.

Em seguida, define-se uma sequência de comandos que se referem aos registradores, CMCON desativa todos os dois comparadores do PIC e ADCON1 estabelece que todos os pinos de AN0 a AN7 sejam configurados coo digitais, em TRISAT têm-se a configuração das entradas e saídas do PORTA (00001111) com RA0 a RA3 como entradas. Nos comandos PORT (A, B e C) inicializa-se as portas. E em TRIS (B, C e D) nota-se a configuração das entradas e saídas. Logo depois, é estabelecido RD2 que seta o lcd light (pino RD2) como 1, e o RD3 que seta o pino RD7 como 1. Na linha 42 há a conexão dos pinos de led no PIC (PORT, RS, EN, B4, B5, B6, B7), logo após, a inicialização do lcd.

```
24  int main ( void ) {
25
26      int i = 0;
27      int c = 16;
28      int b = 0;
29      CMCON = 0x07;           //Desativa os do
30      ADCON1 = 0x06;          //TOdos os pinos
31      TRISA = 0x0F;          // configuração d
32      PORTA = 0x00;           //inicialização
33      PORTD = 0x00;           //inicialização
34      PORTC = 0x00;           //inicialização
35      TRISB = 0x00;          //configuração do
36      TRISC = 0x00;          //configuração do
37      TRISD = 0x80;          //configuração do
38
39      RD2 = 1;    //setando lcd light (pino RD
40      RD7 = 1;    //setando o pino RD7 como 1
41
42      LCD lcd = { &PORTB, 2, 3, 4, 5, 6, 7 };
43
44      LCD_Init(lcd);  //inicialização do LCD
45
```

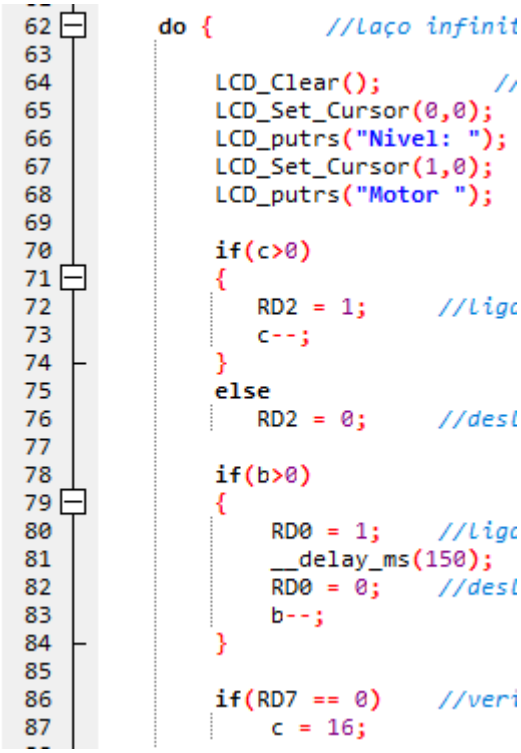
Figura 3 – Configurações dos registradores

Posteriormente, a função LDC_Clear() limpa o LCD para que LDC_SET_CURSOR() inicie o cursor nas coordenadas (0, 0), assim, o comando LCD_putrs enviará o string "Projeto" para o LCD. Logo após, o curso será iniciado nas coordenadas (0 ,1) e a string "Sensor de Nível" será enviada. Observa-se também um laço FOR que foi definido para mover o texto para a direita 16 vezes.

```
46      // Nome do Projeto
47      LCD_Clear();    //limpa o LCD
48      __delay_ms(500);  //delay de 500 m
49      LCD_Set_Cursor(0,0);    //Iniciand
50      LCD_putrs("Projeto");   //enviando
51      __delay_ms(200);        //delay de
52      LCD_Set_Cursor(1,0);    //Iniciand
53      LCD_putrs("Bomba D'Agua");      //
54      __delay_ms(500);    //delay de 500
55
56      for(i=0; i<15; i++) { // Move o te
57        LCD_Shift_Right();
58        __delay_ms(150);
59      }
60      i=0;
```

Figura 4 – Impressão da mensagem de inicialização

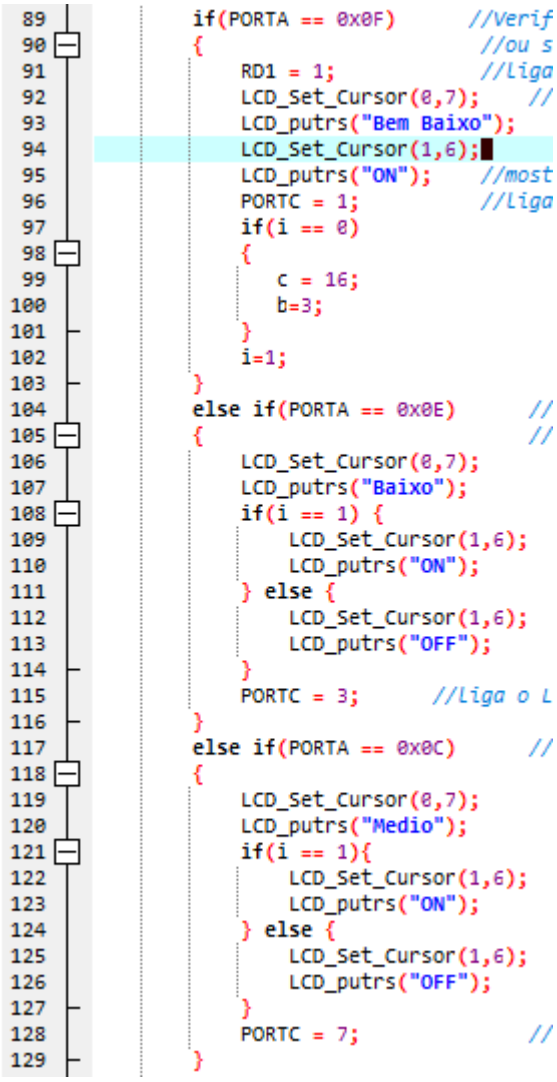
É estabelecido um laço infinito de repetição, dentro deste laço primeiramente limpa o LCD, e inicia o curso em (0, 0), em seguida envia-se a string "Nível: 0" para o lcd. Observa-se que na ordem, as mesmas etapas anteriores vão se repetir, iniciando o curso em (0, 1) é feito o despacho da string "Nível: 1" para o lcd. Posteriormente, têm-se uma séria de estrutura condicional if para ligar ou desligar o LED caso c seja maior que 0, ligar ou desligar o buzzer caso b seja maior que 1 e verificar se o pino RD7 está desligado.



```
62  do {        //laço infinit
63
64      LCD_Clear();          //
65      LCD_Set_Cursor(0,0);
66      LCD_putrs("Nivel: ");
67      LCD_Set_Cursor(1,0);
68      LCD_putrs("Motor ");
69
70      if(c>0)
71      {
72          RD2 = 1;      //liga
73          c--;
74      }
75      else
76          RD2 = 0;      //desl
77
78      if(b>0)
79      {
80          RD0 = 1;      //liga
81          __delay_ms(150);
82          RD0 = 0;      //desl
83          b--;
84      }
85
86      if(RD7 == 0)      //veri
87          c = 16;
```

Figura 5 – Laço de repetição (Parte I)

Logo depois, inicia-se uma série de verificações em PORTA utilizando a condicional if. Na linha 89 verifica-se se PORTA está em seu estado original, ou seja, nenhuma chave fechada, caso esteja o motor será ligado, o cursor será iniciado em (0, 7) e a string “Bem baixo” será enviada ao lcd, além de mostrar no display que o motor está ligado através do ON, e por fim, o LED D3 será ligado. A verificação seguinte exibirá “Baixo” no lcd caso PORTA esteja com a primeira chave fechada, o LED D3 e D5 serão ligados. Já na próxima, caso PORTA esteja com a segunda chave fechada, será exibido “Médio” e o LED D3, D5 e D4 ligarão.



```
89      if(PORTA == 0x0F)       //Verif
90      {                       //ou s
91          RD1 = 1;            //liga
92          LCD_Set_Cursor(0,7);    //
93          LCD_putrs("Bem Baixo");
94          LCD_Set_Cursor(1,6);
95          LCD_putrs("ON");    //most
96          PORTC = 1;          //liga
97          if(i == 0)
98          {
99              c = 16;
100             b=3;
101         }
102         i=1;
103     }
104     else if(PORTA == 0x0E)      //
105     {                           //
106         LCD_Set_Cursor(0,7);
107         LCD_putrs("Baixo");
108         if(i == 1) {
109             LCD_Set_Cursor(1,6);
110             LCD_putrs("ON");
111         } else {
112             LCD_Set_Cursor(1,6);
113             LCD_putrs("OFF");
114         }
115         PORTC = 3;      //liga o L
116     }
117     else if(PORTA == 0x0C)      //
118     {
119         LCD_Set_Cursor(0,7);
120         LCD_putrs("Medio");
121         if(i == 1){
122             LCD_Set_Cursor(1,6);
123             LCD_putrs("ON");
124         } else {
125             LCD_Set_Cursor(1,6);
126             LCD_putrs("OFF");
127         }
128         PORTC = 7;              //
129     }
```

Figura 6 – Laço de repetição (Parte II)

Dando continuidade, ao analisar a se a PORTA está com a terceira chave fechada, a tela terá os caracteres “Alto” e o LED D3, D5, D4 e D1 irão ligar. Por último, caso PORTA esteja com a quarta chave fechada será exibido “Cheio”, o motor será desligado e liga-se o LED D3, D5, D4, D1 e D2. Nota-se que seguidamente caso nenhuma das condicionais citadas anteriormente sejam verdadeiras o registrador PORTA será estabelecido com seu valor original. E ao finalizar o programa, retorna uma saída bem sucedida.

```
130             else if(PORTA == 0x08)      //
131             {
132                 LCD_Set_Cursor(0,7);
133                 LCD_putrs("Alto");
134                 if(i == 1){
135                     LCD_Set_Cursor(1,6);
136                     LCD_putrs("ON");
137                 } else {
138                     LCD_Set_Cursor(1,6);
139                     LCD_putrs("OFF");
140                 }
141                    PORTC = 15;          //
142             }
143             else if(PORTA == 0x00)     //Ve
144             {
145                 LCD_Set_Cursor(0,7);
146                 LCD_putrs("Cheio");
147                 LCD_Set_Cursor(1,6);
148                 LCD_putrs("OFF");
149                 RD1 = 0;               //Desl
150                 if(i == 1)
151                 {
152                   c = 16;
153                   b = 3;
154                 }
155                 i=0;
156                 PORTC = 31;          //liga
157             }
158             else                       //Cas
159                 PORTA = 0x0F;          //O re
160             __delay_ms(125);           //Del
161         }while(1);                 // Loop infinit
162
163         return (EXIT_SUCCESS);         //Fin
164     }
```

Figura 7 – Laço de repetição (Parte III)

III. RESULTADOS OBTIDOS

As figuras, a seguir, exibem os resultados da execução do programa. Nas Figuras 8 e 9, é possível verificar a impressão no display 16x2 do atual estado da caixa d'água e do motor, de tal forma que a primeira linha exibe o nível de água presente dentro da caixa d'água e a segunda linha mostra se o motor que bombeia a água está ligado ou não.

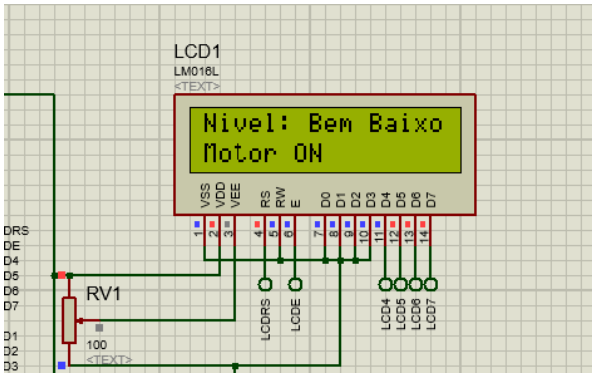


Figura 8 – Impressão do nível de água e status do motor

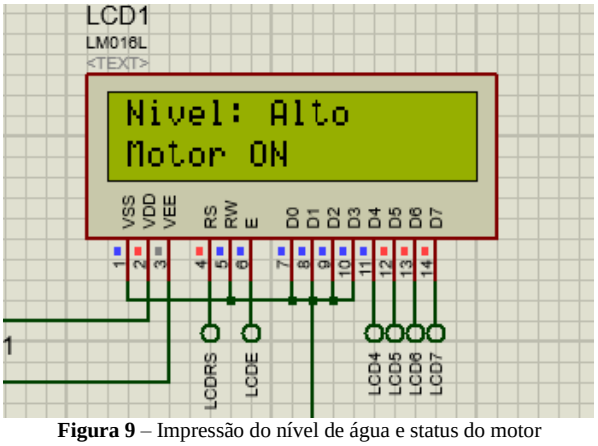


Figura 9 – Impressão do nível de água e status do motor

Já a Figura 10, abaixo, exibe o motor em funcionamento, este, é acionado após o nível de água atingir o seu nível mais baixo (Bem Baixo), ficando em funcionamento até que a água atinja o nível mais alto (Cheio).

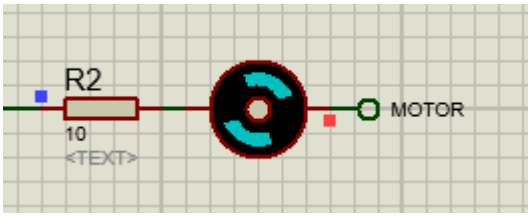


Figura 10 – Motor em funcionamento

A Figura 11 exibe uma outra forma de constatação do nível de água presente na caixa, nela 5 LEDs são apresentados representando cada nível de água de acordo com o número de LED acesos, sendo que, caso somente o LED1 esteja aceso, isso indica que o nível de água está bem baixo; caso o LED1 e o LED2 estejam acesos, então o nível de água está baixo; caso os LEDs 1, 2 e 3 estejam acesos, então o nível de água está médio; e assim sucessivamente.

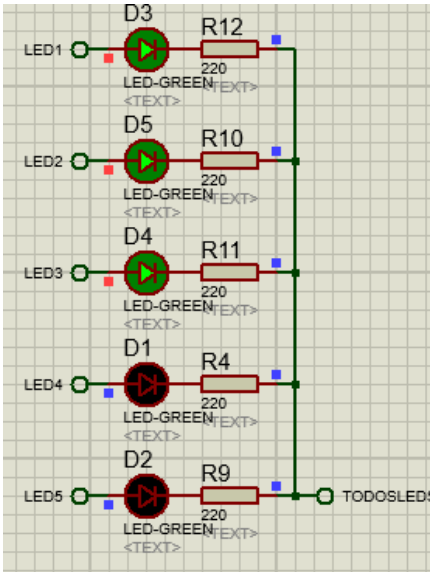


Figura 11 – LEDs representando o nível de água

Por fim, a Figura 12 retrata a situação em que a caixa é completamente cheia e o motor é então desligado.

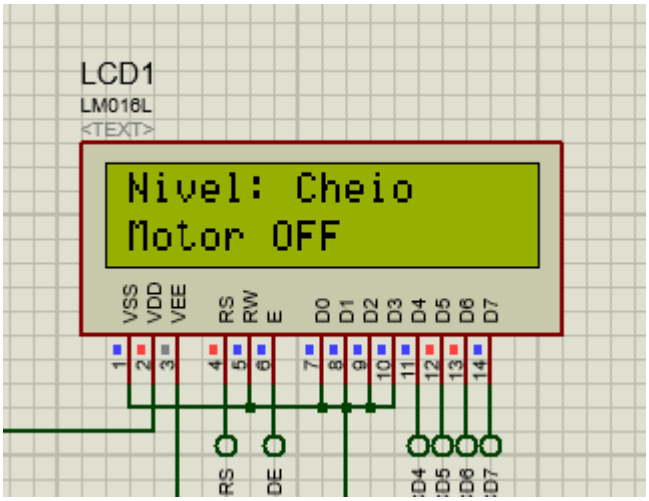


Figura 12 – Motor é desligado

IV. Conclusão

Conclui-se que os microcontroladores se mostram cada vez mais presentes nas mais diversas áreas de estudo da atualidade, como uma ferramenta bastante útil e eficiente para os mais diversos projetos de aplicação lógica, podendo também ser utilizado para fins didáticos, pois através dele pode-se colocar em prática conhecimentos teóricos. Tal afirmação pode ser comprovada através dos objetivos deste trabalho que foram alcançados com êxito, efetuando assim, a simulação do funcionamento de uma bomba d'água.

Referências

[1] PIC16F87XA Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers. U.S.A.: Microchip Technology Incorporated, 2003, 234p.

[2] AGKOPIAN, Manolis. Pic-xc8-lcd-library. Github, 2021. Disponível em: <https://github.com/magkopian/pic-xc8-lcd-library>. Acesso em: 29 de novembro de 2021.

[3] MICROCONTROLADOR PIC16F877A. Baú da Eletrônica, 2021. Disponível em: <https://www.baudaeletronica.com.br/microcontrolador-pic16f877a.html>. Acesso em: 30 de novembro de 2021.

Apêndice (Código em C)

A. bomba.c

```
#define _XTAL_FREQ 8000000

#pragma config FOSC = HS   // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF  // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = ON  // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF   // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)
#pragma config CPD = OFF   // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF   // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF    // Flash Program Memory Code Protection bit (Code protection off)

#include <stdio.h>      //biblioteca para entrada e saida dos dispositivos de E/S
#include <stdlib.h>     //funções envolvendo alocação de memória, controle de processos, conversões e outras.
#include <stdbool.h>    // usada para manipular variáveis lógicas, como verdadeiro e falso
#include <xc.h>     //Biblioteca para microcontroladores PIC

#include "lcd.h"    //biblioteca para utilização do LCD

int main ( void ) {

    int i = 0;
    int c = 16;
    int b = 0;
    CMCON = 0x07;          //Desativa os dois comparadores do PIC
    ADCON1 = 0x06;         //TOdos os pinos de AN0 a AN7 são configurados como digitais
    TRISA = 0x0F;          // configuração das entradas e saídas do PORTA (00001111) com RA0 a RA3 como entradas
    PORTA = 0x00;          //inicialização do PORTA
    PORTD = 0x00;          //inicialização do PORTD
    PORTC = 0x00;          //inicialização do PORTC
    TRISB = 0x00;          //configuração das entradas e saídas do PORTB com todos como saidas
    TRISC = 0x00;          //configuração das entradas e saídas do PORTC com todos como saidas
    TRISD = 0x80;          //configuração das entradas e saídas do PORTD (10000000) com todos como saídas, exceto RD7

    RD2 = 1;    //setando lcd light (pino RD2) como 1
    RD7 = 1;    //setando o pino RD7 como 1

    LCD lcd = { &PORTB, 2, 3, 4, 5, 6, 7 };    //PORT, RS, EN, B4, B5, B6, B7 (conexão dos pinos do led no PIC)

    LCD_Init(lcd);  //inicialização do LCD

    // Nome do Projeto
    LCD_Clear();    //limpa o LCD
    __delay_ms(500);  //delay de 500 milissegundos (1/2 segundo)
    LCD_Set_Cursor(0,0);    //Iniciando o cursor nas coordenadas 0, 0
```

```
   LCD_putrs("Projeto");   //enviando a string "Projeto" para o
LCD
   __delay_ms(200);        //delay de 200 milissegundos
   LCD_Set_Cursor(1,0);    //Iniciando o cursor nas
coordenadas 1, 0
   LCD_putrs("Bomba D'Agua");      //enviando a string
"Sensor de Nível" para o LCD
   __delay_ms(500);    //delay de 500 milissegundos

   for(i=0; i<15; i++) { // Move o texto para a direita 16 vezes
     LCD_Shift_Right();
     __delay_ms(150);
   }
   i=0;

   do {       //laço infinito

      LCD_Clear();        //Limpa o LCD
      LCD_Set_Cursor(0,0);    //Cursor inicia em 0, 0
      LCD_putrs("Nivel: ");   //enviando a string "Nível: 0"
para o LCD
      LCD_Set_Cursor(1,0);
      LCD_putrs("Motor ");

      if(c>0)
      {
        RD2 = 1;     //liga o LED lcd light
        c--;
      }
      else
        RD2 = 0;     //desliga o LED lcd light

      if(b>0)
      {
         RD0 = 1;    //liga o buzzer
         __delay_ms(150);    //delay de 150 milissegundos
         RD0 = 0;    //desliga o buzzer
         b--;
      }

      if(RD7 == 0)    //verifica se o pino RD7 está desligado
         c = 16;

      if(PORTA == 0x0F)      //Verifica se PORTA está em seu
estado original (nenhuma chave fechada)
      {                    //ou seja 00001111
         RD1 = 1;           //liga o motor
         LCD_Set_Cursor(0,7);    //Inicia o curso em 0, 7
         LCD_putrs("Bem Baixo");     //Envia a String "Bem
baixo" no LCD
         LCD_Set_Cursor(1,6);
         LCD_putrs("ON");    //mostra no display que o motor
está ligado
         PORTC = 1;         //liga o LED D3 - (00000001
significa 7 em binário)
         if(i == 0)
         {
           c = 16;
           b=3;
         }
         i=1;
      }
      else if(PORTA == 0x0E)      //Verifica se PORTA está
com a primeira chave fechada (0x0E é 00001110)
      {                    //o ultimo bit (foi de 1 para 0) significa
que a chave foi fechada
         LCD_Set_Cursor(0,7);
         LCD_putrs("Baixo");
         if(i == 1) {
           LCD_Set_Cursor(1,6);
           LCD_putrs("ON");
         } else {
           LCD_Set_Cursor(1,6);
           LCD_putrs("OFF");
         }
         PORTC = 3;      //liga o LED D3 e D5 (3 em decimal
significa 00000011 em binario)
      }
      else if(PORTA == 0x0C)      //Verifica se PORTA está
com a segunda chave fechada (0x0C é 00001100)
      {
         LCD_Set_Cursor(0,7);
         LCD_putrs("Medio");
         if(i == 1){
           LCD_Set_Cursor(1,6);
           LCD_putrs("ON");
         } else {
           LCD_Set_Cursor(1,6);
           LCD_putrs("OFF");
         }
         PORTC = 7;             //liga o LED D3, D5 e D4 (7 em
decimal significa 00000111 em binario)
      }
      else if(PORTA == 0x08)      //Verifica se PORTA está
com a terceira chave fechada (0x08 é 00001000)
      {
         LCD_Set_Cursor(0,7);
         LCD_putrs("Alto");
         if(i == 1){
           LCD_Set_Cursor(1,6);
           LCD_putrs("ON");
         } else {
           LCD_Set_Cursor(1,6);
           LCD_putrs("OFF");
         }
          PORTC = 15;         //liga o LED D3, D5, D4 e D1
(15 em decimal significa 00001111 em binario)
      }
      else if(PORTA == 0x00)    //Verifica se PORTA está
com a quarta chave fechada (0x00 de hexadecimal é 00000000
binario)
      {
         LCD_Set_Cursor(0,7);
         LCD_putrs("Cheio");
         LCD_Set_Cursor(1,6);
         LCD_putrs("OFF");
         RD1 = 0;           //Desliga o motor
         if(i == 1)
```

```
        {
          c = 16;
          b = 3;
        }
        i=0;
        PORTC = 31;       //liga o LED D3, D5, D4, D1 e D2
(31 em decimal significa 00011111 em binario)
      }
      else                //Caso nenhuma das condicionais
anteriores sejam verdadeiras
        PORTA = 0x0F;     //O registrador PORTA é setado
com seu valor original
      __delay_ms(125);    //Delay de 125 milissegundos
}while(1);          // Loop infinito

   return (EXIT_SUCCESS);     //Finalização do programa,
retornando saída bem sucedida
}
```

B. lcd.h

```
#ifndef LCD_H
#define  LCD_H

#ifdef    __cplusplus
extern "C" {
#endif

typedef struct {
   volatile unsigned char*  PORT;  // Pointer to the LCD port
e.g &PORTC
   unsigned RS :3;               // The RS bit of the LCD PORT
e.g. 2
   unsigned EN :3;               // The EN bit of the LCD PORT
e.g. 3
   unsigned D4 :3;               // The D4 bit of the LCD PORT
e.g. 4
   unsigned D5 :3;               // The D5 bit of the LCD PORT
e.g. 5
   unsigned D6 :3;               // The D6 bit of the LCD PORT
e.g. 6
   unsigned D7 :3;               // The D7 bit of the LCD PORT
e.g. 7
} LCD;

extern LCD lcd;

// Macros that correspond to LCD commands
#define LCD_Clear() LCD_Cmd(0x01)
#define LCD_Decrement_Cursor() LCD_Cmd(0x04)
#define LCD_Increment_Cursor() LCD_Cmd(0x05)
#define LCD_Shift_Display_Right() LCD_Cmd(0x06)
#define LCD_Shift_Display_Left() LCD_Cmd(0x07)
#define LCD_Shift_Right() LCD_Cmd(0x1C)
#define LCD_Shift_Left() LCD_Cmd(0x18)

// Sets the LCD cursor position
#define LCD_Set_Cursor( x, y )          \
do {                                    \
   if ( x == 0 ) {                      \
      LCD_Cmd(0x80 + y);                \
   }                                    \
   else if ( x == 1 ) {                 \
      LCD_Cmd(0xC0 + y);                \
   }                                    \
   else if ( x == 2 ) {                 \
      LCD_Cmd(0x94 + y);                \
   }                                    \
   else if ( x == 3 ) {                 \
      LCD_Cmd(0xD4 + y);                \
   }                                    \
} while ( false )

// [Internal use only] Sets the display on/off, the cursor on/off
and if it will blink or not
#define LCD_Display( on, cursor, blink ) \
do {                                     \
   unsigned char cmd = 0x08;             \
                                         \
   if ( on == true ) {                   \
      cmd |= 1 << 2;                     \
   }                                     \
                                         \
   if ( cursor == true ) {               \
      cmd |= 1 << 1;                     \
   }                                     \
                                         \
   if ( blink == true ) {                \
      cmd |= 1;                          \
   }                                     \
                                         \
   LCD_Cmd(cmd);                         \
} while ( false )

// [Internal use only] Sends a command to the LCD
#define LCD_Cmd( c )                    \
do {                                    \
   LCD_Write( (c & 0xF0) >> 4 );        \
   LCD_Write( c & 0x0F);                \
} while ( false )

// Initializes the LCD (See LCD struct)
bool LCD_Init ( LCD display );

// Prints a character on the LCD
void LCD_putc ( char a );

// Prints a string on the LCD that is allocated on the RAM
void LCD_puts ( char *a );

// Prints a string on the LCD that is allocated on the program
memory
void LCD_putrs ( const char *a );

// [Internal use only] Write a byte to the LCD
void LCD_Write ( unsigned char c );

// [Internal use only] Outputs the data to the LCD Port
```

```
void LCD_Out ( char a );

#ifdef     __cplusplus
}
#endif

#endif    /* LCD_H */
```

C. lcd.c

```
#define _XTAL_FREQ 8000000

#include <stdbool.h>
#include <xc.h>
#include "lcd.h"

LCD lcd;

void LCD_Out ( char c ) {

   if ( c & 1 ){
      *(lcd.PORT) |= 1 << lcd.D4;
   }
   else {
       *(lcd.PORT) &= ~(1 << lcd.D4);
   }

   if ( c & 2 ) {
      *(lcd.PORT) |= 1 << lcd.D5;
   }
   else {
      *(lcd.PORT) &= ~(1 << lcd.D5);
   }

   if ( c & 4 ) {
      *(lcd.PORT) |= 1 << lcd.D6;
   }
   else {
      *(lcd.PORT) &= ~(1 << lcd.D6);
   }

   if ( c & 8 ) {
       *(lcd.PORT) |= 1 << lcd.D7;
   }
   else {
       *(lcd.PORT) &= ~(1 << lcd.D7);
   }

}

void LCD_Write ( unsigned char c ) {

   *(lcd.PORT) &= ~(1 << lcd.RS); // => RS = 0
   LCD_Out(c);

   *(lcd.PORT) |= 1 << lcd.EN;    // => E = 1
   __delay_ms(4);
   *(lcd.PORT) &= ~(1 << lcd.EN); // => E = 0
}

bool LCD_Init ( LCD display ) {

   lcd = display;

   /*
    * TODO:
    * The function should clear only the appropriate bits, not
the whole PORT
    *
    * TODO:
    * "#if defined" needs to support more microcontrollers that
have PORTD and PORTE
    */
   if ( lcd.PORT == &PORTA ) {
      TRISA = 0x00;
   }
   else if ( lcd.PORT == &PORTB ) {
      TRISB = 0x00;
   }
   else if ( lcd.PORT == &PORTC ) {
      TRISC = 0x00;
   }
   #if defined(_16F877) || defined(_16F877A)
   else if ( lcd.PORT == &PORTD ) {
      TRISD = 0x00;
   }
   else if ( lcd.PORT == &PORTE ) {
      TRISE = 0x00;
   }
   #endif
   else {
      return false;
   }

   // Give some time to the LCD to start function properly
   __delay_ms(20);

   // Send reset signal to the LCD
   LCD_Write(0x03);
   __delay_ms(5);
   LCD_Write(0x03);
   __delay_ms(16);
   LCD_Write(0x03);

   // Specify the data lenght to 4 bits
   LCD_Write(0x02);

   // Set interface data length to 8 bits, number of display lines
to 2 and font to 5x8 dots
   LCD_Cmd(0x28);

   // Set cursor to move from left to right
   LCD_Cmd(0x06);

   LCD_Display(true, false, false); // Turn on display and set
cursor off
```

```
    LCD_Clear();

    return true;
}

void LCD_putc ( char c ) {

   *(lcd.PORT) |= 1 << lcd.RS;   // => RS = 1
    LCD_Out((c & 0xF0) >> 4);    //Data transfer

    *(lcd.PORT) |= 1 << lcd.EN;
    __delay_us(40);
    *(lcd.PORT) &= ~(1 << lcd.EN);

    LCD_Out(c & 0x0F);

    *(lcd.PORT) |= 1 << lcd.EN;
    __delay_us(40);
    *(lcd.PORT) &= ~(1 << lcd.EN);

}

void LCD_puts ( char *a ) {

    for ( int i = 0; a[i] != '\0'; ++i ) {
        LCD_putc(a[i]);
    }

}

void LCD_putrs ( const char *a ) {

    for ( int i = 0; a[i] != '\0'; ++i ) {
        LCD_putc(a[i]);
    }

}
```