Universidade Federal de Itajubá - UNIFEI

Projeto final de programação embarcada – Balança digital

Gabriel Jun Ito - 2019010462

ITAJUBÁ

2021

Universidade Federal de Itajubá - UNIFEI

Projeto final de programação embarcada – Balança digital

Gabriel Jun Ito - 2019010462

Relatório solicitado pelo professor Otávio de Souza Martins Gomes, para avaliação das disciplinas de ECOP04 – Programação Embarcada e ECOP14 – Laboratório de Programação Embarcada.

ITAJUBÁ

2021

Introdução

O microcontrolador simulará o funcionamento de uma balança digital, ela foi desenvolvida utilizando o MPLab X IDE, XC8 e o PICSimLab (com o PICGenios + PIC18F4520).

Objetivo e dificuldades

A balança digital irá medir o peso e a altura do usuário, e assim calcular o IMC (Índice de massa corporal) e fornecer o grau do IMC do usuário de maneira simples e alertar o usuário caso seu IMC esteja fora das medidas consideradas saudáveis.

Uma das dificuldades encontradas durante o desenvolvimento do projeto foi a utilização de diversas componentes da placa ao mesmo tempo e a adaptação e incrementação das bibliotecas que foram disponibilizadas pelo professor da disciplina.

Funcionalidades Utilizadas

-Display LCD

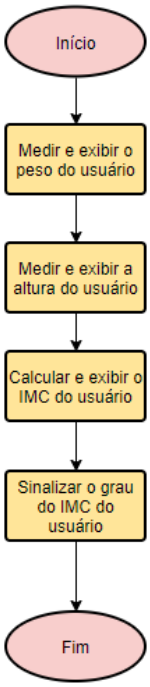
-Display de 7-segmentos

-LEDs

-Cooler (Ventoinha)

-Teclas

Fluxograma da balança digital:



Código

O código do projeto foi escrito e organizado na plataforma MPLab X IDE utilizando o compilador XC8 e executado na plataforma PICSimLab (PicGenios + PIC18F4520) por meio do arquivo gerado pelo MPLab X (.hex).

-main.c

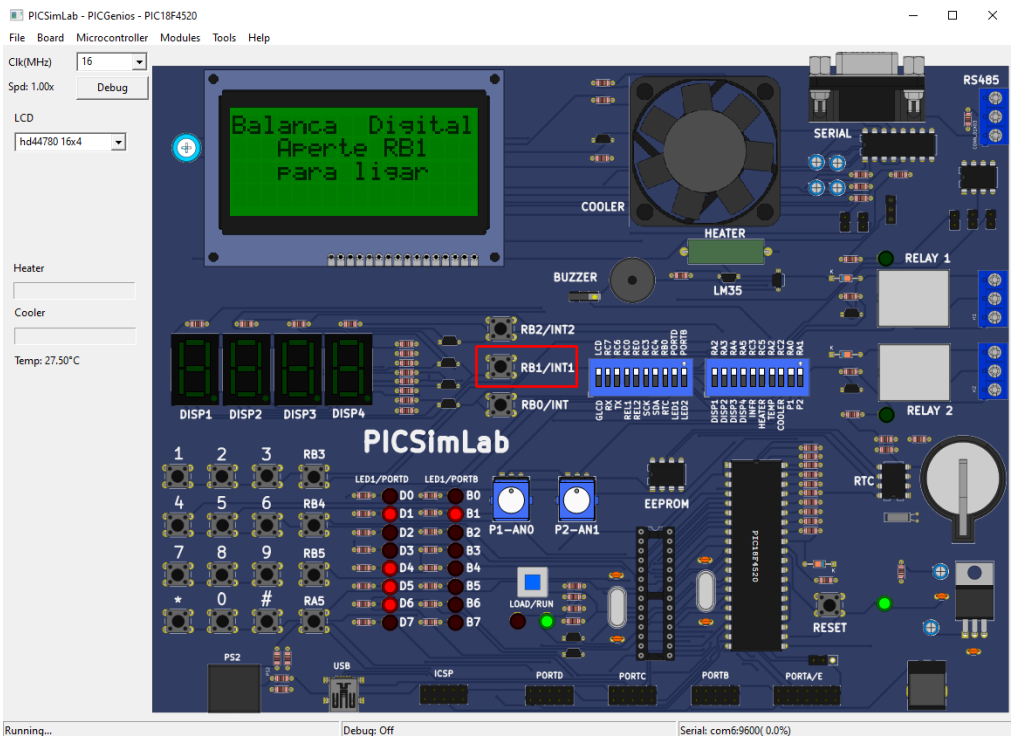
```
1  //Nome: Gabriel Jun Ito
2  //Matrícula: 201901462
3  #include <pic18f4520.h>
4  #include "bits.h"
5  #include "config.h"
6  #include "io.h"
7  #include "keypad.h"
8  #include "lcd.h"
9  #include "pwm.h"
10 #include "ssd.h"
11 #include "balanca.h"
12 #include "adc.h"
13 #include "atraso.h"
14
15 void main(void) {
16     //Inicializacao de variaveis e funcionalidades
17     int peso = 743;   //74,3kg
18     int altura = 180; //1,80m
19     TRISA = 0xC3;
20     TRISB = 0x03;
21     TRISC = 0x01;
22     TRISD = 0x00;
23     TRISE = 0x00;
24     ADCON1 = 0x06;
25     TRISB = 0x03;
26     kpInit();
27     lcdInit();
28     pwmInit();
29     adc_init();
30     TRISCbits.TRISC7 = 1;
31     TRISCbits.TRISC6 = 0;
32     PORTB = 0;
```

Linhas 3-13: Inclusão das bibliotecas.

Linhas 15-32: Inicialização de variáveis e funcionalidades que serão utilizadas.

```
34      //Menu de inicio
35      lcdPosition(1, 0);
36      lcd_str("Balanca  Digital");
37      lcdPosition(2, 3);
38      lcd_str("Aperte RB1");
39      lcdPosition(3, 3);
40      lcd_str("para ligar");
41      while (bitTst(PORTB, 1)); // Botao que inicia o programa
42
43      //peso do usuario
44      lcdCommand(L_CLR);
45      lcdPosition(4, 0);
46      lcd_str("Pesando...");
47      medidorPeso(peso);
48      ledp();
49      atraso_ms(2000);
50
51      //altura do usuario
52      lcdCommand(L_CLR);
53      menu();
54      lcdPosition(4, 0);
55      lcd_str("Medindo altura..");
56      medidorAltura(altura);
57      ledp();
58      atraso_ms(2000);
59
```

Linhas 24-41: Menu que aguarda o pressionamento do botão RB1 para o início do funcionamento da balança digital.



Indicação da localização do botão RB1.

Linhas 44-49: Simula a medição de peso de um usuário em uma balança digital e exibe uma interface com os valores das medidas.

Linhas 52-58: Simula a medição de altura de um usuário em uma balança digital e exibe uma interface com os valores das medidas.

```
60          //imc do usuario
61          lcdCommand(L_CLR);
62          menu();
63          lcdPosition(4, 0);
64          lcd_str("Calculando IMC...");
65          setIMC(peso, altura);
66         // pwmSet1(99);
67          atraso_ms(3000);
68          ledp();
69          menu();
70         // pwmSet1(0);
71
72          for (;;) {
73              ledp();
74          }
```

Linhas 61-69: Calculo do IMC do usuário e exibição das informações no display LCD.
Linhas 72-74: Fim do programa, entra em modo espera e continua exibindo as informações do usuário.

-balanca.h

```
1  //Nome: Gabriel Jun Ito
2  //Matrícula: 201901462
3  #ifndef BALANCA_H
4  #define BALANCA_H
5
6  #include "balanca.h"
7  #include <pic18f4520.h>
8  #include "config.h"
9  #include "bits.h"
10 #include "ssd.h"
11 #include "keypad.h"
12 #include "lcd.h"
13 #include "pwm.h"
14
15 void menu (void);
16
17 void setIMC(int, int);
18
19 void medidorPeso(int);
20
21 void medidorAltura(int);
22
23 void itoa(unsigned int, char*);
24
25 int roundNo(float num);
26
27 void ledp (void);
28 #endif  /* BALANCA_H */
29
```

Declaração das funções de balanca.c e inclusão das bibliotecas:

void menu (void) – Exibe o menu de informações do usuário: peso, altura e IMC.

Void setIMC (int, int) – Recebe a altura e o peso, calcula o IMC e passa o valor para a variável.

void medidorPeso (int) – Recebe o peso do usuário e simula o processo de medição de peso.

void medidorAltura (int) – Recebe a altura do usuário e simula o processo de medição de altura.

void itoa (unsigned int, char*) – Responsável pela adaptação de valores int para char*, permitindo a impressão de valores no display LCD.

void roundNo (float) – Arredonda um valor float para um int.
void ledp (void) – Pisca os LEDs de forma sequencial para sinalizar que alguma informação está sendo medida ou processada.

-balanca.c

```
1  //Nome: Gabriel Jun Ito
2  //Matrícula: 201901462
3  #include "balanca.h"
4  #include <pic18f4520.h>
5  #include "config.h"
6  #include "bits.h"
7  #include "ssd.h"
8  #include "keypad.h"
9  #include "lcd.h"
10 #include "pwm.h"
11
12 int peso = 0;
13 int altura = 0;
14 int imc = 0;
```

Linhas 3-14: Inclusão das bibliotecas e inicialização de variáveis.

```
16  void menu(void) {
17      char str[6];
18      float fImc = 0;
19      lcdCommand(L_CLR);
20
21      //mostra no lcd o peso
22      lcdPosition(1, 0);
23      lcd_str("Peso:");
24      itoa(peso * 10, str);
25      lcdData(str[1]);
26      lcdData(str[2]);
27      lcdData(',');
28      lcdData(str[3]);
29      lcd_str("kg");
30
31      //mostra no lcd a altura
32      lcdPosition(2, 0);
33      lcd_str("Altura:");
34      itoa(altura * 10, str);
35      lcdData(str[1]);
36      lcdData(',');
37      lcdData(str[2]);
38      lcdData(str[3]);
39      lcd_str("m");
40
```

Linhas 22-29: Posicionamento do cursor do display na primeira linha e coluna (lcdPosition (1,0))seguida da impressão do valor do peso, no qual foi adaptado pela função itoa para permitir a impressão.

Linhas 32-39: Posicionamento do cursor do display na segunda linha e primeira coluna seguida da impressão do valor da altura, na qual foi adaptada pela função itoa para permitir a impressão.

```
41      //mostra no lcd o imc
42      lcdPosition(3, 0);
43      lcd_str("IMC:");
44      itoa(imc, str);
45      lcdData(str[1]);
46      lcdData(str[2]);
47      lcdData(',');
48      lcdData(str[3]);
49      lcdData(str[4]);
50      lcdData(str[5]);
```

Linhas 41-50: Posicionamento do cursor do display na terceira linha e primeira coluna seguida da impressão do valor do IMC, no qual foi adaptado pela função itoa para permitir a impressão.

```
52          if(imc != 0){
53              fImc = imc/100;
54              if(fImc < 17){
55                  lcdPosition(4, 0);
56                  pwmSet1(40);
57                  lcd_str("Mt abaixoDoPeso");
58              }
59              if(fImc >= 17 && fImc < 18.5){
60                  lcdPosition(4, 0);
61                  pwmSet1(65);
62                  lcd_str("Abaixo do peso");
63              }
64              if(fImc >= 18.5 && fImc < 25){
65                  lcdPosition(4, 0);
66                  pwmSet1(99);
67                  lcd_str("Peso normal");
68              }
69              if(fImc >= 25 && fImc < 30){
70                  lcdPosition(4, 0);
71                  pwmSet1(65);
72                  lcd_str("Acima do peso");
73              }
74              if(fImc >= 30 && fImc < 35){
75                  lcdPosition(4, 0);
76                  pwmSet1(40);
77                  lcd_str("Obesidade I");
78              }
79              if(fImc >= 35 && fImc < 40){
80                  lcdPosition(4, 0);
81                  pwmSet1(25);
82                  lcd_str("Obesidade II");
83              }
84              if(fImc > 40){
85                  lcdPosition(4, 0);
86                  pwmSet1(10);
87                  lcd_str("Obesidade III");
88              }
```

Linhas 52-88: Verifica se o valor do IMC é válido e verifica em qual faixa o usuário se encontra, e com base no nível saúde, o cooler será ligado em uma potência máxima caso o usuário esteja em condição saudável, e ligando em potências menores de acordo com outros níveis de IMC, e ao final exibe o nome do grau do usuário.

```
92      //calcula o imc
93      void setIMC(int p, int a){
94          float aux = (float)p / ((float) a*(float)a);
95          imc = roundNo(aux*100000);
96      }
97
98      //arrendonda um float para int
99      int roundNo(float num){
100         return num < 0 ? num - 0.5 : num + 0.5;
101     }
```

Linhas 92-96: Calcula o valor do IMC com o arredondamento e adaptação do valor para exibição.

Linhas 98-101: Arredonda um valor float para um int, possibilitando a utilização da função itoa.

```
103     //mede o peso do usuario
104     void medidorPeso(int p) {
105         peso = p;
106         volatile unsigned int tempo;
107         int cont;
108         int cont2;
109         ledp();
110         ssdInit();
111         for (;;) {
112             cont++;
113             ssdDigit(((cont / 100) % 10), 0);
114             ssdDigit(((cont / 90) % 10), 1);
115             ssdDigit(((cont / 50) % 10), 2);
116             ssdDigit(((cont / 25) % 10), 3);
117             ssdUpdate();
118             for (tempo = 0; tempo < 1000; tempo++);
119             if (cont >= p) {
120                 break;
121             }
122         }
123         for (cont2 = 0; cont2 < 750; cont2++) {
124             ssdDigit(((p / 100) % 10), 0);
125             ssdDigit(((p / 10) % 10), 1);
126             ssdDigit(((p / 1) % 10), 2);
127             ssdDigit(((p / 1) % 10), 3);
128             ssdUpdate();
129             for (tempo = 0; tempo < 1000; tempo++);
130         }
131         ledp();
132         bitClr(PORTA, 2);
133         bitClr(PORTA, 3);
134         bitClr(PORTA, 4);
135         bitClr(PORTA, 5);
136     }
```

Linhas 104-136: Cria o efeito visual de uma balança digital medindo o peso do usuário.

```
138     //mede a altura do usuario
139     void medidorAltura(int a) {
140         altura = a;
141         volatile unsigned int tempo;
142         int cont;
143         int cont2;
144         ledp();
145         ssdInit();
146         for (;;) {
147             cont++;
148             ssdDigit(((cont / 100) % 10), 0);
149             ssdDigit(((cont / 80) % 10), 1);
150             ssdDigit(((cont / 50) % 10), 2);
151             ssdDigit(((cont / 25) % 10), 3);
152             ssdUpdate();
153             for (tempo = 0; tempo < 1000; tempo++);
154             if (cont >= a) {
155                 break;
156             }
157         }
158         for (cont2 = 0; cont2 < 750; cont2++) {
159             ssdDigit(((a / 100) % 10), 0);
160             ssdDigit(((a / 10) % 10), 1);
161             ssdDigit(((a / 1) % 10), 2);
162             ssdDigit(((a / 1) % 10), 3);
163             ssdUpdate();
164             for (tempo = 0; tempo < 1000; tempo++);
165         }
166         ledp();
167         bitClr(PORTA, 2);
168         bitClr(PORTA, 3);
169         bitClr(PORTA, 4);
170         bitClr(PORTA, 5);
171     }
```

Linhas 139-171: Cria o efeito visual de uma balança digital medindo a altura do usuário.

```
173     //converte um uint para char*
174     void itoa(unsigned int val, char* str) {
175         str[0] = (val / 10000) + 0x30;
176         str[1] = ((val % 10000) / 1000) + 0x30;
177         str[2] = ((val % 1000) / 100) + 0x30;
178         str[3] = ((val % 100) / 10) + 0x30;
179         str[4] = (val % 10) + 0x30;
180         str[5] = 0;
181     }
```

Linhas 174-181: Converte um unsigned int para char* permitindo a impressão do valor no display LCD.

```
183     //faz os led`s piscarem
184   void ledp(void) {
185       TRISD = 0x00;
186       PORTD = 0x00;
187       float i;
188       for (i = 0; i < 1000; i++);
189       PORTD = 0b00000001;
190       for (i = 0; i < 1000; i++);
191       PORTD = 0b00000010;
192       for (i = 0; i < 1000; i++);
193       PORTD = 0b00000100;
194       for (i = 0; i < 1000; i++);
195       PORTD = 0b00001000;
196       for (i = 0; i < 1000; i++);
197       PORTD = 0b00010000;
198       for (i = 0; i < 1000; i++);
199       PORTD = 0b00100000;
200       for (i = 0; i < 1000; i++);
201       PORTD = 0b01000000;
202       for (i = 0; i < 1000; i++);
203       PORTD = 0b10000000;
204       for (i = 0; i < 1000; i++);
205   }
```

Linhas 184-205: Cria o efeito visual dos LEDs acenderem sequencialmente.

Conclusão

Ao fim deste projeto, nota-se a versatilidade que os sistemas embarcados podem ter, além de compactos, permitem a inclusão de interface para os usuários utilizando as funcionalidades da placa.

Os códigos mostrados neste relatório podem ser acessados e baixados no link:

https://github.com/GJIto/ecop04_14_projeto_final

O código em funcionamento pode ser visto no link:

https://drive.google.com/file/d/14rdJdl7ZdE1fMjs8g-WU8dt5MboRba5L/view?usp=sharing

Softwares Utilizados

MPLab X IDE - https://www.microchip.com/mplab/mplab-x-ide

Compilador XC8 - https://www.microchip.com/mplab/compilers

PICSimLab - https://sourceforge.net/projects/picsim/files/picsim/picsimlab-0.7.5/

Bibliotecas - https://sites.google.com/site/rmaalmeida/unifei/downloads-1/ecop14