Centro Federal de Educação Tecnológica de Minas Gerais

Departamento de Eletrônica e Biomédica

Refrigerador com Placa Peltier

Felipe Reis Campanha Ribeiro

Caio Vilquer Carvalho

Belo Horizonte

2019

Sumário

1 Título

Refrigerador com placa peltier.

2 Objetivo

O objetivo do projeto é criar um refrigerador cuja temperatura poderá ser selecionada pelo usuário, além de poder escolher um horário específico para o seu acionamento.

3 Justificativa

Entregar ao usuário um sistema capaz de refrigerar comidas e bebidas consumidas no seu dia a dia, podendo variar a temperatura e horário de acionamento da forma como achar melhor.

4 Descrição do projeto, escopo, abrangência

O projeto terá como elemento central um microcontrolador MSP430G2553, no qual será responsável por controlar diversos elementos do sistema. É por meio dele que a IHM irá operar, além de realizar os cálculos necessários e dar os comandos dos acionamentos.

Para fazer o resfriamento foi utilizada uma placa peltier, dispositivo que quando energizado esquenta de um lado e esfria do outro. Acoplado a esta placa estão dois dissipadores de calor e dois coolers (uma placa e um cooler para cada lado), utilizados para não danificar a placa por conta das temperaturas atingidas. A parte fria da placa se encontra do lado de dentro do refrigerador e a quente do lado de fora.

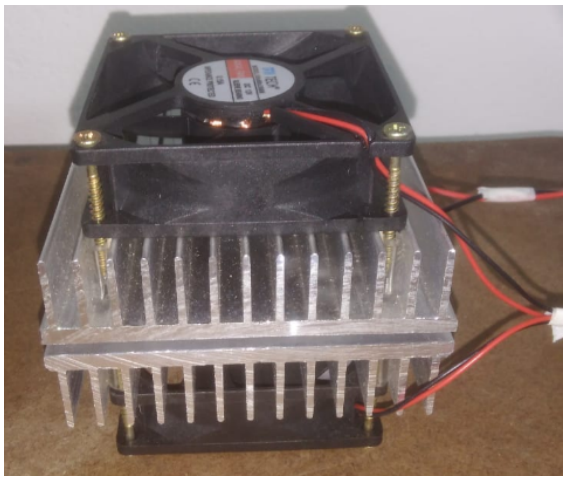


Figura 4.1: *sistema refrigerador.*

Para fazer os acionamentos são utilizados dois drivers idênticos, que consistem de um transistor com a base ligada ao MSP430 que irá acionar um relé cuja chave estará ligada à carga. Em um dos drivers a carga será a placa peltier e no outro serão os dois coolers ligados em paralelo.

O sistema irá operar em uma malha fechada, uma vez que existe a opção de selecionar a temperatura desejada. Para isso, um sensor de temperatura LM35, ligado a um condicionador de sinais de ganho 7,14 e um filtro passa baixas com frequência de corte igual a 1KHz foi ligado à entrada ADC do MSP430, que irá trabalhar com o dado de temperatura recebido.

A IHM é feita por meio de um teclado de 12 teclas e mais um display 16x2. O teclado irá ser usado para controlar as funções do refrigerador, enquanto o display irá dar as devidas informações ao usuário. Dentre as funcionalidades propostas estão:

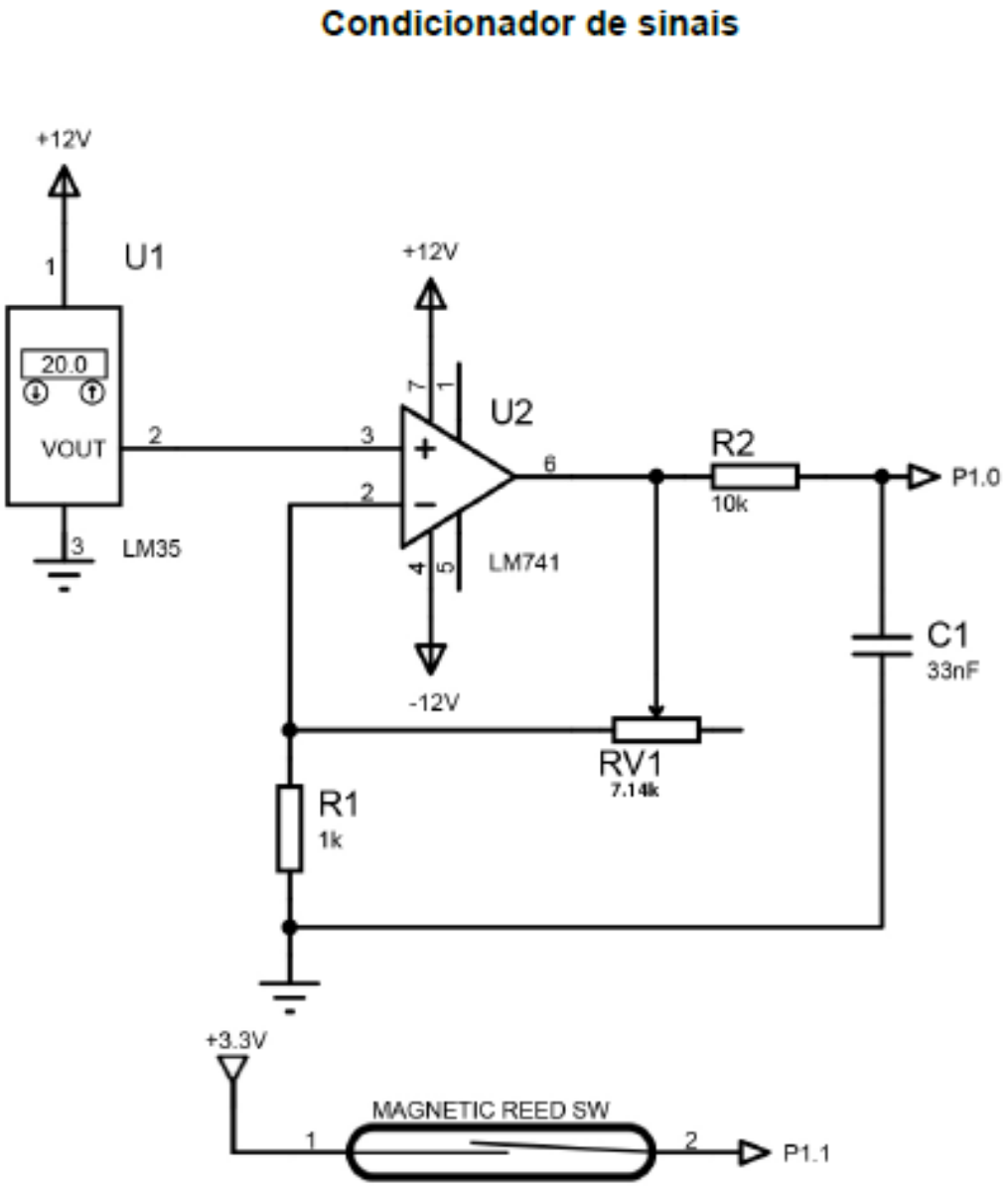
- Seleção da temperatura desejada.
- Modo ligado permanentemente.
- Mudança entre as escalas termométricas célsius e farenheight.
- Temporização para ligar.

Enquanto o sistema estiver em condição normal de operação, sem interferência do usuário, o display mostrará o tempo todo as informações: temperatura atual, temperatura desejada e se a placa peltier está ligada ou desligada.

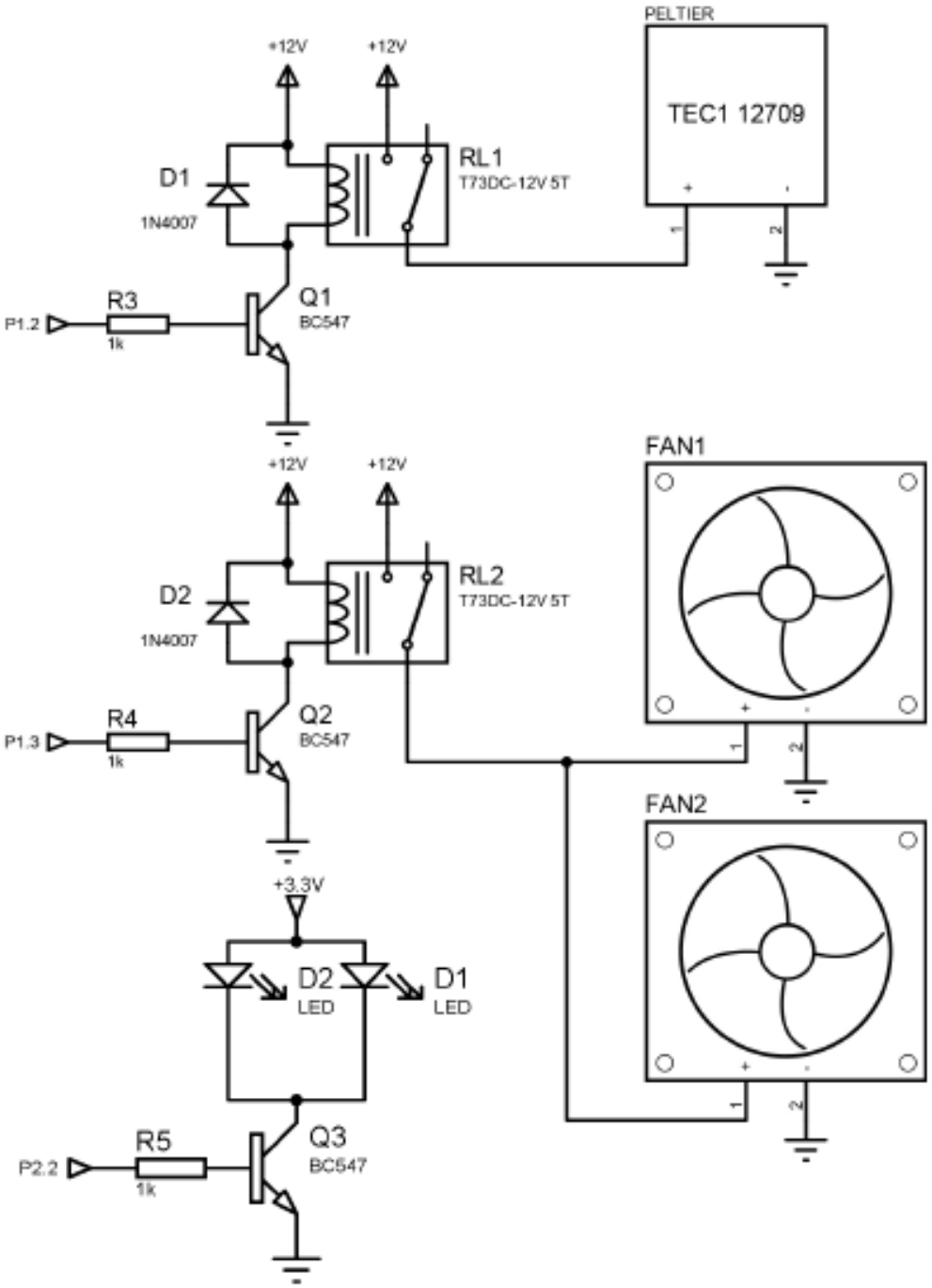
Por fim, o sistema é equipado com uma chave magnética ligada ao MSP430, componente que funciona como uma chave normalmente aberta que se fecha na presença de um ímã. Este componente é usado para identificar o estado da porta do refrigerador. Caso ela esteja fechada, o sistema opera normalmente, caso ela esteja aberta, a peltier para de funcionar (para economizar energia) e dois leds presentes no seu interior acendem.

Figura 4.2: *Chave magnética.*

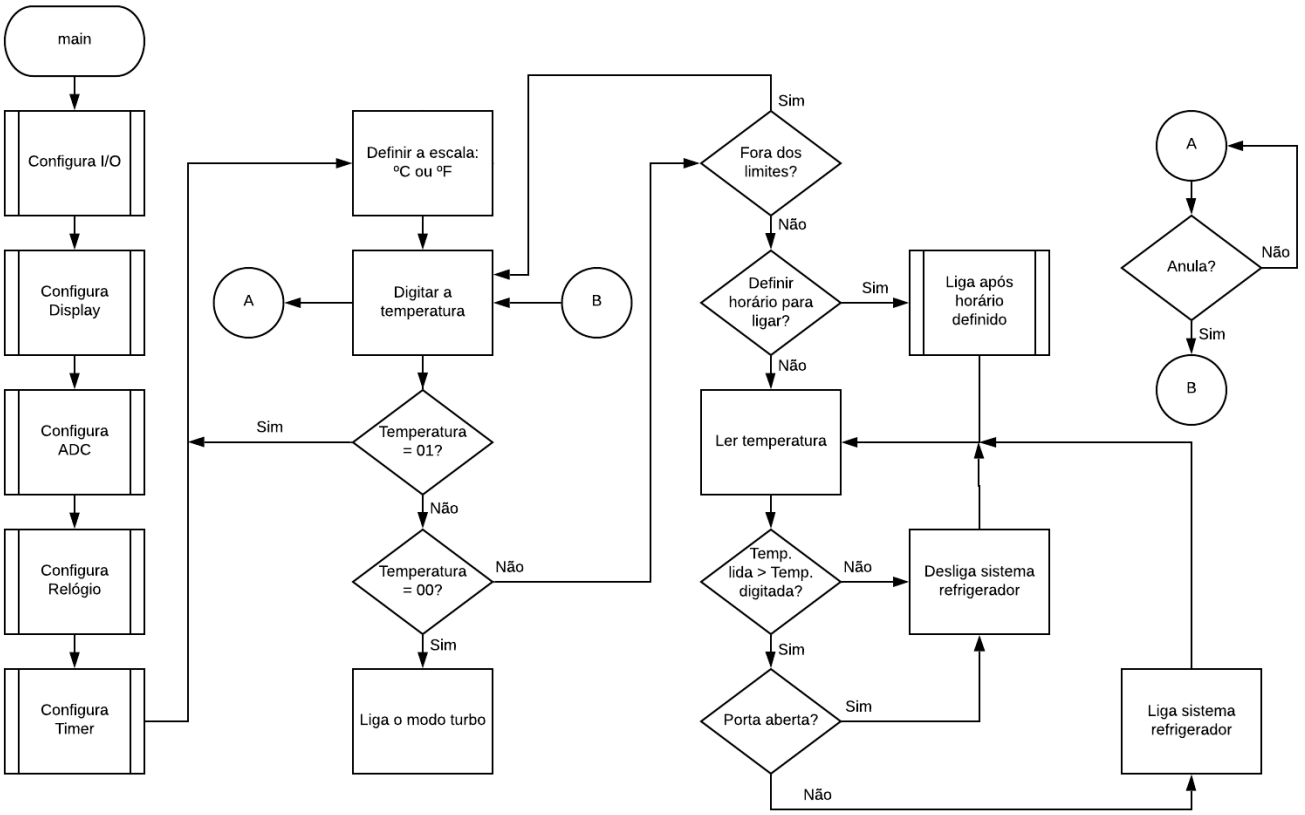
5 Diagramas de hardware



Drivers


PELTIER
TEC1 12709
+12V
+12V
D1
1N4007
RL1
T73DC-12V 5T
R3
1k
P1.2
Q1
BC547
FAN1
+12V
+12V
D2
1N4007
RL2
T73DC-12V 5T
R4
1k
P1.3
Q2
BC547
FAN2
+3.3V
D2
LED
D1
LED
R5
1k
P2.2
Q3
BC547

6 Descrição/especificações do software (fluxograma)



7 Código

```
// Biblioteca de defines

#define tmax_c 20

#define tmin_c 5

#define tmax_f (((tmax_c * 9)/5) + 32)

#define tmin_f (((tmin_c * 9)/5) + 32)

#define tolerancia 0.5

#define liga_pel       P1OUT &= ~BIT2;

#define desliga_pel    P1OUT |= BIT2;

#define liga_vent      P1OUT &= ~BIT3;

#define desliga_vent   P1OUT |= BIT3;

#define envia_temp_f    temp = ((2.5*tensao)/1023);temp = (temp/71.43e-3);temp = (((temp * 9)/5) + 32);sprintf (string, "TG = %2.1f", temp);envia_comando (0x80);envia_string (string);

#define envia_temp_c        temp = ((2.5*tensao)/1023);temp = (temp/71.43e-3);sprintf (string, "TG atual = %2.1f", temp);envia_comando (0x80);envia_string (string);

#define envia_valor_c   sprintf (estring, "TR = %2i", valor_c);envia_comando (0xC0);envia_string (estring);

#define envia_valor_f   sprintf (estring, "TR = %2i", valor_f);envia_comando (0xC0);envia_string (estring);

#define envia_erro     envia_comando (0x80);envia_string (erro);envia_comando (0xC0);envia_string ("              ");

#define envia_turbo    envia_comando (0xC0);envia_string (turbo);

#define envia_desliga   envia_comando (0x80);envia_string (gel);envia_comando (0xC0);envia_string (des);

#define envia_esc      envia_comando (0x80);envia_string (tempe);envia_comando (0xC0);envia_string ("              ");

#define envia_anu      envia_comando (0x80);envia_string (anu);envia_comando (0xC0);envia_string ("              ");

#define para_adc       ADC10CTL0 &= ~(ENC + ADC10SC);

#define inicia_adc     ADC10CTL0 |= ENC + ADC10SC;

#define envia_ini      envia_comando (0xC0);envia_string (ini);

#define envia_conf     envia_comando (0xC0);envia_string (conf);

#define envia_graus_c   envia_dado (0xdf);envia_dado ('C');

#define envia_graus_f   envia_dado (0xdf);envia_dado ('F');

#define envia_dez      envia_comando (0x8C);envia_dado (tecla);

#define envia_uni      envia_comando (0x8D);envia_dado (tecla);

#define sensor_mag     (P1IN & BIT1)

#define envia_lig      envia_comando (0xC9);envia_string ("|Ligado");

#define envia_par      envia_comando (0xC9);envia_string ("|Parado");

#define temp_em                envia_comando (0x80);envia_string ("Temperatura em ");envia_comando (0xC0);envia_graus_c;envia_string ("(1) ou ");envia_graus_f;envia_string ("(2)  ");
```

```
#define limpa_1        envia_comando (0x80);envia_string ("                ");

#define limpa_2        envia_comando (0xC0);envia_string ("                ");

#define envia_c_f      envia_comando (0x80);envia_string ("Caio e Felipe");envia_comando (0xC0);envia_string ("Resfriador");
__delay_cycles (_2s);

#define envia_deseja      envia_comando (0x80);envia_string ("Deseja escolher");envia_comando (0xC0);envia_string ("um
horario?");

#define envia_digite    envia_comando (0x80);envia_string ("   00:00:00        ");envia_comando (0xC0);envia_string ("DIGITE
O HORARIO  ");

#define celsius        (t == 0)

#define fah            (t == 1)

// Biblioteca de definições e funções para o kit G2553


#define Fclk   1.2e6          // Freq. de clk do DCO para o G2553

#define _5ms   5e-3 *Fclk

#define _10ms  10e-3*Fclk

#define _15ms  15e-3*Fclk

#define _20ms  20e-3*Fclk

#define _30ms  30e-3*Fclk

#define _2ms   (int)(2e-3*Fclk)

#define _2s    2*Fclk

#define _100us 100e-6*Fclk

#define _500ms 500e-3*Fclk

#define _50ms  50e-3*Fclk

#define _1s    1*Fclk

#define _1_5s  1.5*Fclk

//prototipo

void envia (unsigned char palavra);

int descobrir (unsigned char j);

void envia_dado(unsigned char palavra);

// Dando nomes mais interessantes para os bits das portas P1 e P2

// Porta P1 - conexões com o LCD

#define LCDPort_out  P1OUT                  // Os bits de dados do LCD estão conectados

#define LCDPort_dir  P1DIR                  // na porta P1

#define LCDPort_sel  P1SEL

#define LCD_D4       BIT4

#define LCD_D5       BIT5

#define LCD_D6       BIT6
```

```
#define LCD_D7      BIT7
#define LCDData     (LCD_D4+LCD_D5+LCD_D6+LCD_D7) // Barramento de 4 bits do display

// Porta P1 - conexõeos com o teclado
#define LedVm       BIT0                // Leds que já vêm montado no kit
#define LedVd       BIT1
#define ChS2        BIT3                // Micro chave que já vem montada no kit

#define TecLin_out   P1OUT              // As linhas do teclado estão conectadas na
#define TecLin_dir   P1DIR              // porta P1
#define TecLin_sel   P1SEL
#define TecL1       BIT4                // Cada uma das linhas do teclado
#define TecL2       BIT5
#define TecL3       BIT6
#define TecL4       BIT7
#define TecLins     (TecL1+TecL2+TecL3+TecL4)

// Porta P2
#define LCDCtl_out   P2OUT              // Os bit En e RS estão conecetados na
#define LCDCtl_dir   P2DIR              // porta P2
#define LCDCtl_sel   P2SEL
#define LCDCtl_ie    P2IE
#define LCDCtl_ifg   P2IFG
#define LCDEn       BIT1                // Enable  = Bit de controle do LCD
#define LCDRS       BIT7                // Este bit é compartilhado com a coluna 1
                                        // do teclado e o pino RS do LCD.
#define TecCol_in    P2IN               // As colunas do teclado estão conectadas
#define TecCol_dir   P2DIR              // na porta P2
#define TecCol_sel   P2SEL
#define TecCol_ie    P2IE
#define TecCol_ies   P2IES
#define TecCol_ifg   P2IFG
#define TecCol_ren   P2REN
#define TecCol_out   P2OUT
#define TecC1       BIT0                // Cada uma das colunas do teclado
```

```
#define TecC2       BIT5
#define TecC3       BIT6
#define TecCols     (TecC1+TecC2+TecC3)


// Primeira palavra de configuração do display LCD para o kit LaunchPad
#define LCD_DL      0                   // BIT4 = DL, 1 = barramento de 8 bits, 0 = 4 bits
#define LCD_N       BIT3                // BIT3 = N, 1 = 2 linhas, 0 = 1 linha
#define LCD_F       0                   // BIT2 = f, 1 = matriz 5x10, 0 = matriz 5x8


//-------------------------------------------------------------------------
//NOME: envia_comando
//FUNÇÃO: Faz RS = 0 e envia uma palavra para o display que será interpretada
//       como comando
//ENTRADA: palavra
//SAIDA: -
//-------------------------------------------------------------------------

void envia_comando(unsigned char palavra)
{
  LCDCtl_ie &= ~TecCols;                // Desabilita as INTs das colunas do teclado
  LCDCtl_dir |= LCDRS;                  // Redireciona bit como saída
  LCDCtl_out &= ~LCDRS;                 // Faz RS = 0 = comando
  envia(palavra);                       // Envia a palavara para o display
  LCDCtl_dir &= ~LCDRS;                 // Redireciona bit como entrada
  LCDCtl_ifg &= ~TecCols;               // Reseta os flags de INT que podem ter sido ativados
  LCDCtl_ie |= TecCols;                 // Rehabilita as INTs das colunas do teclado

  // Ao final da rotina é importante que o registrador de saída que atende ao
  // teclado tenha em nível zero os bits correspondentes às colunas C1 a C4 para
  // que assim seja efetivado o pull down.
  LCDCtl_out &= ~(TecC1+TecC2+TecC3);
}
```

```
//-------------------------------------------------------------------------
//NOME: envia_dado
//FUNÇÃO: Faz RS = 1 e envia uma palavra para o display que será interpretada
//       como caracteres ASCII a ser exibido
//ENTRADA: palavra
//SAIDA: -
//-------------------------------------------------------------------------

void envia_dado(unsigned char palavra)
{
  LCDCtl_ie &= ~TecCols;                    // Desabilita as INTs das colunas
  LCDCtl_dir |= LCDRS;                      // Redireciona bit como saída
  LCDCtl_out |= LCDRS;                      // Faz RS = 1 = Dado ou caracterer
  envia(palavra);                           // Envia a palavara para o display
  LCDCtl_dir &= ~LCDRS;                     // Redireciona bit como entrada
  LCDCtl_ifg &= ~TecCols;                   // Reseta os flags de INT que podem ter sido ativados
  LCDCtl_ie |= TecCols;                     // Rehabilita as INTs das colunas

  // Ao final da rotina é importante que o registrador de saída que tbm atende ao
  // teclado tenha em nível zero os bits correspondentes às colunas C1 a C4 para
  // que assim seja efetivado o pull down.
  LCDCtl_out &= ~(TecC1+TecC2+TecC3);
}



//-------------------------------------------------------------------------
// Função para o display LCD - Conexão de 4bits
// Nome: Envia
// Função: envia um byte para o display. Pode ser tanto comando quanto dado
//       Utiliza a conexão de 4 bits com o display, sendo:
//       D7 = P1.7
//       D6 = P1.6
//       D5 = P1.5
//       D4 = P1.4
//-------------------------------------------------------------------------
```

```
void envia (unsigned char palavra)
{
 LCDPort_out = (LCDPort_out & 0x0F) + (palavra & 0xF0); // Envia apenas o nibble superior
 LCDCtl_out &= ~LCDEn;          //Gera pulso de nível alto no pino Enable do display
 LCDCtl_out |= LCDEn;
 LCDCtl_out &= ~LCDEn;
 __delay_cycles(_100us);
 LCDPort_out = (LCDPort_out & 0x0F) + ((palavra<<4) & 0xF0); // Envia apenas o nibble inferior
 LCDCtl_out |= LCDEn;          //Gera pulso de nível alto no pino Enable do display
 LCDCtl_out &= ~LCDEn;


 // Restabelece os 4 bits das linhas do teclado com nível alto.
 LCDPort_out |= TecLins;
}
/*****************************************************************************
 Nome: EnviaString
 Função para enviar palavras para o display
*****************************************************************************/
void envia_string(char *string)
 {
   int i=0;
   while(*(string+i))
    {
    envia_dado(*(string+i));
    i++;
    }
 }

//---------------------------------------------------------------------------
// Função de configuração do I/Os relativos ao teclado
// Nome: Config_Tec
//---------------------------------------------------------------------------
```

```
void Config_Tec(void)
{
  // Configurar a linhas como saídas e as colunas como entrada
  // Ativar os resistores de pull-down nas C1 a C4.
  // A coluna C1 é compartilhada com o pino RS do display e já possui o seu
  // pull-down (um resistor externo de 4k7).
  TecLin_sel &= ~TecLins;            // Seleciona I/O
  TecLin_dir |= TecLins;             // As Linhas são saídas
  TecLin_out |= TecLins;             // Inicializa as linhas com 1

  TecCol_sel &= ~TecCols;            // Seleciona I/O
  TecCol_dir &= ~TecCols;            // As Colunas são entradas
  TecCol_ies &= ~TecCols;            // Seleciona INT na borda de subida
  TecCol_ifg &= ~TecCols;            // Reseta flags de INT
  TecCol_ie  |=  TecCols;            // Habilita INTs
  TecCol_ren |= (TecC1+TecC2+TecC3);            // Habilita pull up ou down
  TecCol_out &= ~(TecC1+TecC2+TecC3);           // Seleciona pull down
}
//-------------------------------------------------------------------------
// Função de configuração do ADC10
// Nome: Config_ADC
//-------------------------------------------------------------------------
void Config_ADC(void)
{
  P1SEL |= BIT0;
  ADC10CTL1 = INCH_0 + ADC10DIV_7 + ADC10SSEL_0 + CONSEQ_2;
  ADC10CTL0 = ADC10SHT_3 + ADC10ON + REF2_5V  + REFON + SREF_1 + ADC10IE + MSC;
  P1DIR |= (BIT2);
  desliga_pel;
  P1DIR |= (BIT3);
  desliga_vent;
  P1DIR &= ~BIT1;
  P1REN |= BIT1;
  P1OUT &= ~BIT1;
}
```

```
//---------------------------------------------------------------------------
// Função de configuração do I/Os relativos ao display
// Nome: Config_LCD
//---------------------------------------------------------------------------

void Config_LCD(void)
{
 // As linhas do teclado são compartilhadas com o barramento de dados de 4
 // bits do display, mesmo assim vamos configura-las aqui nesta rotina tbm.
 // Assim a rotina fica independente a rotina de configuração de tecaldo,
 // caso as conexões sejam alterada e deixem de ser compartilhadas.

 // Configura os 4 bits do barramento de dados como I/O
 LCDCtl_sel &= ~(LCDEn + LCDRS);      //Seleciona I/O

 // Configura pinos Enable e RS como saída e inicializa com zero
 LCDCtl_dir |=   LCDEn + LCDRS;
 LCDCtl_out &= ~(LCDEn + LCDRS);

 // Configura os bit do barramento de dados como saída
 LCDPort_sel &= ~LCDData;            // Seleciona I/O
 LCDPort_dir |=  LCDData;            // Configura como saída

}
//---------------------------------------------------------------------------
//NOME: envia_meio_comando
//FUNÇÃO: Faz RS = 0 e envia o nibble superir para o display que será interpretado
//       como comando
//       Esta função é utilizada na inicialzação do display com a interface de
//       4 bits. Neste caso o nibble 0010 deve ser enviado uma vez sozinho e
//       depois combinado com o nibble inferior. Este último envio é feito
//       com a função envia_comando já existente.
//ENTRADA: palavra
//SAIDA: -
//---------------------------------------------------------------------------
```

```
void envia_meio_comando(unsigned char palavra)
{
  LCDCtl_ie &= ~TecCols;                    // Desabilita as INTs das colunas do teclado
  LCDCtl_out &= ~LCDRS;                     // Faz RS = 0 = comando

  // Envia apenas o nibble superior
  LCDPort_out = (LCDPort_out & 0x0F) + (palavra & 0xF0);
  LCDCtl_out &= ~LCDEn;          //Gera pulso de nível alto no pino Enable do display
  LCDCtl_out |= LCDEn;
  LCDCtl_out &= ~LCDEn;
  __delay_cycles(_100us);
  LCDCtl_ifg &= ~TecCols;                   // Reseta os flags de INT que podem ter sido ativados
  LCDCtl_ie |= TecCols;                     // Rehabilita as INTs das colunas do teclado

  // Restabelece os 4 bits das linhas do teclado com nível alto.
  LCDPort_out |= TecLins;

}
//-------------------------------------------------------------------------------
//NOME: Init_LCD
//FUNÇÃO: inicializa o display
//-------------------------------------------------------------------------------
void Init_LCD()
{
  __delay_cycles(_50ms);
  envia_meio_comando(0x30);     // Envia apenas o nibble superior 0011 por 3 vezes seguidas
  __delay_cycles(_5ms);
  envia_meio_comando(0x30);
  __delay_cycles(_100us);
  envia_meio_comando(0x30);
  envia_meio_comando(0x20);     // Envia apenas o nibble superior 0010
  envia_comando(0x28);          // 4 bits de dados- duas linhas- matriz 5x8
  __delay_cycles(_5ms);
  envia_comando(0x0C);          //display e cursor ativos sem piscar
```

```
 envia_comando(0x06);         // deslocamento cursor para direita
 envia_comando(0x01);         // limpa display e retorna o cursor para 1 posição
 __delay_cycles(_2ms);
}
/*****************************************************************************
 Nome: teclado
 Descobre linha do teclado pressionada
*****************************************************************************/
#pragma vector=PORT2_VECTOR
__interrupt void teclado ()
{
 const char TAB_TEC [] = {"\00""123456789*0#"};
   unsigned char j = 0;
   __delay_cycles (_15ms);
   if (TecCol_in & (TecC1+TecC2+TecC3))              // Alguma linha pressionada
  {
      TecLin_out |= TecL1;
      TecLin_out &= ~(TecL2+TecL3+TecL4);
    if (TecCol_in & (TecC1+TecC2+TecC3))             // Primeira linha pressionada
      {
       j = descobrir(j);
              }
      TecLin_out |= TecL2;
      TecLin_out &= ~(TecL1+TecL3+TecL4);
    if (TecCol_in & (TecC1+TecC2+TecC3))             // Segunda linha pressionada
      {
       j += 0x03;
       j = descobrir(j);
      }
      TecLin_out |= TecL3;
      TecLin_out &= ~(TecL1+TecL2+TecL4);
    if (TecCol_in & (TecC1+TecC2+TecC3))             // Terceira linha pressionada
      {
       j += 0x06;
       j = descobrir(j);
```

```
      }
      TecLin_out |= TecL4;
      TecLin_out &= ~(TecL1+TecL2+TecL3);
    if (TecCol_in & (TecC1+TecC2+TecC3))              // Quarta linha pressionada
    {
      j += 0x09;
      j = descobrir(j);
    }
    tecla = TAB_TEC [j];
    j = 0;
    TecLin_out  |= (TecL1+TecL2+TecL3+TecL4);
  }
   TecCol_ifg  =  TecCol_ifg & ~(TecC1+TecC2+TecC3);       // Zera os flags de interrupção
}
/*************************************************************************
 Nome: descobrir
 Descobre a coluna do teclado pressionada
*************************************************************************/

 int descobrir (unsigned char j)
 {
   unsigned char coluna;
   coluna = TecCol_in & (TecC1+TecC2+TecC3);
   switch (coluna)
   {
    case TecC1:                          // Primeira coluna pressionada
      j +=1;
      break;
    case TecC2:                          // Segunda coluna pressionada
      j +=2;
      break;
    case TecC3:                          // Terceira coluna pressionada
      j +=3;
      break;
   }
```

```
    return j;
 }

//---------------------------------------------------------------------------
//NOME: atualiza
//FUNÇÃO: Função que atualiza o display com o horario atual
//---------------------------------------------------------------------------
void atualiza (void)
{
  char string [4];
  envia_comando (0x83);
  sprintf (string, "%2d:", horas);
  if (string [0] == 0x20)string [0] = 0x30;
  envia_string (string);
  sprintf (string, "%2d:", minutos);
  if (string [0] == 0x20)string [0] = 0x30;
  envia_string (string);
  sprintf (string, "%2d", segundos);
  if (string [0] == 0x20)string [0] = 0x30;
  envia_string (string);
}

//---------------------------------------------------------------------------
//NOME: config_timer
//FUNÇÃO: Configuração do timer0
//---------------------------------------------------------------------------
void config_timer()
{
  BCSCTL2  = DIVS_2;                    //Divide o clock de 1MHz do DCO por 4
  TA0CCTL0 = CCIE;                      // Ativa a interrupção de captura/compara
   TA0CTL    = TASSEL_2 + ID_3 + MC_1 + TACLR;    // Seleciona o SMCLK como fonte, divide o clock por 8, seleciona o
modo crescente e limpa o TA
  TA0CCR0  = 34375;                     // Configura como 1 segundo a base de tempo
}
```

```
//-------------------------------------------------------------------------
//NOME: relogio
//FUNÇÃO: Função de interrupção do timer, que atualiza o relogio
//-------------------------------------------------------------------------
#pragma vector = TIMER0_A0_VECTOR
__interrupt void relogio()
{
 // Parte para o teclado não ficar com delay
 TA0CCTL0 &= ~CCIFG;
 TA0CCTL0 &= ~CCIE;
 __enable_interrupt ();

 if(segundos < 59)
 {segundos ++;}
 else
 {
  segundos = 0;
  if(minutos <59)
  {minutos ++;}
  else
  {
   minutos = 0;
   if(horas <23)
   {horas ++;}
   else
   {
    segundos = 0;
    minutos = 0;
    horas = 0;
   }
  }
 }
```

```
  if (h == 1)atualiza ();                  // So irá atualizar o display se h for igual a 1
  TA0CCTL0 &= ~CCIFG;
  TA0CCTL0 = CCIE;
}

//-------------------------------------------------------------------------
//NOME: configura_relogio
//FUNÇÃO: Função de configuração inicial do relogio
//-------------------------------------------------------------------------
void configura_relogio (void)
{
  char i = 0;                          // Variavel que define qual posição do caracter esta sendo escrita
  envia_comando (0x80);
  envia_string ("   00:00:00          ");
  envia_comando (0xC0);
  envia_string ("DIGITE O HORARIO  ");
  tecla = 0;
  while (i <= 5)
  {
    if (tecla)
    {
      switch (i)
      {
      case 0:                          // Dezena das horas
        envia_comando (0x83);
        envia_dado (tecla);
        horas = 10 * (tecla - 0x30);
        break;
      case 1:                          // Unidade das horas
        envia_comando (0x84);
        envia_dado (tecla);
        horas += (tecla - 0x30);
        break;
```

```
   case 2:                          // Dezena das horas
    envia_comando (0x86);
    envia_dado (tecla);
    minutos = 10 * (tecla - 0x30);
    break;
   case 3:                          // Unidade das horas
    envia_comando (0x87);
    envia_dado (tecla);
    minutos += (tecla - 0x30);
    break;
   case 4:                          // Dezena das horas
    envia_comando (0x89);
    envia_dado (tecla);
    segundos = 10 * (tecla - 0x30);
    break;
   case 5:                          // Unidade das horas
    envia_comando (0x8A);
    envia_dado (tecla);
    segundos += (tecla - 0x30);
    break;
   }
   i++;                             // Proxima posição
   tecla = 0;
  }
}
if ((horas > 23) || (minutos > 59) || (segundos > 59))        // Caso horario digitado seja maior que o permitido
{
  envia_comando (0x80);
  envia_string ("     ERRO     ");
  __delay_cycles (_1s);
  configura_relogio ();
}
envia_comando (0xC0);
```

```
 envia_string ("CONFIRMA?      ");
 while (tecla != '#')
 {
  if (tecla == '*')configura_relogio ();                  // Anula a ação e volta ao inicio da função
 }
 tecla = 0;
}
//-------------------------------------------------------------------------------
//NOME: sis_ligado_horario
//FUNÇÃO: Função que define o horario de ligar a geladeira, caso o usuario queira
//-------------------------------------------------------------------------------
void sis_ligado_horario (void)
{
 char horas_d;                          // Hora desejada
 char minutos_d;                        // Minuto desejado
 char segundos_d;                       // Segundo desejado
 char i = 0;                            // Variavel que define qual posição do caracter esta sendo escrita
 envia_digite;
 tecla = 0;
 while (i <= 5)
 {
  if (tecla)
  {
   switch (i)
   {
   case 0:                              // Dezena das horas
    envia_comando (0x83);
    envia_dado (tecla);
    horas_d = 10 * (tecla - 0x30);
    break;
   case 1:                              // Unidade das horas
    envia_comando (0x84);
    envia_dado (tecla);
    horas_d += (tecla - 0x30);
    break;
```

```
   case 2:                          // Dezena das horas
    envia_comando (0x86);
    envia_dado (tecla);
    minutos_d = 10 * (tecla - 0x30);
    break;
   case 3:                          // Unidade das horas
    envia_comando (0x87);
    envia_dado (tecla);
    minutos_d += (tecla - 0x30);
    break;
   case 4:                          // Dezena das horas
    envia_comando (0x89);
    envia_dado (tecla);
    segundos_d = 10 * (tecla - 0x30);
    break;
   case 5:                          // Unidade das horas
    envia_comando (0x8A);
    envia_dado (tecla);
    segundos_d += (tecla - 0x30);
    break;
   }
   i++;                             // Proxima posição
   tecla = 0;
  }
}
if ((horas_d > 23) || (minutos_d > 59) || (segundos_d > 59))        // Caso horario digitado seja maior que o permitido
{
  envia_comando (0x80);
  envia_string ("     ERRO     ");
  __delay_cycles (_1s);
  sis_ligado_horario ();
}
envia_comando (0xC0);
envia_string ("CONFIRMA?      ");
while (tecla != '#')
```

```
 {
  if (tecla == '*')sis_ligado_horario ();          // Anula a ação e volta ao inicio da função
 }
 tecla = 0;
 limpa_2;                                  // Limpa a segunda linha
 envia_comando (0xC0);
 envia_string ("ESPERANDO A HORA  ");
 h = 1;
  while (((horas != horas_d) || (minutos != minutos_d) || (segundos != segundos_d)) && (tecla != '*'))  // Enquanto não der o
horario fica nesse loop
 {}
 tecla = 0;
}
//-------------------------------------------------------------------------------
//NOME: sis_ligado
//FUNÇÃO: Função que ativa o sistema de resfriamento e medição
//-------------------------------------------------------------------------------
void sis_ligado (float temp, char valor, char t)
{
 char string [14];
 char estring [8];
 char valor_c, valor_f;
 char i = 0;                               // Variavel que diz se a geladeira esta ligada (0) ou não (1)
 if (celsius)                              // Caso unidade seja celsius
 {
  valor_c = valor;
  valor_f = (((valor_c * 9)/5) + 32);
 }
 if (fah)                                  // Caso unidade seja fahrenheit
 {
  valor_f = valor;
  valor_c = (((valor_f - 32) * 5)/9);
 }
 while (tecla != '*')                      // Enquanto * não for pressionada a geladeira continuará ligada
 {
  if (y==1500)
```

```
{
  envia_comando (0xCE);
  if (celsius)
  {
    envia_valor_c
    envia_graus_c;
    temp = ((2.5*tensao)/1023);              // Parte que converte um valor binario em tensão
    temp = (temp/71.43e-3);                  // Converte tensão em temperatura
    envia_temp_c;
    envia_graus_c;
    envia_string ("  ");
  }
  if (fah)
  {
    envia_valor_f
    envia_graus_f;
    temp = ((2.5*tensao)/1023);              // Parte que converte um valor binario em tensão
    temp = (temp/71.43e-3);                  // Converte tensão em temperatura
    temp = (((temp * 9)/5) + 32);            // Converte celsius em fahrenheit
    envia_temp_f;
    envia_graus_f;
  }
  if (sensor_mag && (i == 0))                // Ira ligar a geladeira somente se a porta estiver fechada e tecla = #
  {
    envia_lig;
    if (temp >= valor) liga_pel;             // Caso a temperatura for maior que o valor a peltier é ligada
       if (temp <= (valor - tolerancia)) desliga_pel;  // Caso a temperatura for menor que o valor - a tolerancia a peltier é
desligada
    liga_vent;
  }
  if (sensor_mag==0)                         // Caso a porta for aberta, as ventoinhas e a peltier são desligadas
  {
    desliga_pel;
    desliga_vent;
    envia_par;
  }
```

```
      if ((i == 1) && (sensor_mag != 0))                // Caso a porta esteja fechada e a tecla * for pressionada somente a peltier é
desligada
    {
      desliga_pel;
      envia_par;
      liga_vent;
    }
   }
   if (tecla == '1')                          // A tecla 1 troca as unidades de temperatura
   {
    t ^= BIT0;
    tecla = 0;
   }
   if (tecla == '#')i = 0;
   if (tecla == '0')i = 1;
 }
}
//-------------------------------------------------------------------------------
//NOME: adc
//FUNÇÃO: Interrupção ADC
//-------------------------------------------------------------------------------
#pragma vector = ADC10_VECTOR
__interrupt void adc ()
{
 if (y==1501)
 {y=0;}
 y++;
 tensao = ADC10MEM;
}
#include "io430.h"
#include <stdlib.h>
#include <stdio.h>

// Variáveis globais.
unsigned char tecla=0;                          // Variavel que é atribuido o temp da tecla pressionada
int tensao = 0;                                 // Valor de tensão do ADC10
```

```
int y=0;                                    // Contador de interrupções
unsigned char segundos, minutos, horas = 0;          // Variaveis usadas no relogio
char h = 0;                                  // Variavel que diz se o horario deve ser mostrado (1) ou não (0)

// Includes de bibliotecas criadas
#include "defines.h"                         // Biblioteca de defines
#include "Lib_G2553.h"                         // Biblioteca de funções do kit G2553
#include "geladeira.h"                         // Biblioteca com as funções de funcionamento da geladeira
#include "relogio.h"                         // Biblioteca com as funções de funcionamento do relogio

//-------------------------------------------------------------------------
//NOME: main
//FUNÇÃO: Função principal do programa
//-------------------------------------------------------------------------
void main( void )
{
 WDTCTL = WDTPW + WDTHOLD;                   // Stop watchdog timer to prevent time out reset
 // Variaveis Usadas
 char anu [] = {"    Anulado     "};
 char conf  [] = {"Confirma?      "};
 char gel   [] = {"   Geladeira    "};
 char des   [] = {"   Desligada      "};
 char  ini  [] = {" DIGITE A TEMP:  "};
 char erro  [] = {"Temp nao aceita  "};
 char turbo [] = {"MODO TURBO      "};
 char tempe [] = {"Temperatura:  "};
 char string [15];
 char j = 0;                                 // Contem o estagio em que a varredura se encontra
 char t = 0;                                 // Variavel que contem se a temperatura será em celsius (0) ou em fahrenheit(1)
 char dez = 0;                                 // Variavel das dezenas do valor de temperatura
 char uni = 0;                                 // Variavel das unidades do valor de temperatura
 char valor = 0;                                 // Contem o valor de temperatura escolhido pelo usuario
 float temp;                                 // Variavel que contem a temperatura dentro da geladeira

 // Configurações
```

```
Config_LCD();                          // Configuração dos I/Os relativos ao display
Config_Tec();                          // Configuração dos I/Os relativos ao teclado
Init_LCD ();                           // Configuração do display
Config_ADC ();                         // Configuração do ADC
__enable_interrupt ();                 // Ativa as interrupções do msp
envia_c_f;                             // Envia inicialização pro display
configura_relogio ();                  // Função de configuração do horario atual
config_timer();                        // Configura o timer para o relogio, com base de tempo de 1s

while(1)
{
 temp_em;                              // Envia "temperatura em  celsius (1) ou em fahrenheit(2)"
 while (j == 0)                        // Primeiro estagio de varredura
 {
  if (tecla == '2')                    // Caso tecla = 2, a temperatura será em fahrenheit
  {
   t = 1;
   j++;                                // Proximo estagio
   tecla = 0;
  }
  if (tecla == '1')                    // Caso tecla = 1, a temperatura será em celsius
  {
   t = 0;
   j++;                                // Proximo estagio
   tecla = 0;
  }
 }

 h = 1;                                // O horario será mostrado no display
 limpa_1;                              // Limpa a linha 1
 envia_ini;                            // Envia "DIGITE A TEMP:"

 while (j==1)                          // Segundo estagio de varredura
 {
  if (tecla && tecla != '*' && tecla != '#')          // Caso tecla for pressionada e for diferente de # e *
```

```
 {
  h = 0;                                  // O horario não será mostrado no display
  dez = ((tecla-0x30)*10);                // Dezena da temperatura desejada
  envia_esc;                              // Envia "Temperatura:"
  __delay_cycles (_50ms);
  envia_dez;                              // Envia a dezena para o display
  tecla = 0;
  j++;                                    // Proximo estagio
 }
}

while (j==2)                              // Terceiro estagio de varredura
{
 if (tecla && tecla != '*' && tecla != '#')   // Caso tecla for pressionada e for diferente de # e *
 {
  uni = (tecla - 0x30);                   // Unidade da temperatura desejada
  envia_uni;                              // Envia a unidade para o display
  envia_comando (0x8E);
  if (t == 0)
  {envia_graus_c;}                        // Caso temperatura esteja em celsius, envia ºC
  if (t == 1)
  {envia_graus_f;}                        // Caso temperatura esteja em fahrenheit, envia ºF
  envia_conf;                             // Envia "Confirma?"
  j++;                                    // Proximo estagio
 }
 if (tecla == '*')                        // Caso pressione * a ação é anulada
 {
  envia_anu;
  __delay_cycles (_1s);
  envia_ini;
  j=1;                                    // Retorna ao segundo estagio
 }
}
while (j==3)                              // Quarto estagio
{
```

```
    valor = dez + uni;                          // Valor da temperatura escolhida

     if (valor == 01)                                  // caso valor seja igual a 01, é possivel escolher a unidade de temperatura
novamente

    {

     j = 0;                                 // Retorna ao primeiro estagio

     tecla = 0;

    }

    if (tecla == '#')                       // Caso tecla = #, a ação é confirmada

    {

     tecla = 0;

       if (celsius)                         // Caso temperatura seja maior que a escala, envia erro ao display e volta ao segundo
estagio (celsius)

     {

      if ((valor > tmax_c) || (valor < tmin_c) && valor !=0 && valor !=01)envia_erro;

     }

       if (fah)                             // Caso temperatura seja maior que a escala, envia erro ao display e volta ao segundo
estagio (fahrenheit)

     {

      if ((valor > tmax_f) || (valor < tmin_f) && (valor !=0) && (valor !=01))envia_erro;

     }

     if (valor == 00)                       // Caso temperatura escolhida seja igual a 00, o modo turbo é ativado

     {

      envia_turbo;

      liga_pel;

      liga_vent;

      inicia_adc;

      while (tecla != '*')                  // Cancela a ação caso * seja pressionada

      {

       if (celsius)                         // Envia temperatura em graus celsius

       {

        if (y == 1500)

        {

         envia_graus_c;

         envia_temp_c;

         envia_graus_c;

         envia_string ("          ");

        }
```

```
  }
  if (fah)                              // Envia temperatura em graus fahrenheit
  {
   if (y == 1500)
   {
    envia_graus_f;
    envia_temp_f;
    envia_graus_f;
    envia_string ("              ");
   }
  }
  if (tecla == '1')                     // A tecla 1 troca as unidade de temperatura
  {
   t ^= BIT0;
   tecla = 0;
  }
 }
 para_adc;
 envia_desliga;                         // Envia "desligado"
}

if (celsius)                            // Liga a geladeira caso esteja em celsius
{
 if ((valor <= tmax_c) && (valor >= tmin_c))        // Só entra na função caso o valor esteja dentro da escala
 {
  limpa_1;                              // limpa a primeira linha
  limpa_2;                              // limpa a segunda linha
  envia_deseja                          // Envia "Deseja escolher um horario?"
  while (tecla != '*' && tecla != '#'){}            // Confirma = #, Não desejo = *
  if (tecla == '#')sis_ligado_horario ();           // Caso aperte #, podera escolher o horario de ligar a geladeira
  limpa_1;                              // limpa a primeira linha
  limpa_2;                              // limpa a segunda linha
  tecla = 0;
  h = 0;                                // Horario não será mais mostrado
  inicia_adc
```

```
    sis_ligado (temp, valor, t);                    // Chama função de ligar a geladeira
    valor =0;
    desliga_pel;
    desliga_vent;
    para_adc;
    envia_desliga;                                  // Envia "Desliga"
   }
  }
  if (fah)                                          // Liga a geladeira caso esteja em fahrenheit
  {
   if ((valor <= tmax_f) && (valor >= tmin_f))      // Só entra na função caso o valor esteja dentro da escala
   {
    limpa_1;                                        // limpa a primeira linha
    limpa_2;                                        // limpa a segunda linha
    envia_deseja;                                   // Envia "Deseja escolher um horario?"
    while (tecla != '*' && tecla != '#'){}          // Confirma = #, Não desejo = *
    if (tecla == '#')sis_ligado_horario ();         // Caso aperte #, podera escolher o horario de ligar a geladeira
    limpa_1;                                        // limpa a primeira linha
    limpa_2;                                        // limpa a segunda linha
    tecla = 0;
    h = 0;                                          // Horario não será mais mostrado
    inicia_adc
    sis_ligado (temp, valor, t);                    // Chama função de ligar a geladeira
    valor = 0;
    desliga_pel;
    desliga_vent;
    para_adc;
    envia_desliga;                                  // Envia "Desliga"
   }
  }
  __delay_cycles (_1s);
  envia_ini;                                        // Envia "DIGITE A TEMP:"
  tecla = 0;
  j=1;                                              // Volta ao primeiro estagio de varredura
 }
```

```
    if (tecla == '*')                       // Caso aperte * o valor escolhido é anulado
    {
      envia_anu;
      __delay_cycles (_1s);
      envia_ini;                            // Envia "DIGITE A TEMP:"
      j=1;                                  // Volta ao primeiro estagio de varredura
    }
   }
   tecla = 0;
  }
}
```

8 Conclusões

Por fim, o projeto apresentou um funcionamento satisfatório e correspondeu às expectativas que se tinham dês de o início. Uma possível mudança para melhorar o funcionamento, seria substituir a peltier por mecanismos de resfriamento mais potentes, podendo obter temperaturas ainda menores.