PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

EletroTerm

CURITIBA

2009

Leandro Alves de Menezes

EletroTerm

Documentação referente ao Projeto Integrado do Curso de Graduação Engenharia de Computação da Pontifícia Universidade Católica do Paraná, orientado pelo professor Afonso Ferreira Miguel.

CURITIBA

2009

Agradecimentos

Aos amigos que contribuíram positivamente para o termino desse projeto.

Ao nosso professor que graças aos seus ensinamentos contribuíram para o termino desse projeto.

RESUMO

O projeto “EletroTerm” tem como finalidade aplicar os conceitos de microprocessadores e eletrônica, que se obteve com o decorrer do curso, para o desenvolvimento de um termômetro digital o qual ao obter a temperatura mostra no display.

Sumário

1. INTRODUÇÃO

Este trabalho tem como finalidade desenvolver um projeto baseado no MSP 430 que monitora o ambiente com a utilização de componentes eletrônicos dispostos no kit disponibilizado pela Universidade, cuja temperatura será observada em um display LCD em graus Celsius.

O software desenvolvido na linguagem C/C++ utilizando conceitos aprendidos em aulas como na experiência 09 e 10 que consiste em desenvolver funções para controlar o LCD em ambas as linguagens junto utilizando um exemplo fornecido pela *TI* (em anexo).

2. OBJETIVO

O projeto *EletroTerm* tem como objetivo monitorar a temperatura ambiente de acordo com o que o que for registrado pelo sensor. A temperatura será mostrada por display.

3. Desenvolvimento

O projeto necessita englobar os dois grandes grupos que são a parte de Hardware e Software. A parte de Hardware consiste no kit de desenvolvimento MSP 430 fornecido pela Universidade.

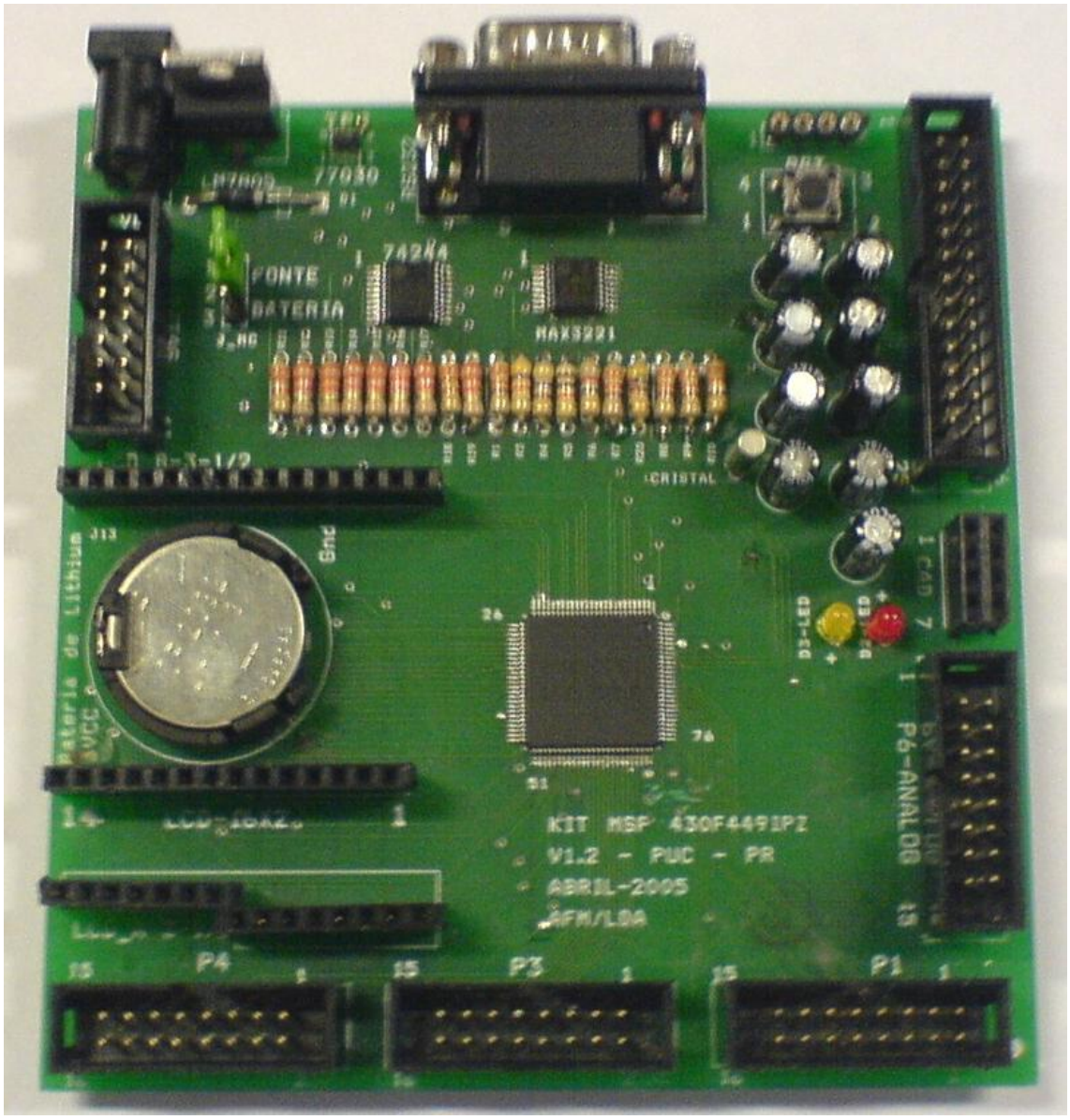

Imagem 01: TEXAS - MSP430 PUCPR Classroom Kit V1.2

Para a parte de interação com o usuário foi utilizado um *Display LCD 16x2*

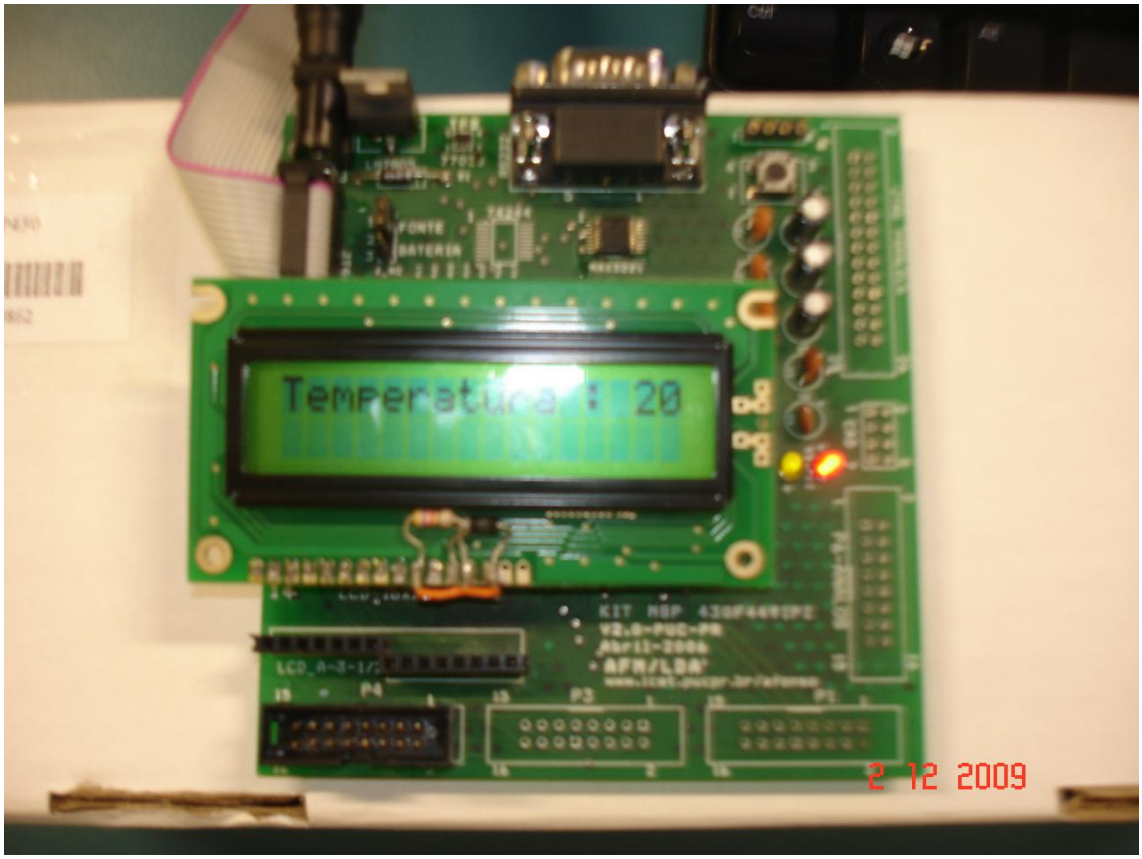

Imagem 02: Display LCD em funcionamento

Referente ao sensor, foi utilizado um sensor interno do próprio MSP430, assim facilitando sua implementação.

3.1 Software

Toda a parte lógica de programação foi desenvolvida utilizando a linguagem C/C++, utilizando o *IAR Embedded Workrech*.
O software que será transposto abaixo está todo comentado. Este, conta com várias subrotinas de acesso ao LCD, conversor analógico, leitura do sensor, conversão para Celsius.

```
#include  <msp430x44x.h>
//definição de ports
#define   Lcdrs_Out        P2OUT
#define   Lcdrs            BIT6
#define   Lcdhabilita_Out   P2OUT
#define   Lcdhabilita      BIT7
#define   Lcd_Dados        P5OUT

float conversao = 0, temp;
int Temperatura [6], Index = 0,Result = 0, Leitura = 0;
int temp_inteiro, Dezena, Unidade;
int BitIndica = 0;
void EscreveFrase (char *str, int tamstr);
void InitDisplay(void);
void Envia(void);
int Calcula (int Valor);

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P2DIR = Lcdrs+Lcdhabilita;
  P5DIR = 0xFF;                             // seta P5 como saída
  ADC12CTL1 = SHS_1 + SHP + CONSEQ_2;       // TA1, samp timer, rpt
  ADC12MCTL0 = SREF_1 + INCH_10;            // VRef+, A10
  ADC12IE = 0x01;                           // Enable ADC12MEM0 interrupt
  ADC12CTL0 = SHT0_7 + REFON + ADC12ON + ENC; // sample for 192
ADC12CLK cycles
  TACCTL1 = OUTMOD_4;                       // Toggle
  TACTL = TASSEL_2 + MC_2;                  // SMCLK, cont-mode
  _BIS_SR(LPM0_bits + GIE);                 // Enter LPM0, enable interrupts
  InitDisplay();

  while(1)
  {
    if (Leitura){
      //30mV = 26 Graus
      conversao = 0.0000080566*Result;
      temp = (conversao*26)/0.03;
      temp_inteiro = temp;
      EscreveFrase ("Temperatura : ", 14);
      Index = 0;
      Leitura = 0;
```

```
      Result = 0;
      _BIS_SR(LPM0_bits + GIE);
    }
  }
 }

// ADC12 Interrupt Service Routine - Media
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void)
{
  if (Index < 6)
  {
    Temperatura [Index] = ADC12MEM0;
    Index++;
  }
  else
  {
    for (int i=0; i<6; i++)
    {
        Result += Temperatura [i];
    }
    Result /= 6;
    Index = 0;
    Leitura = 1;
   _BIC_SR_IRQ(CPUOFF);                    // Exit LPM0
  }
}
void Envia() //Peteleco para sair o dado do display, o lcdhabilita e o negado é para o pulso, ou seja o enable
{
  if (!BitIndica)
    Lcdrs_Out &= ~Lcdrs;//0 = and
  else
    Lcdrs_Out |= Lcdrs;//1 = or
  //Enable
  Lcdhabilita_Out |= Lcdhabilita;
  for (int i = 0; i <500; i++);
  Lcdhabilita_Out &= ~Lcdhabilita;
}

void InitDisplay()
{
  BitIndica = 0;
//barramento tem 8  bits
   Lcd_Dados = 0x30;
        Envia ();
//barramento tem 8  bits
   Lcd_Dados = 0x30;
        Envia ();
//barramento tem 8  bits
```

```
   Lcd_Dados = 0x30;
       Envia ();
// Function set (8-bit interface, 2 linhas,matriz de 5*7)
   Lcd_Dados = 0x38;
       Envia ();
// Desliga o cursor
   Lcd_Dados = 0x0C;
       Envia ();
// Apaga display
   Lcd_Dados = 0x01;
       Envia ();
// Seleção de modo de entrada (Entry mode set )
   Lcd_Dados = 0x06;
       Envia ();
}


void EscreveFrase (char *str, int tamstr)
{
   BitIndica = 0;         //envio do comando
   Lcd_Dados = 0x80;      //posiciona cursor
   Envia ();

  for (int i=0; i<tamstr; i++)
  {
    Lcd_Dados = str[i];
    BitIndica = 1;     //envio do caracter
    Envia();
  }
 //Separação
   Unidade = Calcula (temp_inteiro);
   Lcd_Dados = Dezena+'0';
   Envia ();
   Lcd_Dados = Unidade+'0';
   Envia ();
}

int Calcula (int Valor)
{
  Dezena = (Valor/10);
  if (Valor >= 10)
  {
     while (Valor >= 10)
     {
        Valor  -= 10;
      }
  }
  return Valor;
```

4. Conclusão

Com o projeto conseguiu-se com sucesso utilizar a maioria dos conhecimentos adquiridos até o momento no curso de Engenharia de Computação.
O projeto serviu para demonstrar que tendo um plano de trabalho e seguindo-o é possível desenvolver o projeto tranquilamente.
Mesmo tendo obstáculos no decorrer do projeto, consegui terminar com êxito. Esse projeto mostrou que quando tem dificuldades é possível contar com os amigos e com eles estarei preparado para futuros desafios.

5. REFERÊNCIAS BIBLIOGRÁFICAS

http://www.afonsomiguel.com
http://focus.ti.com/mcu/docs/mcuprodcodeexamples.tsp?sectionId=96&tabId=1468
http://afonsomiguel.com/sites/default/files/msp430f449.pdf

6. Anexo

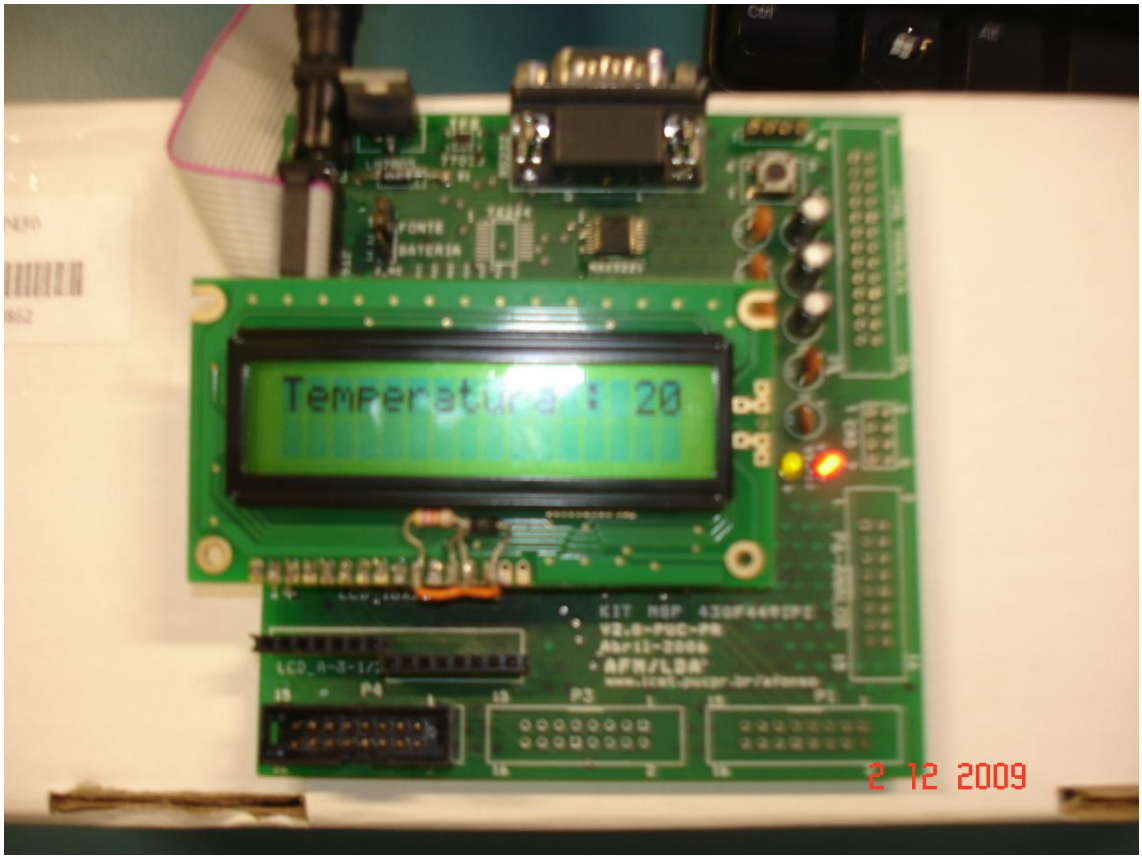

Imagem 03: Imagem do projeto em funcionamento

Código onde o código do projeto foi baseado.

```
//******************************************************************************
//  MSP-FET430P440 Demo - ADC12, Sample A10 Temp and Convert to oC, TA1 Trigger
//
//  Description: Use the ADC12's integrated temperature sensor to measure
//  temperature. Sample time is set for 125 ADC12CLK cycles to allow 30us for
//  the integrated temperature sensor (see datasheet). ADC12 is operated in
//  repeat-single-channel mode with the sample and convert trigger sourced
//  from Timer_A CCR1. Timer_A is configured for continuous mode and is clocked
//  by SMCLK. TA1 is set for toggle mode triggering the ADC12 every 125ms.
//  The ADC12MEM0_IFG bit set at the end of each conversion triggers an ISR.
//  Normal mode is LPM0.
//  ACLK = LFXT1 = 32768, MCLK = SMCLK = DCO = 32xACLK = 1048576Hz,
//  ADC12CLK = ADC12OSC
//  //* An external watch crystal between XIN & XOUT is required for ACLK *//
//
//  Uncalibrated temperature measured from device to devive will vary do to
//  slope and offset variance from device to device - please see datasheet.
//
//               MSP430F449
//            -----------------
```

```
//        /|\|              XIN|-
//         | |               | 32KHZ
//         --|RST          XOUT|-
//           |               |
//           |A10 (Temp)   P5.1|-->LED
//
//  M. Buccini
//  Texas Instruments Inc.
//  Feb 2005
//  Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.21A
//******************************************************************************
#include  <msp430x44x.h>

int long temp;
int long IntDegC;

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;               // Stop watchdog timer
  P5DIR = 0x02;                           // Set P5.2 as output
  ADC12CTL1 = SHS_1 + SHP + CONSEQ_2;       // TA1, samp timer, rpt
  ADC12MCTL0 = SREF_1 + INCH_10;           // VRef+, A10
  ADC12IE = 0x01;                         // Enable ADC12MEM0 interrupt
  ADC12CTL0 = SHT0_7 + REFON + ADC12ON + ENC; // sample for 192 ADC12CLK cycles
  TACCTL1 = OUTMOD_4;                     // Toggle
  TACTL = TASSEL_2 + MC_2;                // SMCLK, cont-mode

  while(1)
  {
    _BIS_SR(LPM0_bits + GIE);
    //  oC = ((x/4096)*1500mV)-986mV)*1/3.55mV = x*423/4096 - 278
    IntDegC = (temp - 2692) * 423;
    IntDegC = IntDegC / 4096;
    _NOP();                               // SET BREAKPOINT HERE
  }
 }

// ADC12 Interrupt Service Routine
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void)
{
  temp = ADC12MEM0;                       // IFG is cleared
  _BIC_SR_IRQ(CPUOFF);                    // Exit LPM0
  P5OUT ^= 0x02;                          // Toggle LED
}
```