Notas de Aula

Microcontroladores PIC - Compilador: MikroC

Autor: Prof. Michel Robert Veiga

Última Atualização 01/08/2018

Sumário

1.1 Introdução. 5
1.2 Apresentação – Kit de Desenvolvimento 5
2 Hardware 5
2.1 Circuito , é apresentado na Figura 1, o esquema elétrico. 7
2.2 Módulo LCD 16x2 8
2.3 Conector de expansão 9
2.4 Layout do Circuito impresso 9
2.5 Silk 10
2.6 Layout impressão. 10
2.7 Lista de Componentes 11
2.8 Código de teste no MikroC. 13
3 Sistema de Numeração 15
3.1 Sistema de numeração binária. 16
3.2 Conversão do sistema decimal para binário. 18
4 Entrada e Saída Digital i/o. 19
5 Criando Projeto no MiKroC e simulando no Proteus. 22
6 Criando um Hardware no Proteus. 29
7 Simulando o Software Criado no MiKroC no Proteus. 30
8 Portas Lógicas 34
9 Comandos relacionais 34
10 Comandos condicionais. 35
11 Display de 7 segmentos 36
11.1 Display de 7 segmentos utilizando o conversor 7447 ou 7448. 36
11.2 Display de 7 segmentos sem o conversor 38
11.3 Display de 7 segmentos multiplexado 38
12 Vetor – exemplo com display de 7 segmentos. 40
13 Quadro Resumo , entrada e saída digital. 42
14 Exercícios entrada e saída digital 43
15 Display de LCD 44
16 Conversor Analógico Digital 48
16.1 Exemplo Prático – Leitura de uma tensão de 0 a 5V. 49
17 Divisor de tensão. 51
18 Sensor de Temperatura 53
19 Exercícios Conversor Analógico Digital 57
20 Função 58

20.1 Função sem variável de entrada. 58
20.2 Função com variável de entrada. 59
21 PWM (Pulse Width Modulation) ou Modulação de Largura de Pulso. 60
21.1 Exemplo 01 – PWM por Software. 61
21.2 PWM por Hardware. 63
22 Bibliografia 66

Lista de Figuras.

Figura 1 - esquema elétrico 7
Figura 2 - Soma em sistemas de numeração. 15
Figura 3 - Sensores de Temperatura 53
Figura 4 - Gráfico da Temperatura X Tensão, Reta de tendência. 55
Figura 5 - Gráfico da leitura da temperatura com a aproximação de um polinômio. 56
Figura 6 - PWM 60
Figura 7 - Sinal PWM 63
Figura 8 - Pic - módulo PWM. 64

1.1 Introdução.

Esse material serve de apoio para as disciplinas de Microcontroladores, Sensores e atuadores do curso de eletrônica automotiva da Fatec Taubaté, e da disciplina LOM 3233 – Microprocessadores, do curso de Engenharia Física da USP Lorena. O objetivo do material é auxiliar no aprendizado da programação de microcontroladores PIC com notas de aula e exemplos dados em aula.

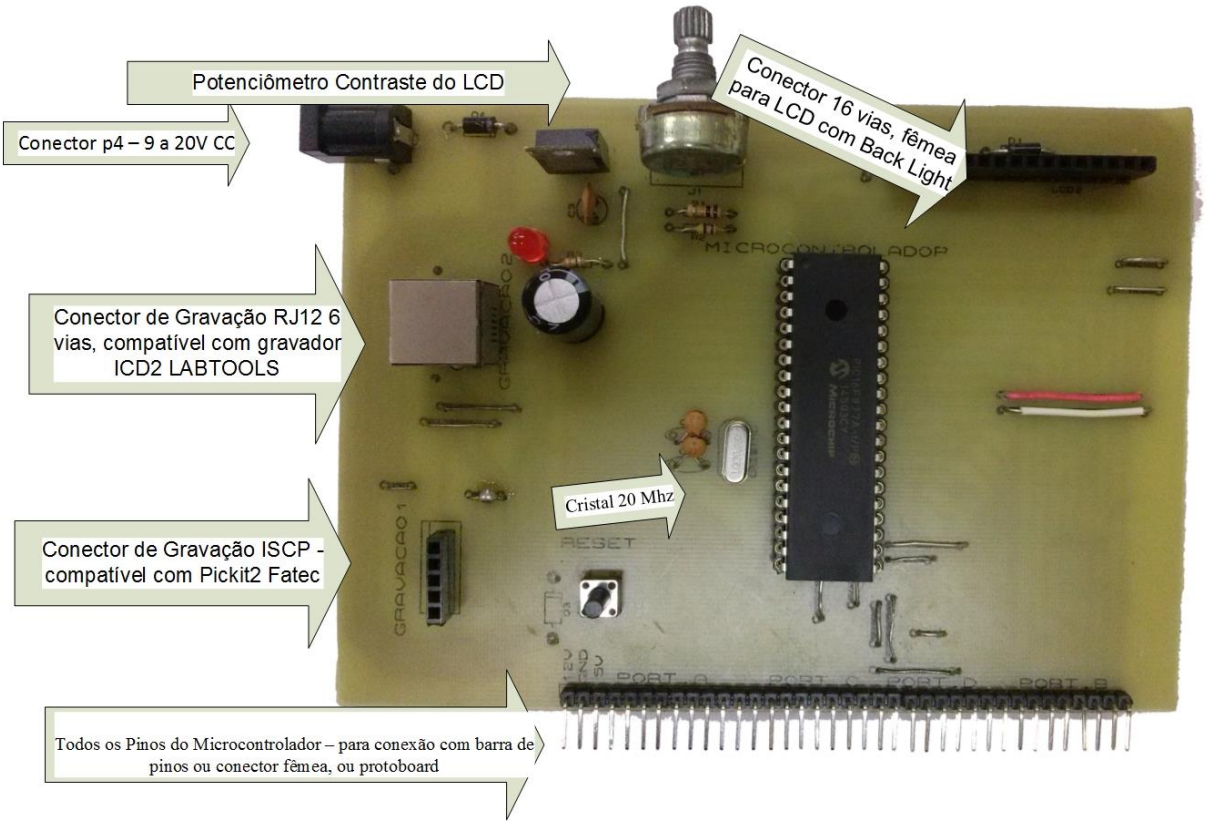
1.2 Apresentação – Kit de Desenvolvimento

A placa Fatec Lab foi desenvolvida para o estudo de microcontroladores da Microchip com 40 pinos, sendo que o microcontrolador recomendado para os primeiros experimentos é o PIC16F877A.

Os recursos disponíveis na placa são:

- LCD alfanumérico.
- Conector de expansão com todos os pinos do microcontrolador.
- Entrada de alimentação – Conector P4 - 9 a 20v.
- Potenciômetro de contrate do lcd
- Conector de 16 vias fêmea para LCD com Back Light
- Conector de Gravação ICSP 5 vias fêmea compatível com PICKIT2 Fatec

2 Hardware



2.1 Circuito , é apresentado na Figura 1, o esquema elétrico.

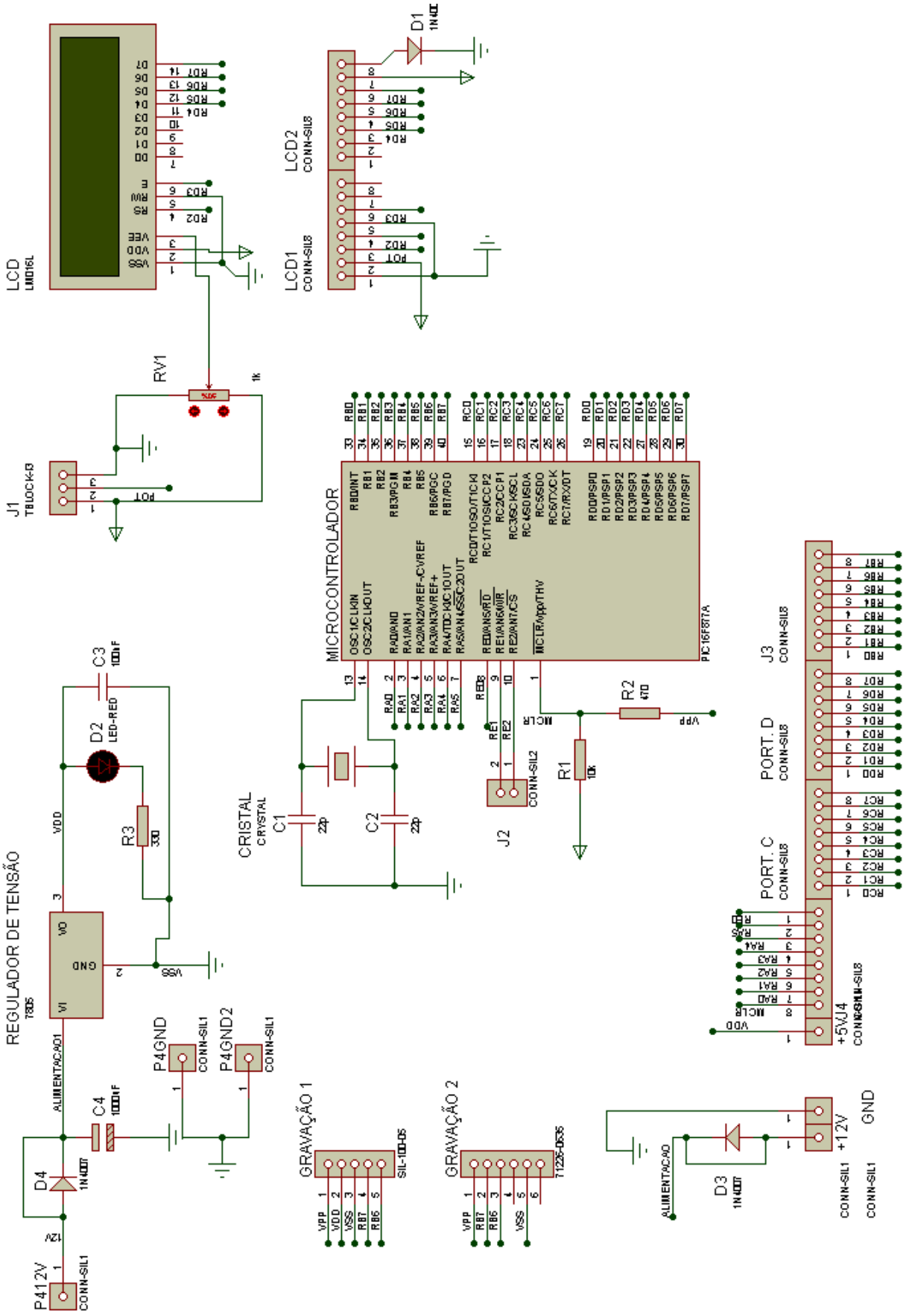
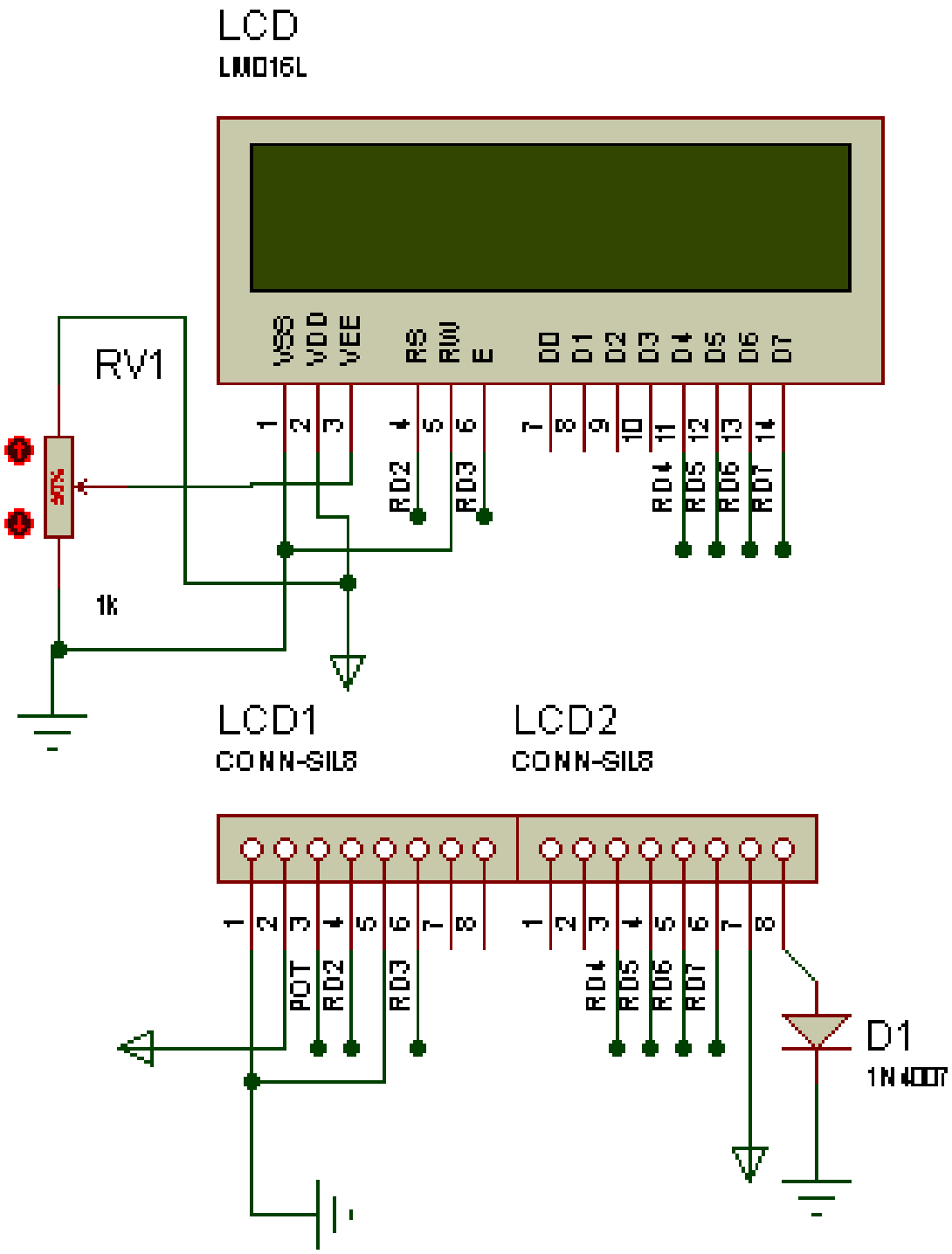
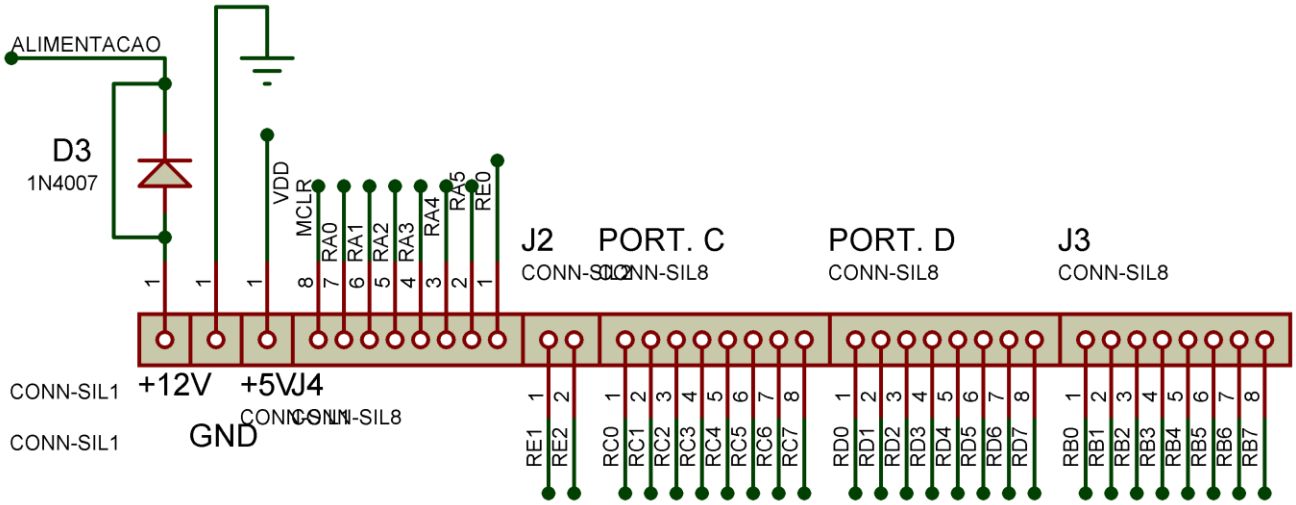


Figura 1 - esquema elétrico

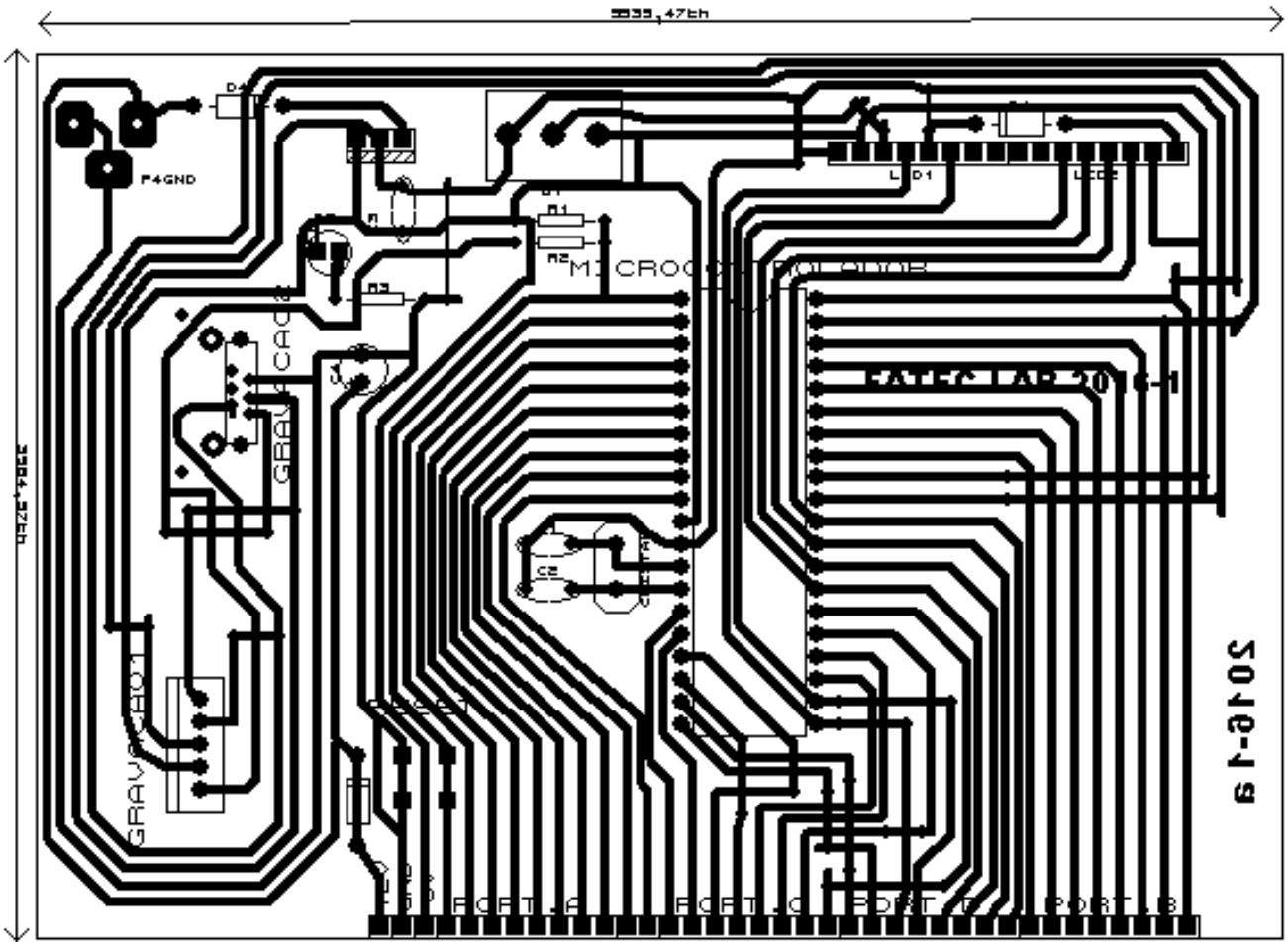
2.2 Módulo LCD 16x2



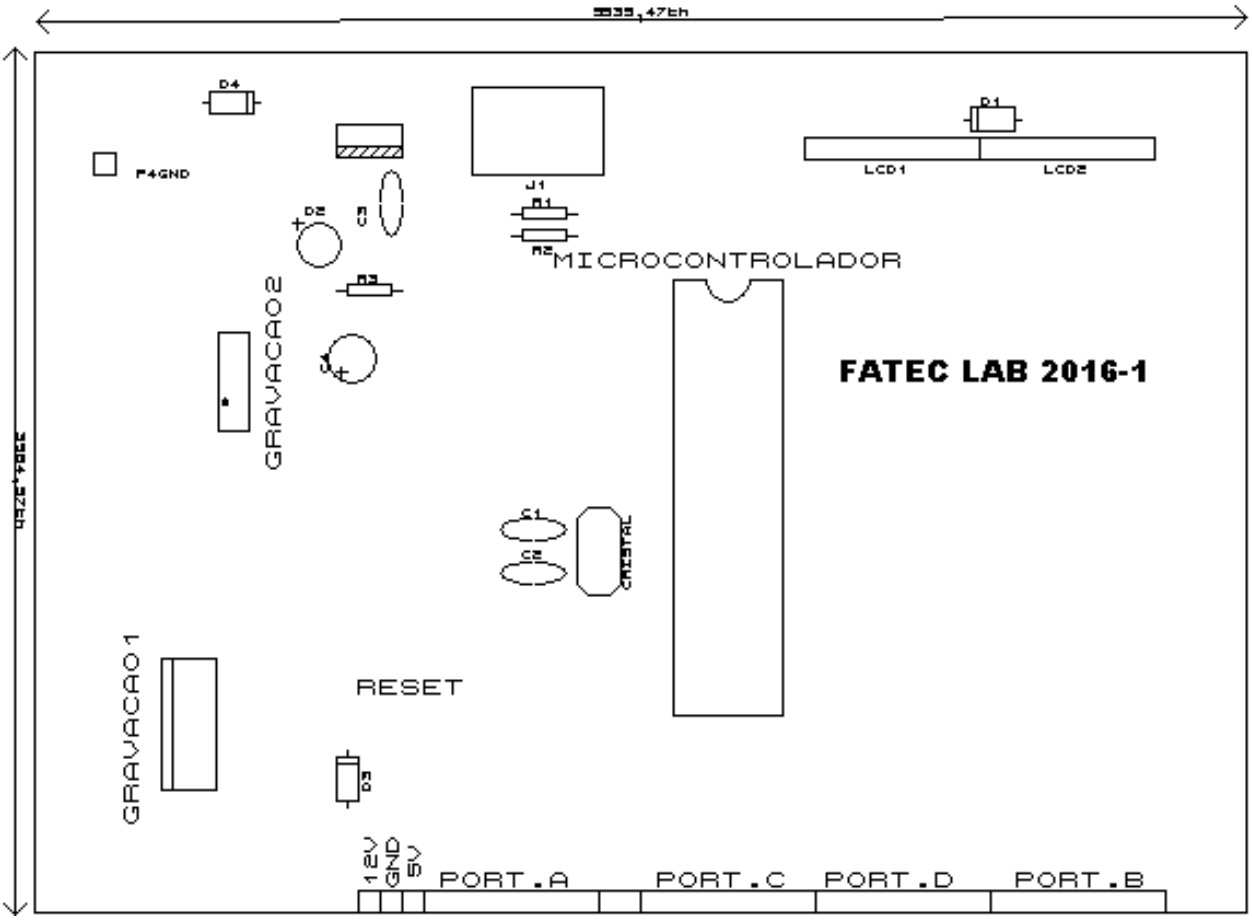
2.3 Conector de expansão



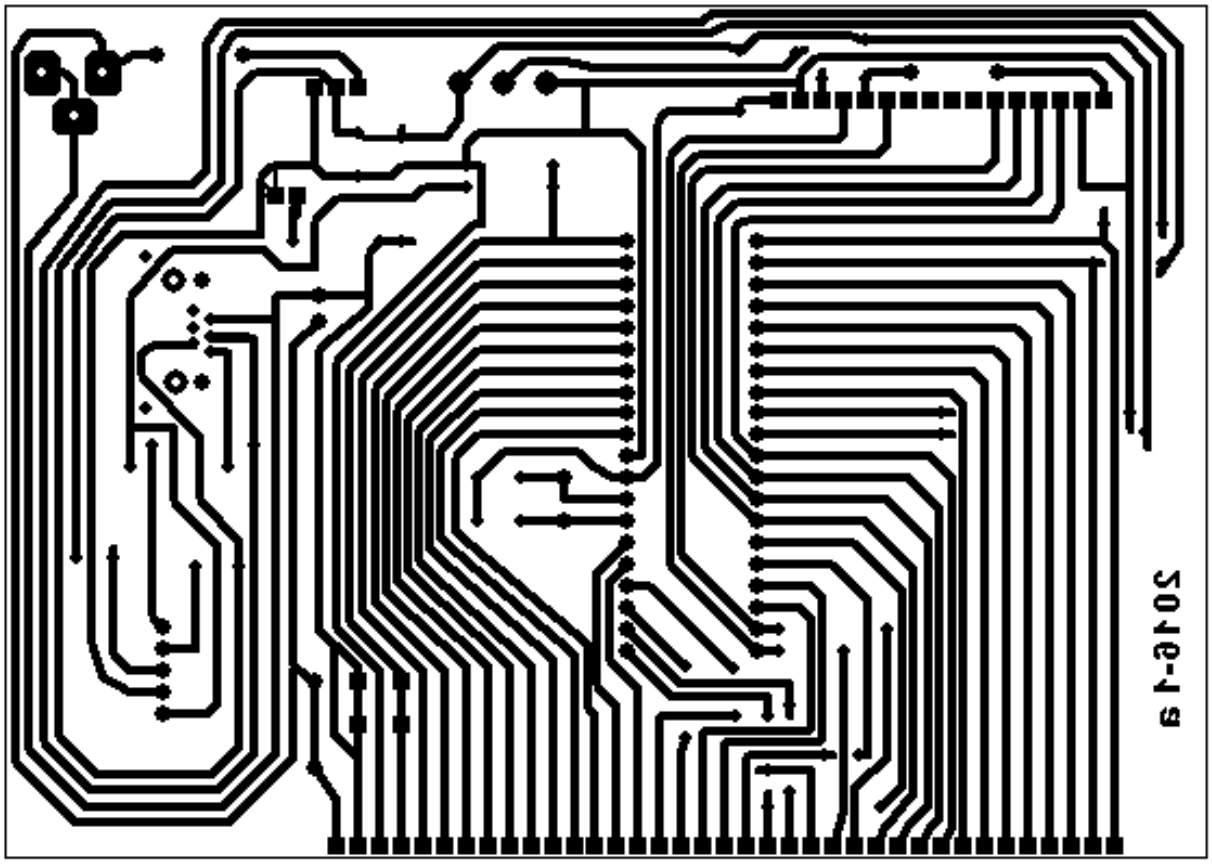
2.4 Layout do Circuito impresso



2.5 Silk



2.6 Layout impressão.



2.7 Lista de Componentes

LISTA DE COMPONENTES			
Quant	.Componente	Referêbcua	Valor
1	resistor 1/4 W	R1	10k
1	resistor 1/4 W	R2	470
1	resistor 1/4 W	R3	330
1	capacitor eletrolítico	C4	1000uF/25v
1	capacitor cerâmico	C3	100nF
2	capacitor cerâmico	C1, C2	22pF
2	conector		conector 40 pinos para o PIC16F877A – um para placa outro no pic.
1	Micro controlador		PIC16F877A
3	Diodo	D1,D4	1n4007
1	Led	D2	LED-RED
1	potenciômetro	J1	1K
1	display de lcd		lm016f
1	regulador de tensão		LM7805
1	cristal	cristal	20MHz
1	conector	ports	conector 90° macho 37 pinos (barra 40 pinos)
1	conector	gravação1	conector femea 5 pinos
1	conector	gravação2	RJ12 de 6 vias
1	conector		conector macho 16 pinos
1	botão	Reset	push botton
1	conector		femea 16 pinos
1	conector		RJ12 de 6 vias (OPCIONAL) PARA GRAVAÇÃO PELO GRAVADOR ICD2
1	conector		P4 - FONTE DE ALIMENTAÇÃO

1	placa de fenolite ou fibra face simples	15x10cm

2.8 Código de teste no MikroC.

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
//  Fim das conexões com LCD.

void main()
{
ADCON1  = 0b00000111;                  // Configure AN pins as digital I/O
Lcd_Init();                  // Initialize LCD
trisa=0;
trisb=0;
trisc=0;
trise=0;

Lcd_Cmd(_LCD_CLEAR);         //  Limpa o display
Lcd_Cmd(_LCD_CURSOR_OFF);    // Cursor off desliga o cursor
Lcd_Out(1,16," Teste Fatec ");    // escreve na linha 01 coluna 06
Lcd_Out(2,16,"          TESTE  FATEC  TAUBATE");    // escreve na linha 2 coluna 06
// Write text in first row
porta=0;
portb=0;
```

```
portc=0;
porte=0;


while(1)
    {
      Lcd_Cmd(_LCD_SHIFT_LEFT);    // desloca para esquerda
      delay_ms(200);
      porta=~porta;
      portb=~portb;
      portc=~portc;
      porte=~porte;
    }

}
```

3 Sistema de Numeração

Os sistemas de numeração foram criados quando o homem primitivo começou a ter a necessidade de contar, desde então, existem vários sistemas, desde o babilônico, romano indu e o sistema atual. Do ponto de vista computacional, o sistema decimal não é o sistema de numeração mais eficiente, para o computador, 10 símbolos que representam números não é o sistema mais simples, e sim um sistema com apenas dois símbolos, veja a questão da

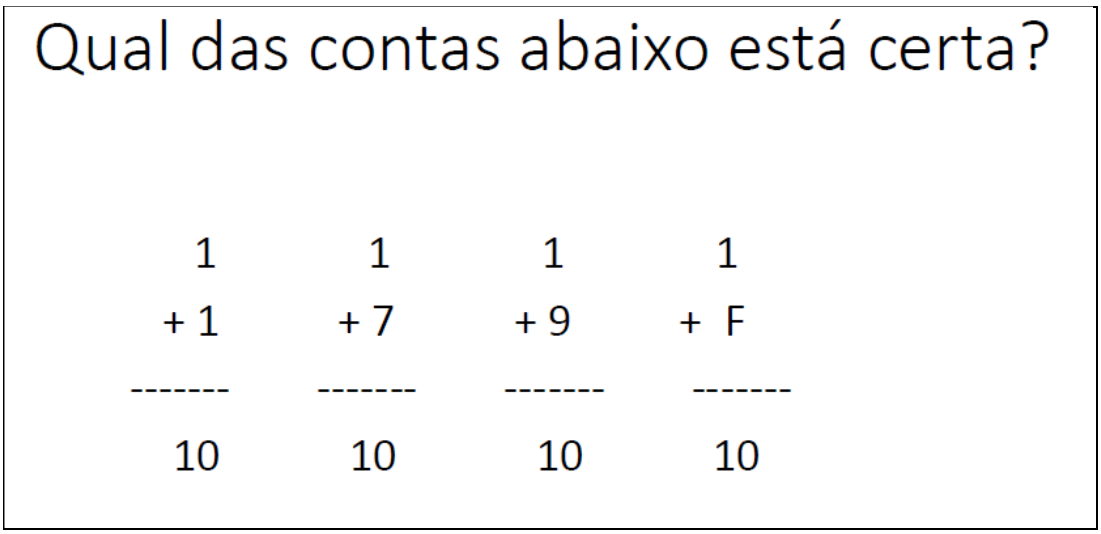


Figura 2 - Soma em sistemas de numeração.

No exemplo acima, acertou quem responde que todas estão corretas, na primeira temos o sistema binário, com seus dois símbolos, na segunda o octal, com oito símbolos, na terceira o nosso conhecido sistema decimal com 10 símbolos, (0 a 9) e na quarta o sistema hexadecimal, com 16 símbolos, do 0 ao F. Em uma soma ou sequência de números, quando chegamos ao símbolo final, iniciamos a repetição dos símbolos.

Na computação o sistema mais utilizado é o binário, e em alguns casos também o sistema hexadecimal. Felizmente na programação em linguagem C, todas as operações na programação podem ser feitas como o sistema decimal, apenas em algumas configurações temos a necessidade de utilizar o sistema binário.

- **Sistema Binário:** importante sistema de numeração, utilizado na tecnologia dos computadores. Sua base é “dois”, tendo somente dois algarismos: { 0, 1 };
- Toda a informação que circula dentro de um sistema informático é organizada em grupos de bits
- Os mais frequentes são os múltiplos de 8 bits: 8, 16, 32, etc.
- **Sistema Decimal:** sistema de números em que uma unidade de ordem vale dez vezes a unidade de ordem imediatamente anterior. Sua base numérica é de dez algarismos: { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }.
- **Sistema Octal:** Sistema de numeração cuja base é oito, adotado na tecnologia de computadores. Sua base numérica é de oito algarismos: { 0, 1, 2, 3, 4, 5, 6, 7 };
- **Sistema Hexadecimal:** Sistema de numeração cuja base é dezesseis. Esse sistema trabalha com dez algarismos numéricos baseados no decimal e com a utilização de mais seis letras. Os algarismos deste sistema são: { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }.

3.1 Sistema de numeração binária.

1 Byte → 8 bits → 256 combinações possíveis

No sistema binário (0 e 1), para determinar o número de combinações com n bits, basta calcular 2^n

Exemplos:

1 bit → 2^1=2 combinações possíveis (0 e 1)

2 bit → 2^2=4 combinações possíveis (0 0) (0 1) (1 0) (1 1)

3 bit → 2^3=8 combinações possíveis

0 0 0

0 0 1

0 1 0

0 1 1

1 0 0

1 0 1

1 1 0

1 1 1

4 bit $\rightarrow$ 2^4=16 combinações possíveis

0 0 0 0

0 0 0 1

0 0 1 0

0 0 1 1

0 1 0 0

0 1 0 1

0 1 1 0

. . . .

1 1 1 1

Sistema de numeração decimal

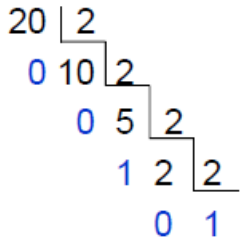
1998 = 1x1000 + 9x100 + 9x10 + 8x1

= $1x10^3 + 9x10^2 + 9x10^1 + 8x10^0$

Decimal	Binário
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0

3.2 Conversão do sistema decimal para binário.

Para converter um número decimal para um número binário é preciso fazer divisões sucessivas por 2 até se obter o quociente 1, agrupar o último quociente e todos os restos da divisão encontrados em ordem inversa.



$20_{(10)} = 10100_{(2)}$

4 Entrada e Saída Digital i/o.

O termo I/O significa input/output, ou seja , entrada / saída

Os microcontroladores PIC possuem pinos físicos destinados à comunicação de dados com os circuitos externos. Através desses pinos podemos enviar níveis lógicos (0 ou 1) para, por exemplo, acender ou apagar um LED, acionar displays LCD, ler botões e sensores, etc.

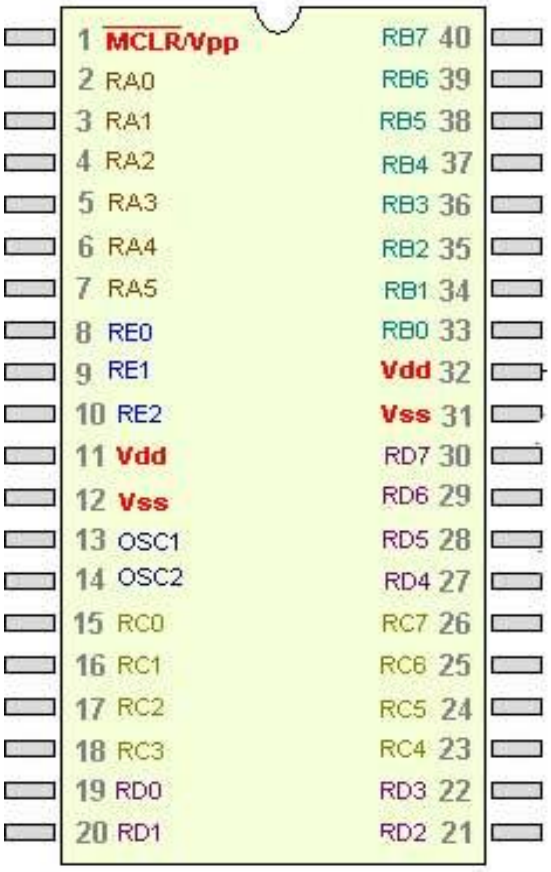
Exemplo : O microcontrolador PIC16F877A da *Microchip* possuem 34 pinos de escrita e leitura (I/O), os quais estão divididos em quatro portas chamadas: PORTA, PORTB, PORTC, PORTD e PORTE.

Cada porta de I/O possui dois registradores que controlam suas funções:

um registrador **PORT** e um registrador **TRIS.**

O **registrador TRIS** configura cada pino como entrada ou saída.

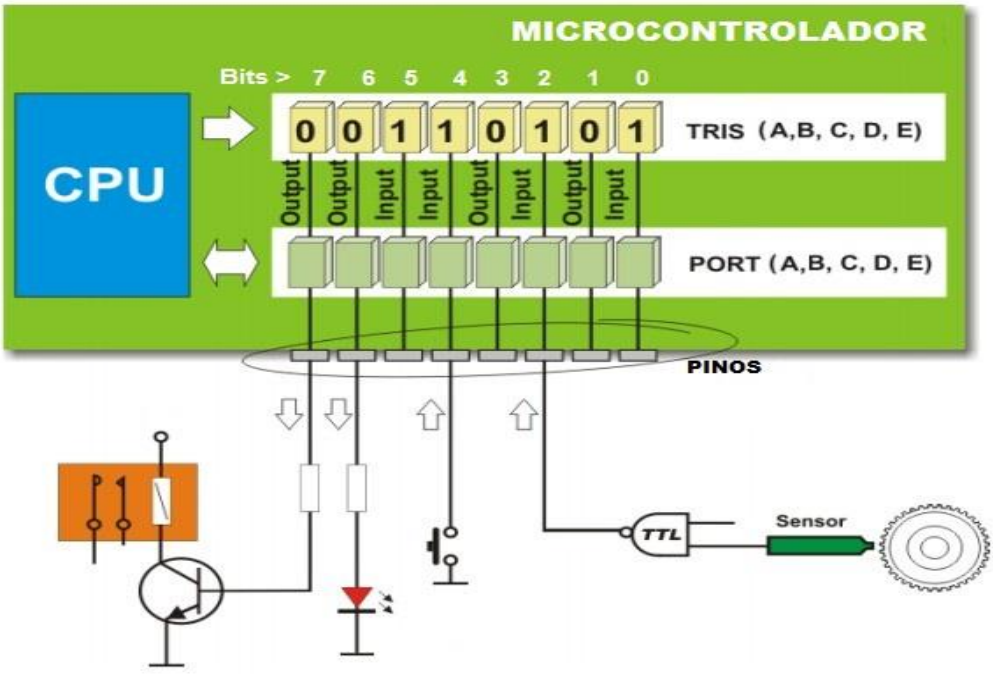
O **registrador PORT** é ligado diretamente a saída do chip. A escrita no registrador PORT altera a saída chip.



Cada porta de I/O possui dois registradores que controlam suas funções:

um registrador **PORT** e um registrador **TRIS.**

O **registrador TRIS** configura cada pino como entrada ou saída.




```
TRISB = 0b00110101;  //Binário

TRISB = 0×35; //Hexadecimal

TRISB = 53 //Decimal
```

O **registrador PORT** é ligado diretamente a saída do chip. A escrita no registrador PORT altera a saída chip.

```
portd = 0;      // nível baixo em todas saídas

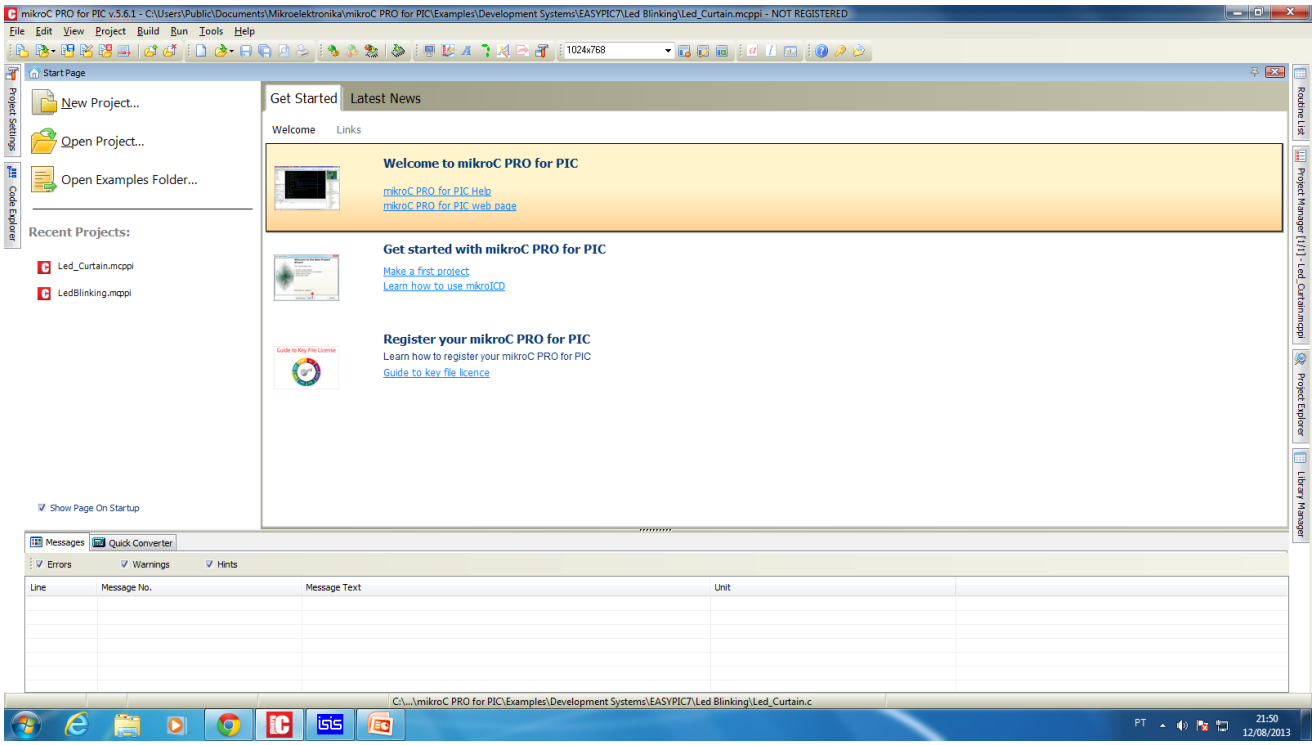
portd = 0b11111111;   // nível alto em todas as saídas do port b.
```

Os pinos que possuem função opcional de entrada analógica, devem ser configurados com digital caso deseje utiliza-lo como entrada ou saída digital, podendo usar a configuração.

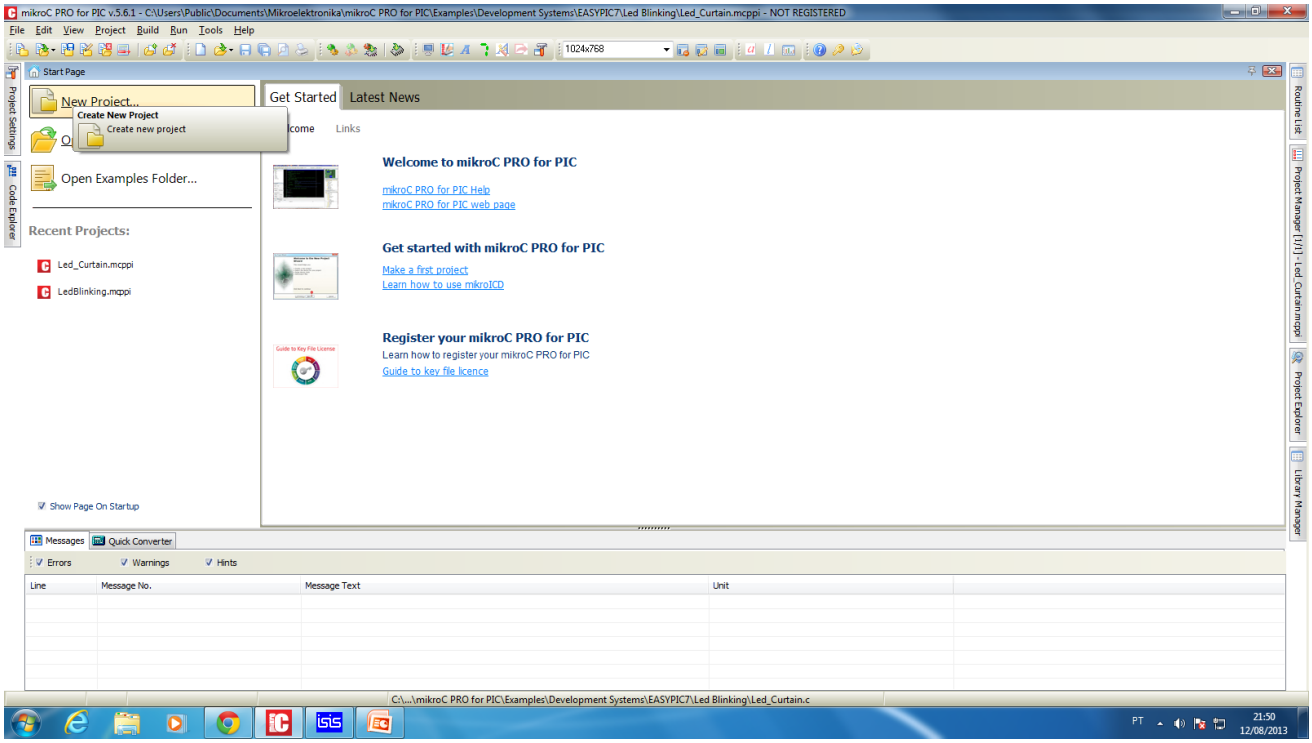
```
Adcon1=0b00001111;
```

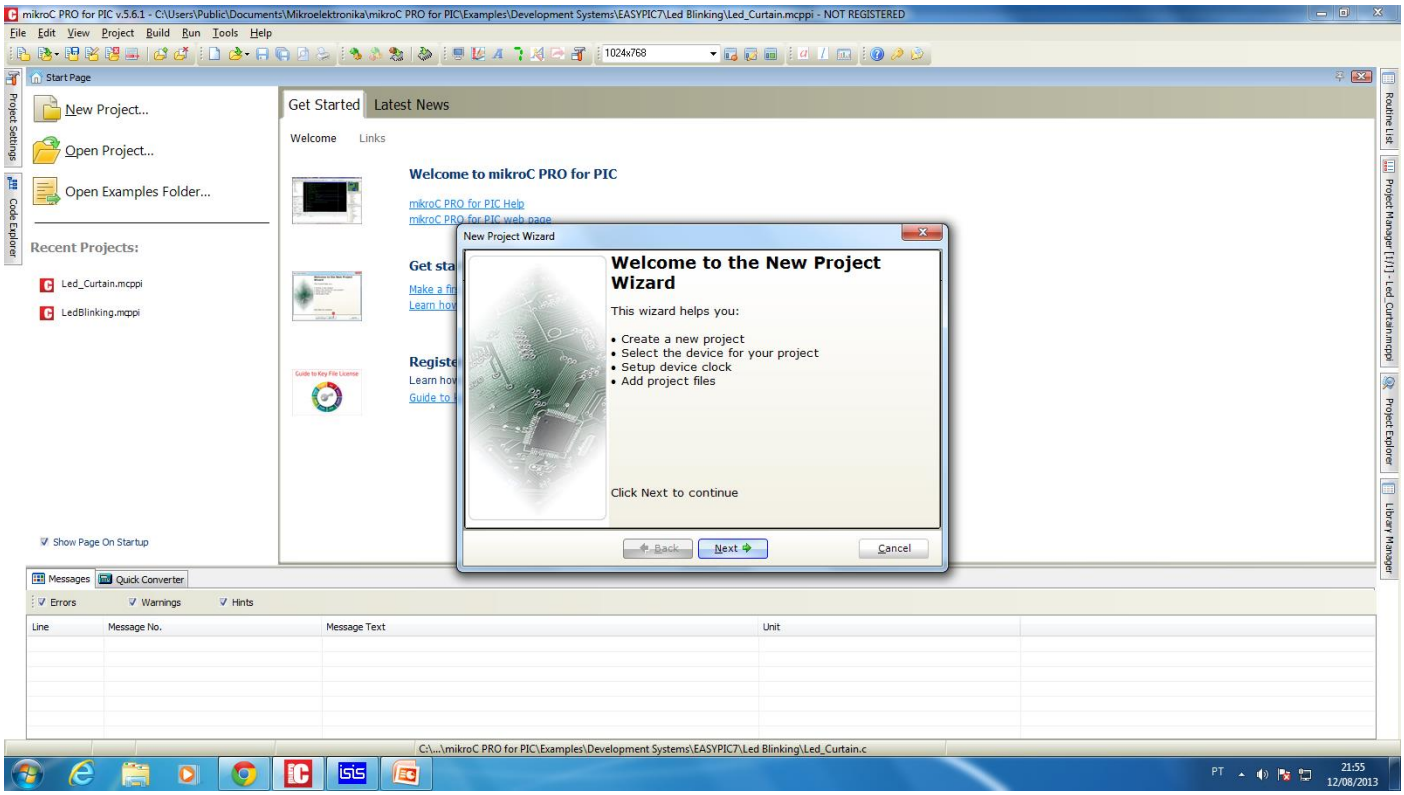
No capítulo sobre conversor analógico digital, essa configuração será explicada de forma detalhada.

5 Criando Projeto no MiKroC e simulando no Proteus.



Tela inicial do MiKroC , clicar em Create New Project (Criar novo Projeto)





Nessa tela aparece a descrição dos próximos procedimentos, apenas clicar em Next.

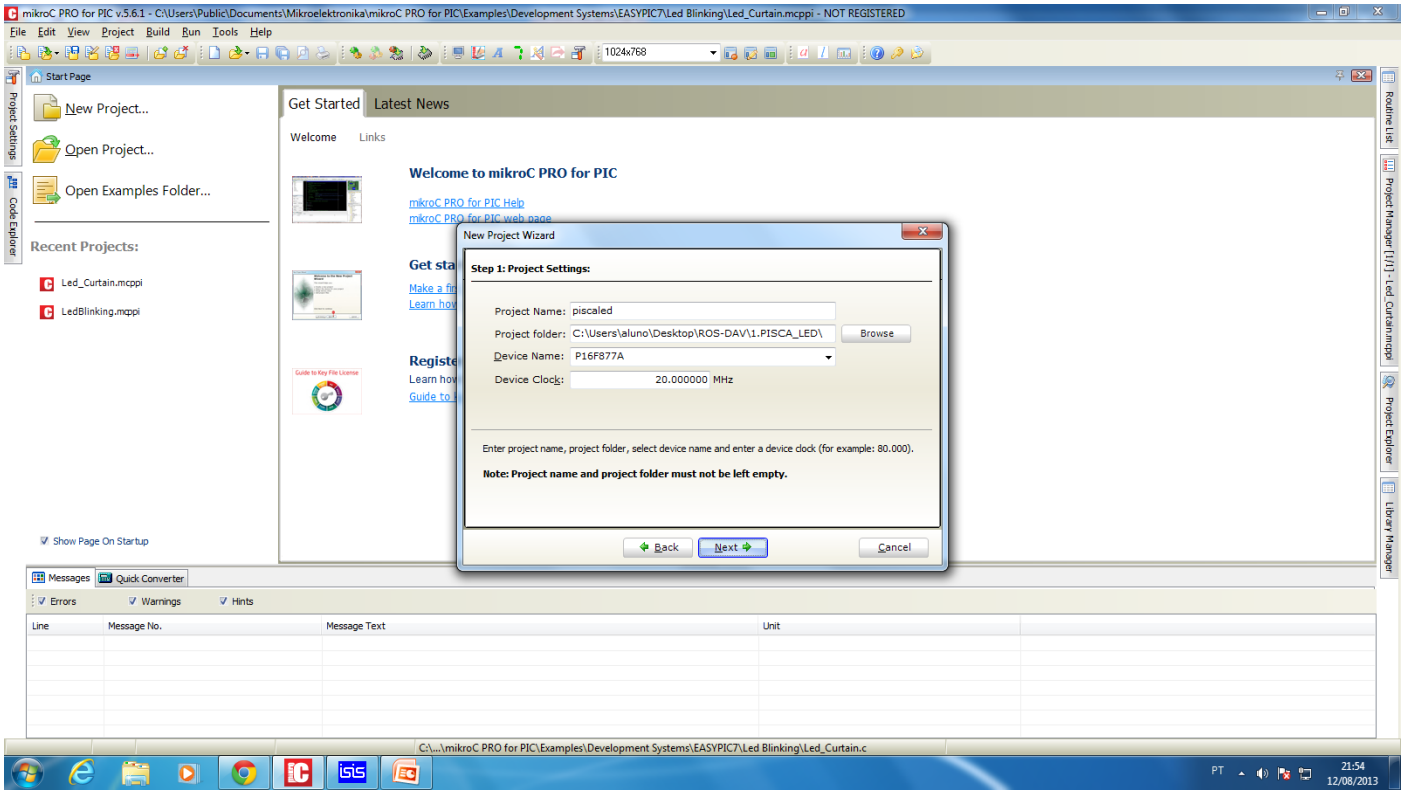
Os próximos passos já aparecem descritos nessa tela.

Create a New project – criar um novo projeto.

Slect the device for you Project, - Selecionar o dispositivo para seu projeto.

Setup device clock – Selecionar o Clock (Cristal oscilador) do seu dispositivo.

Add Project files – Adicionar arquivos do projeto.

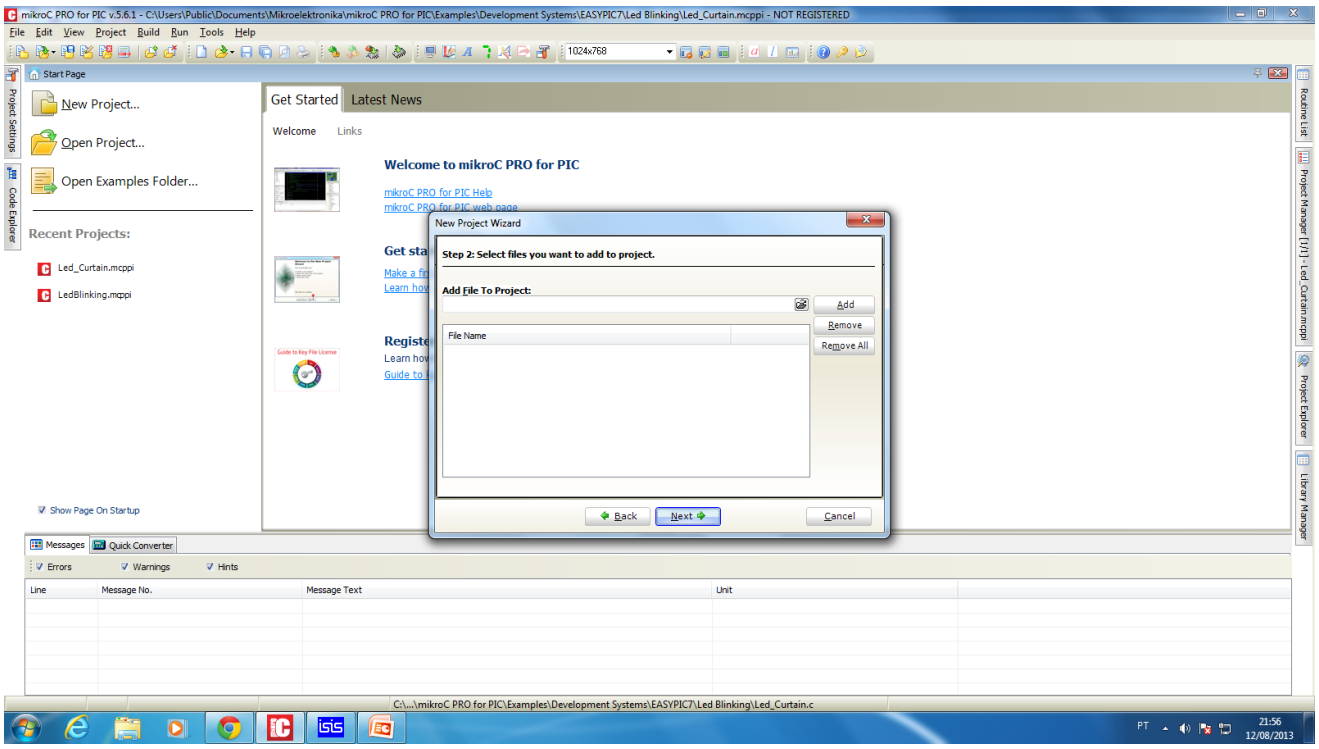


Nessa tela é inserido o Project Name, Nome do Projeto, evite caracteres especiais, espaços, acentuação , crie m nome simples.

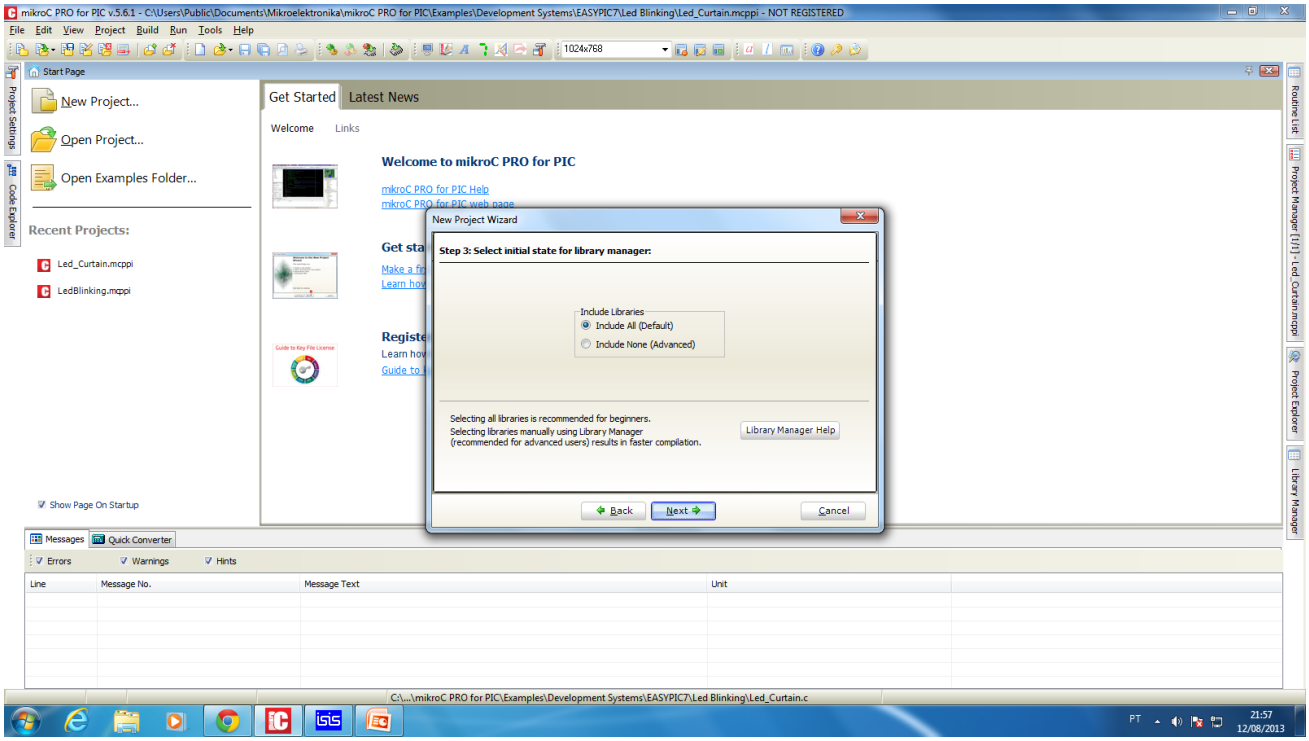
Project Folder : Pasta do Projeto, selecione uma pasta para o seu projeto, de preferência a criação de uma pasta por projeto.

Device Name: Nome do dispositivo. Selecione o disposto que deseja utilizar, nesse exemplo foi utilizado do Pic16f877A.

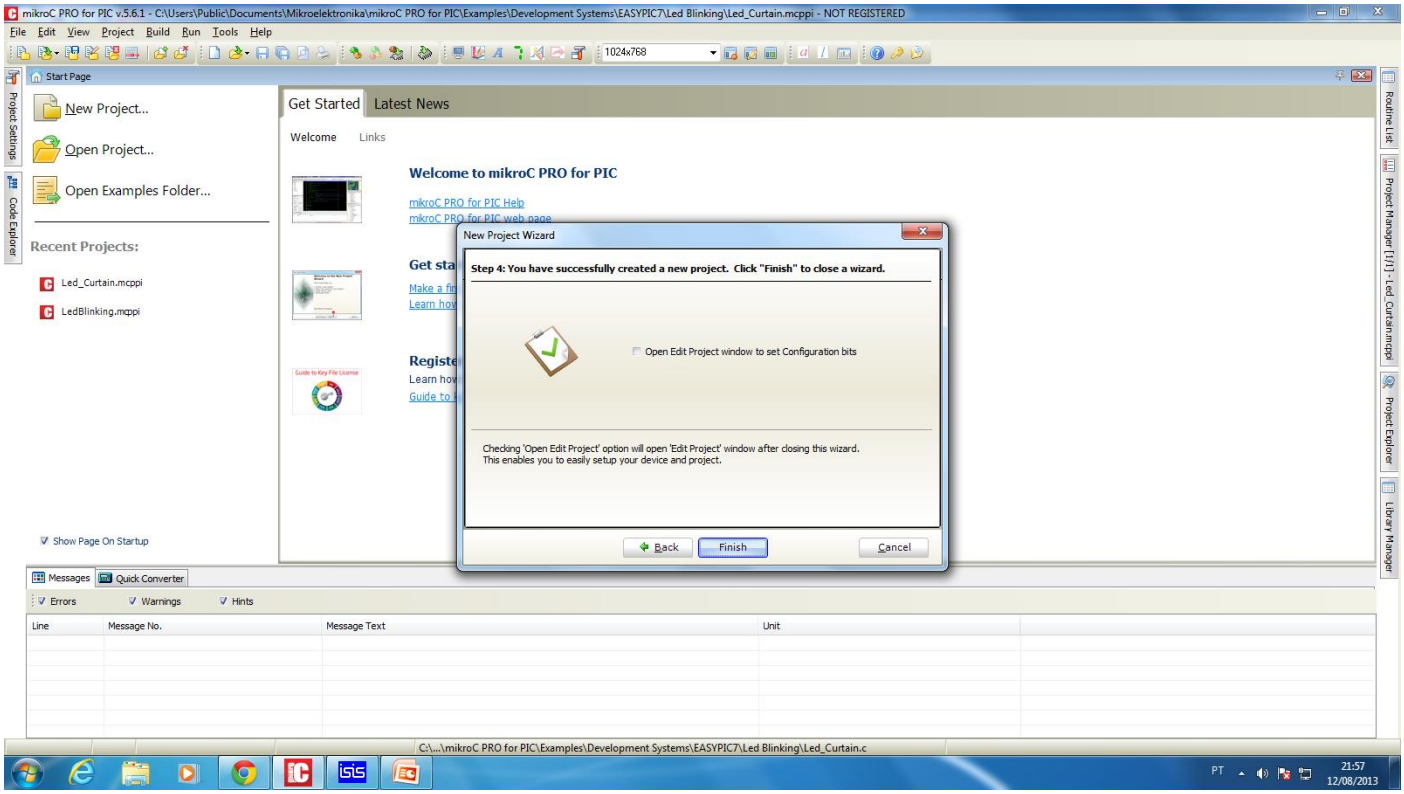
Device Clock: Clock do dispositivo, selecione a velocidade de Clock do dispositivo, o mesmo que esta no Cristal Oscilador do seu Hardware.



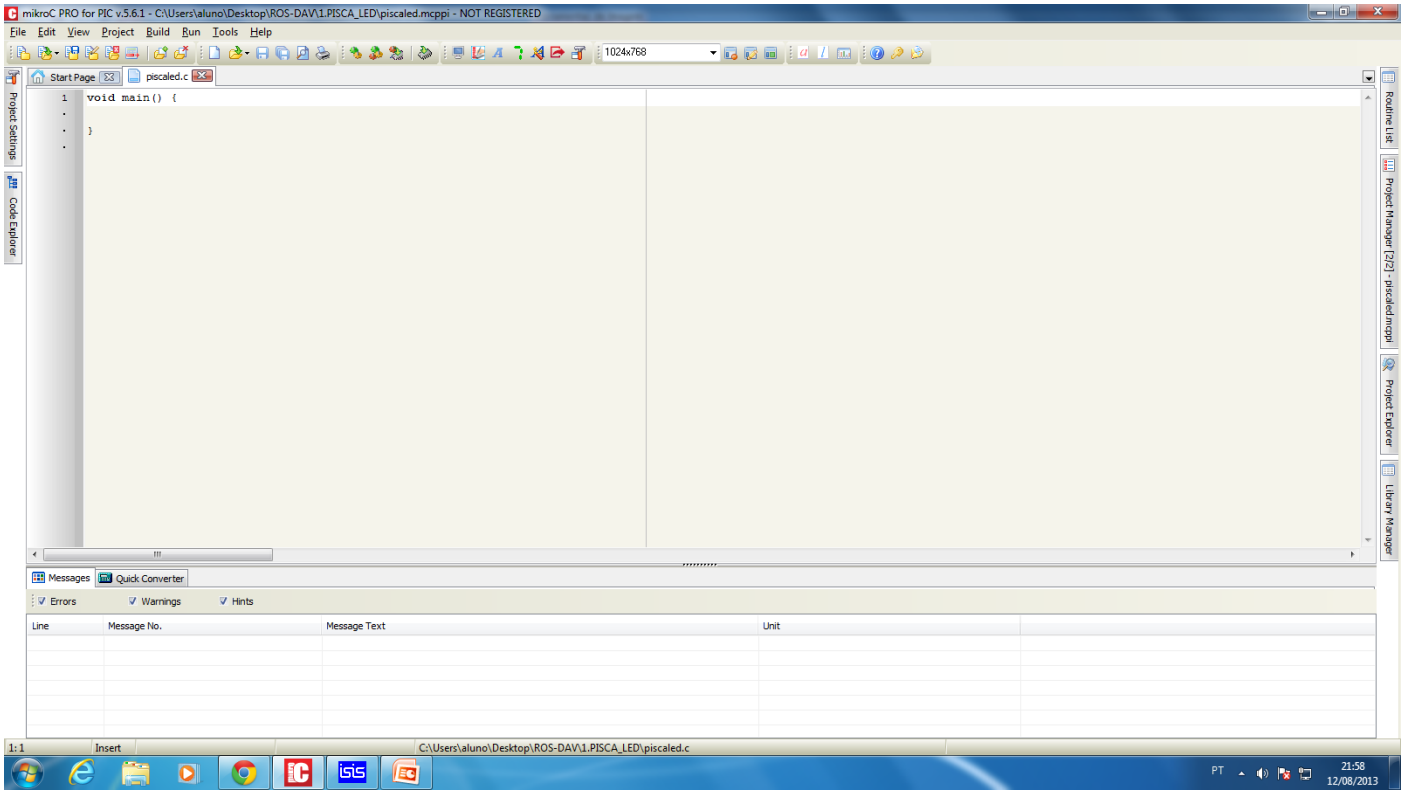
Tela para adicionar arquivos ao seu projeto, bibliotecas, etc. Um projeto novo não precisa ter arquivos adicionados.

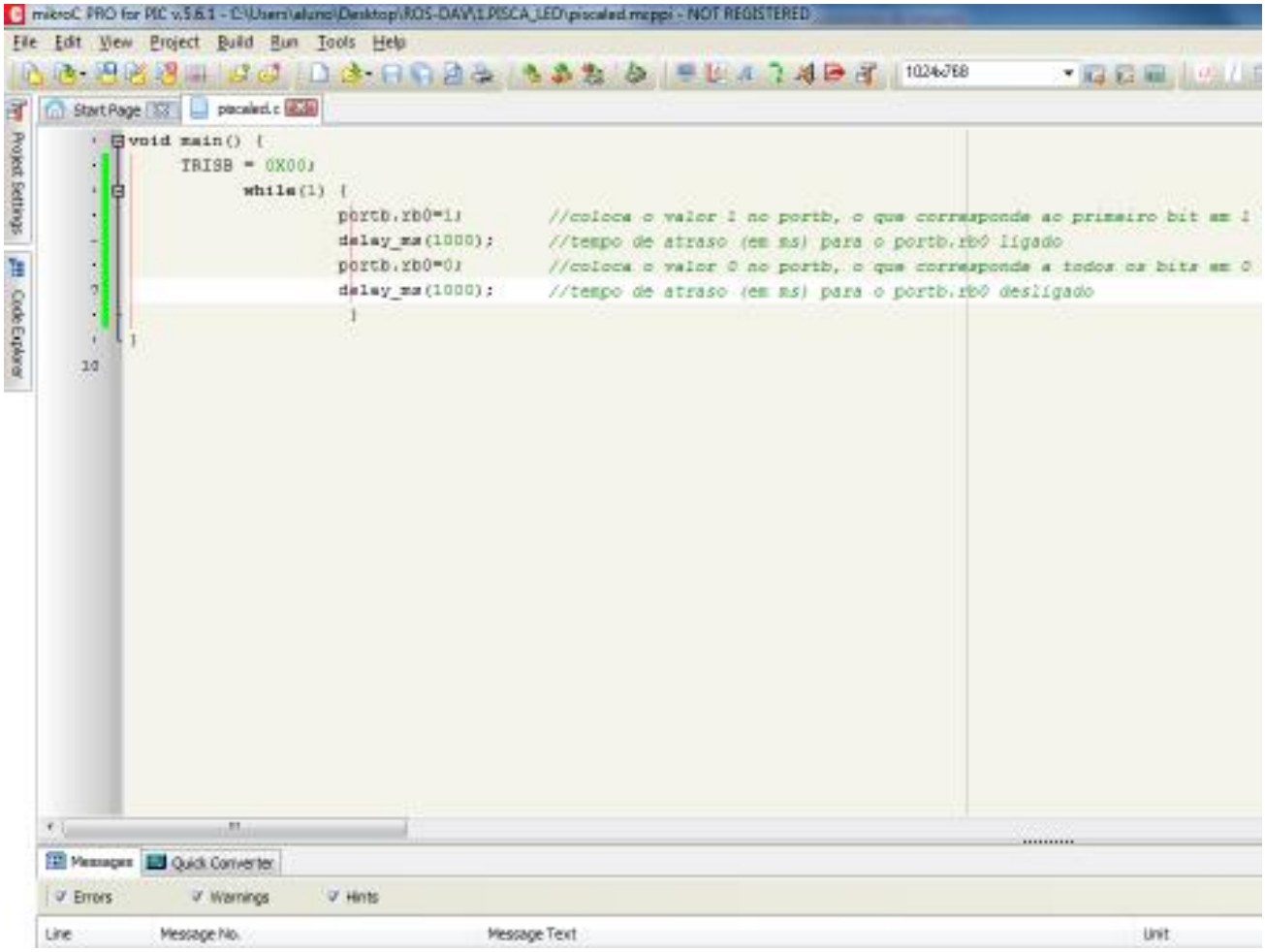


Clicar em next , incluindo as bibliotecas padrão.

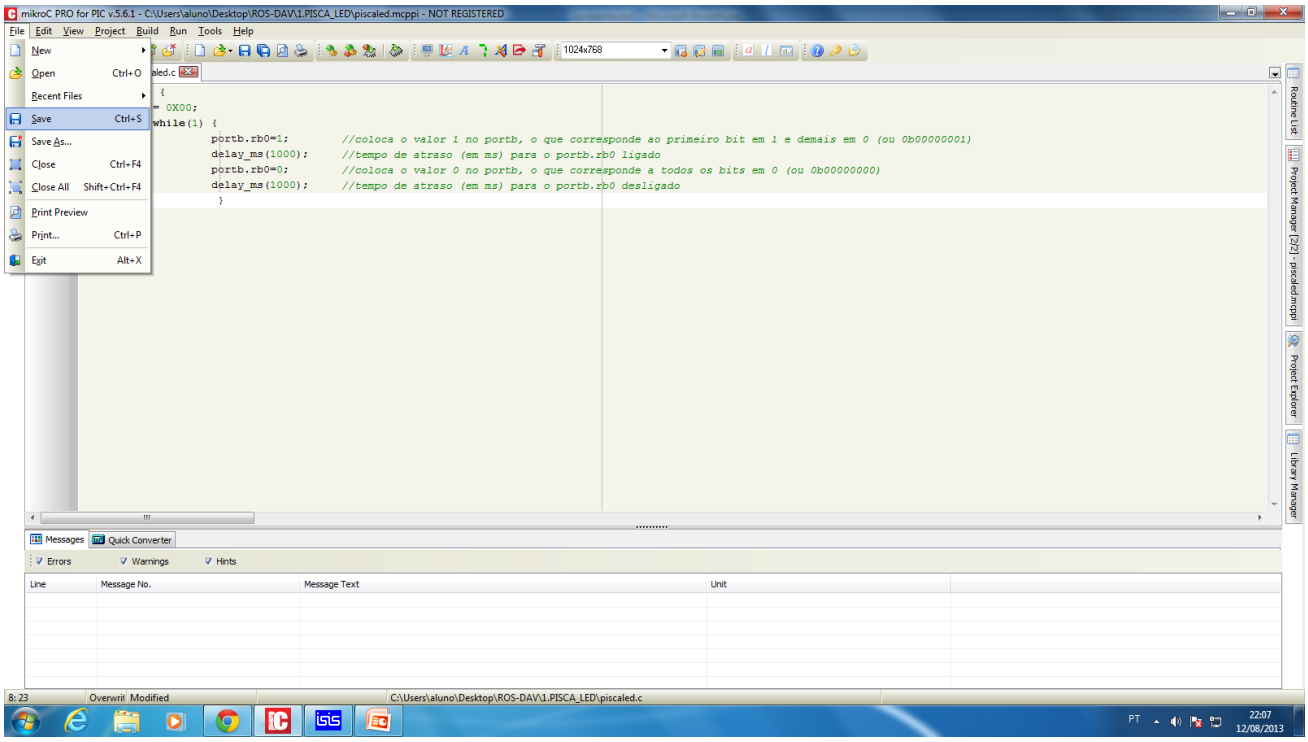


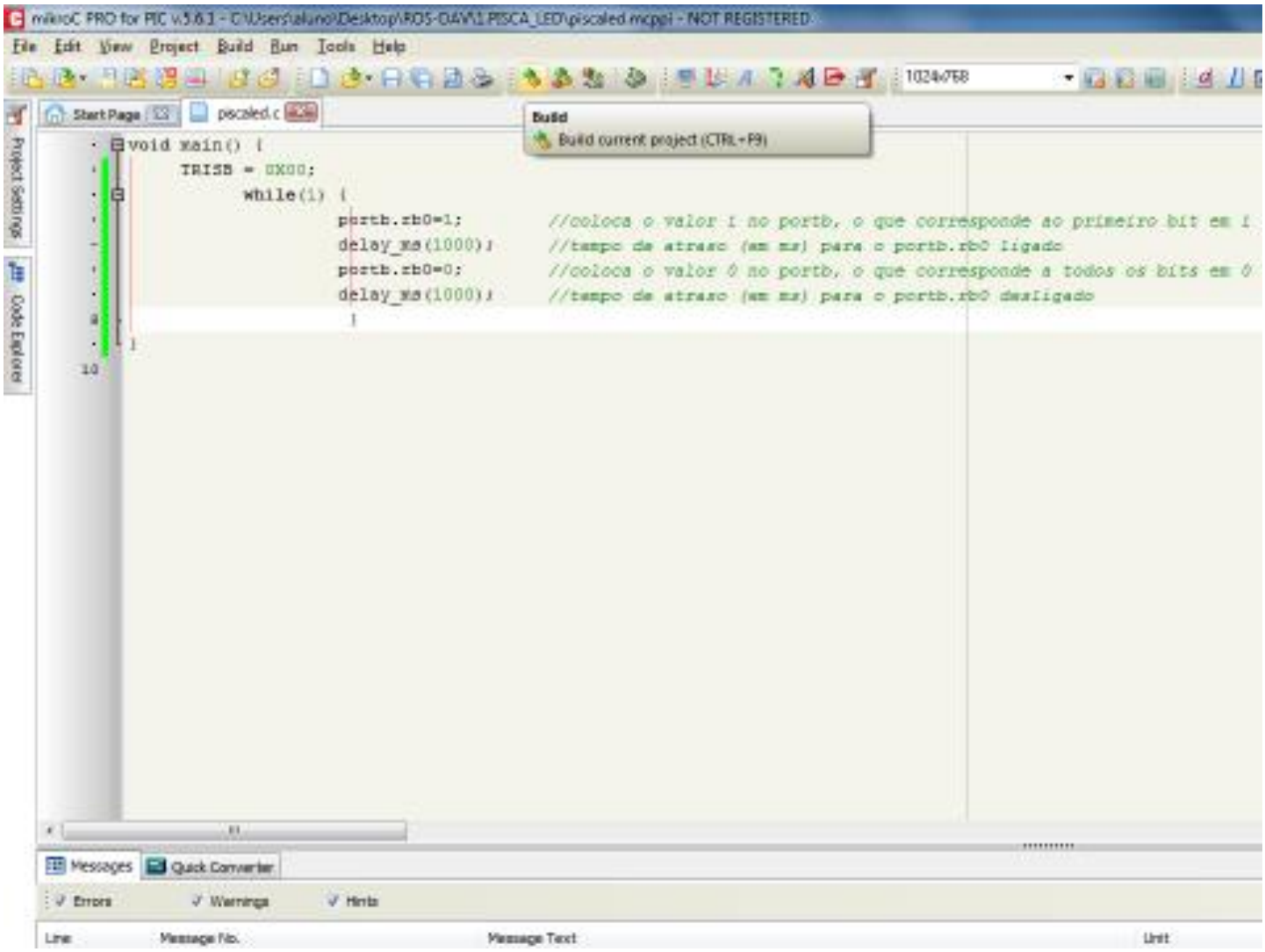
Finish, para finalizar a criação do projeto e ir para a tela de criação do código.


mikroC PRO for PIC v.5.6.1 - C:\Users\aluno\Desktop\ROS-DAV\1.PISCA_LED\piscaled.mcppi - NOT REGISTERED
File Edit View Project Build Run Tools Help
1024x768
Start Page
piscaled.c
1 void main() {
}
Project Settings
Code Explorer
Routine List
Project Manager [2/2] - piscaled.mcppi
Project Explorer
Library Manager
Messages
Quick Converter
Errors
Warnings
Hints
Line
Message No.
Message Text
Unit
1:1
Insert
C:\Users\aluno\Desktop\ROS-DAV\1.PISCA_LED\piscaled.c
PT
21:58
12/08/2013



Criação do código piscaled.



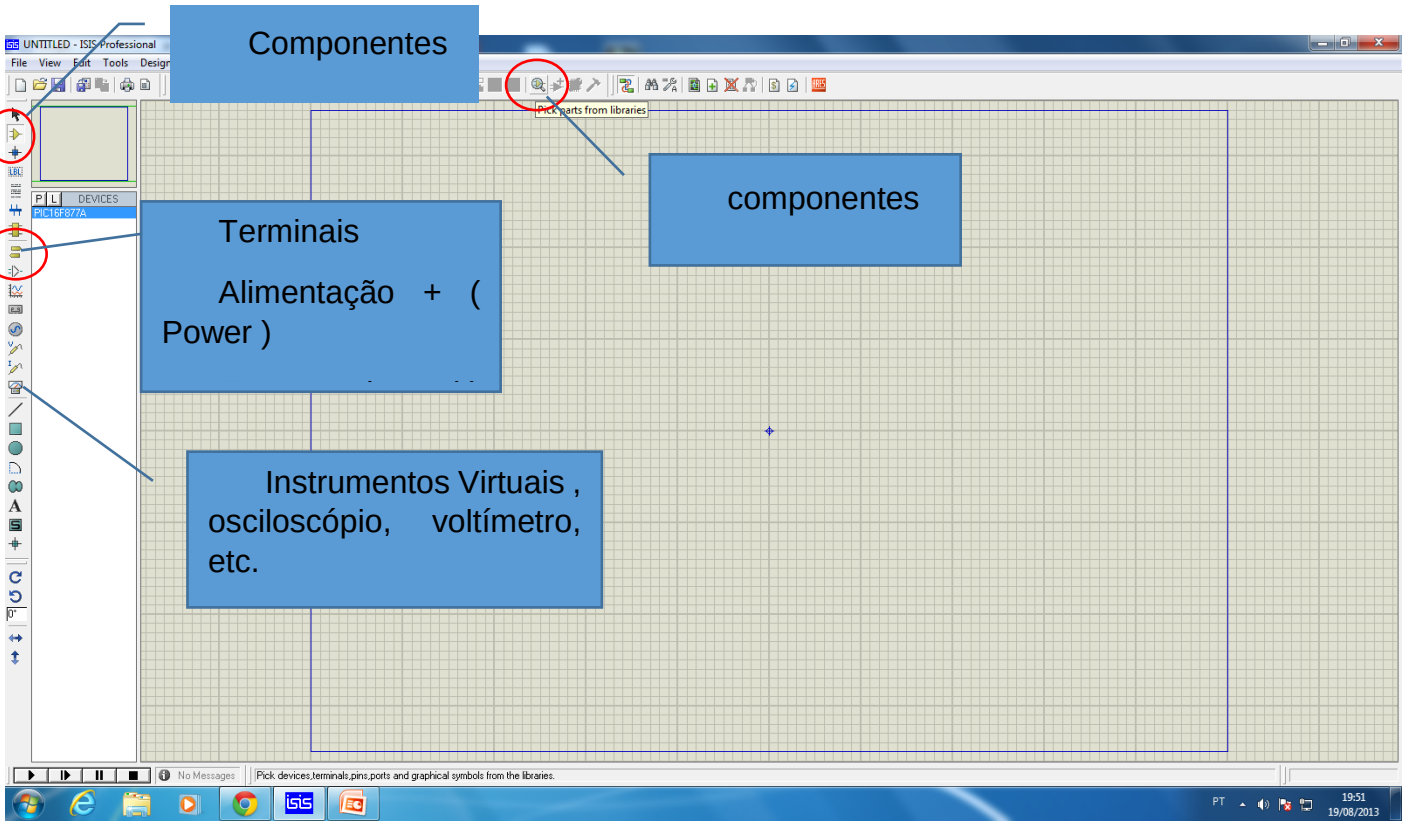


Para compilar o projeto, clique em Build, onde o projeto será construído, gerando o arquivo de projeto para ser gravado no microcontrolador com a extensão, hex.

Nesse exemplo o arquivo criado é o piscaled.hex, esse arquivo pode ser usado para gravação no dispositivo e também para a simulação em um simulador.

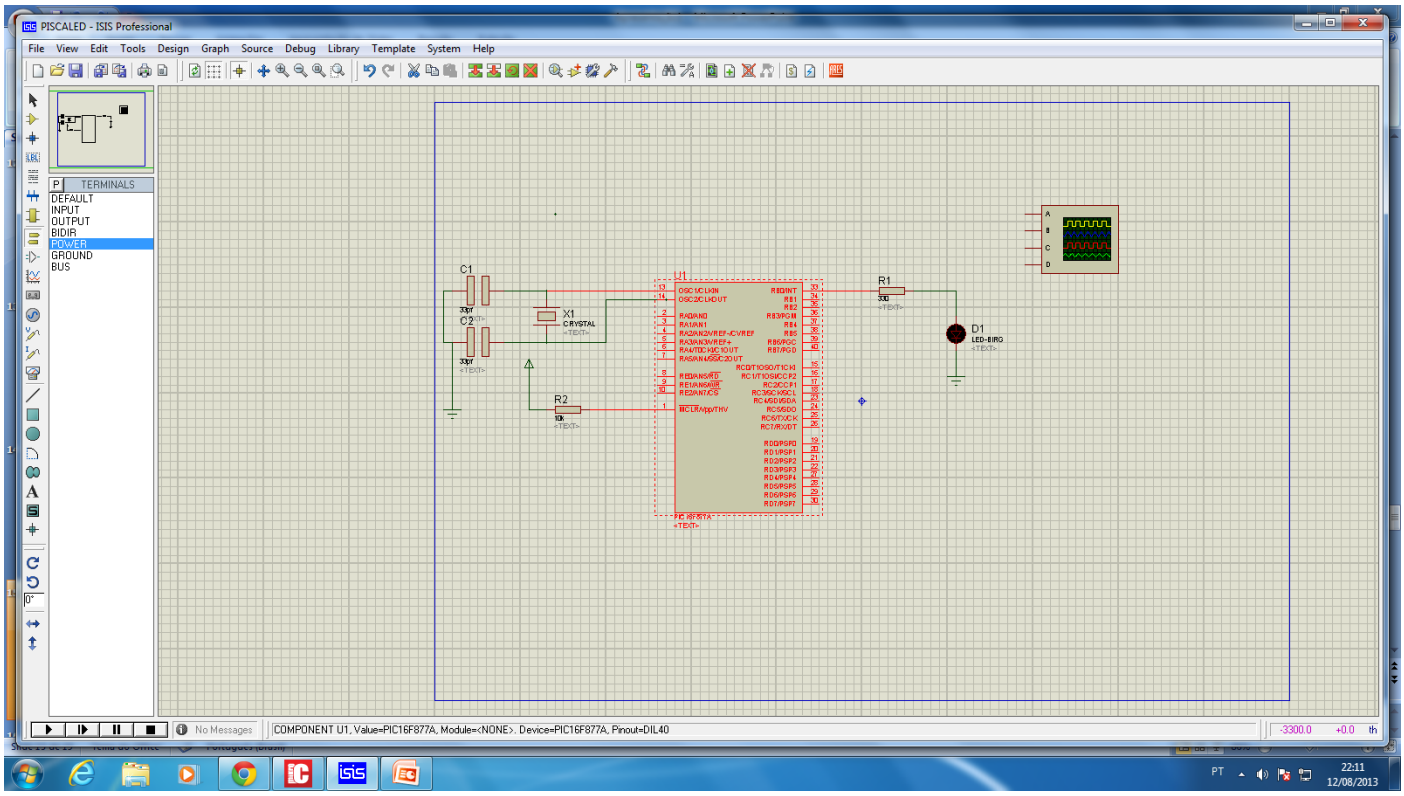
6 Criando um Hardware no Proteus.

Alguns ícones que é preciso conhecer para iniciar um projeto no Proteus Isis.

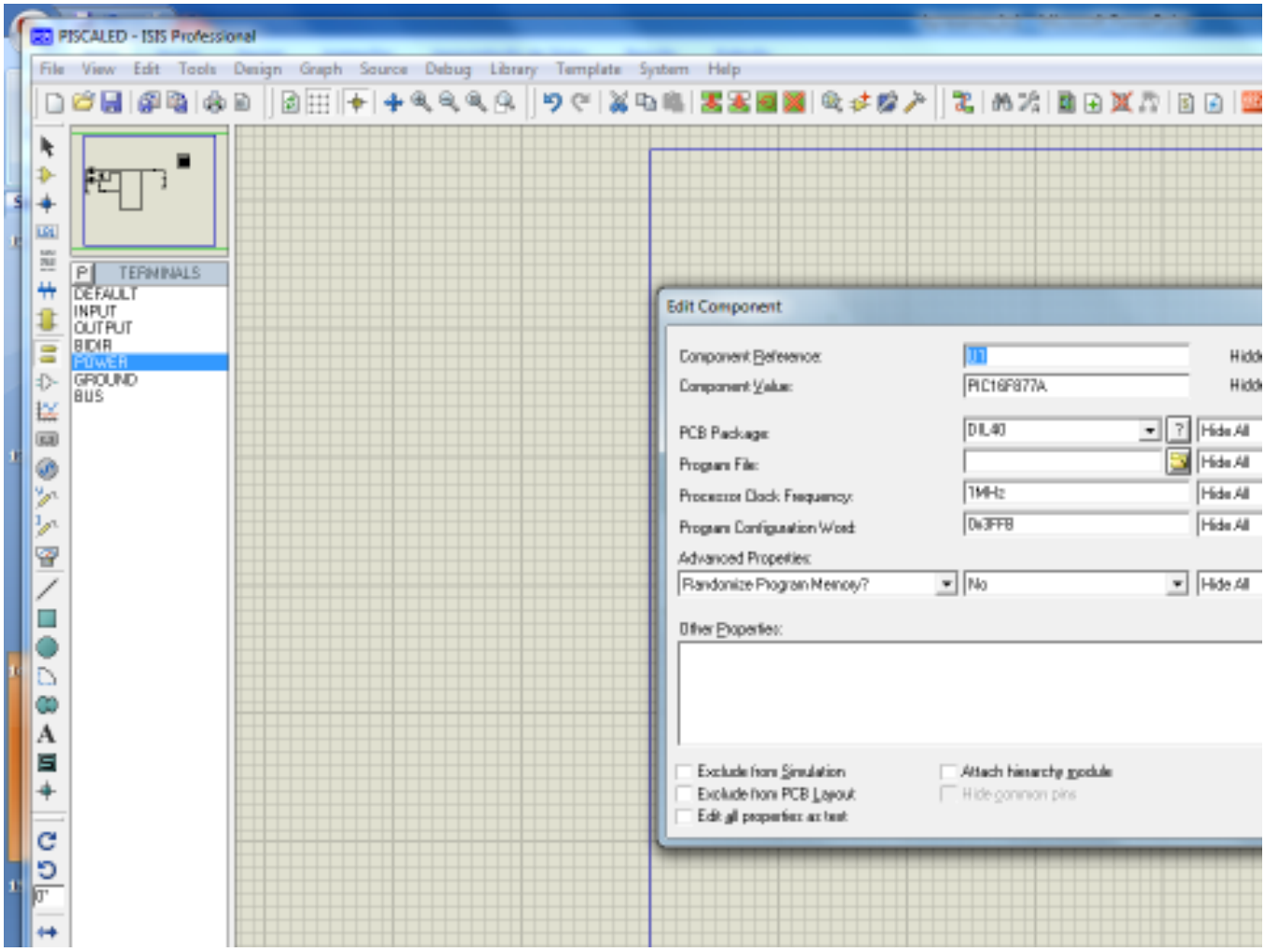


7 Simulando o Software Criado no MiKroC no Proteus.

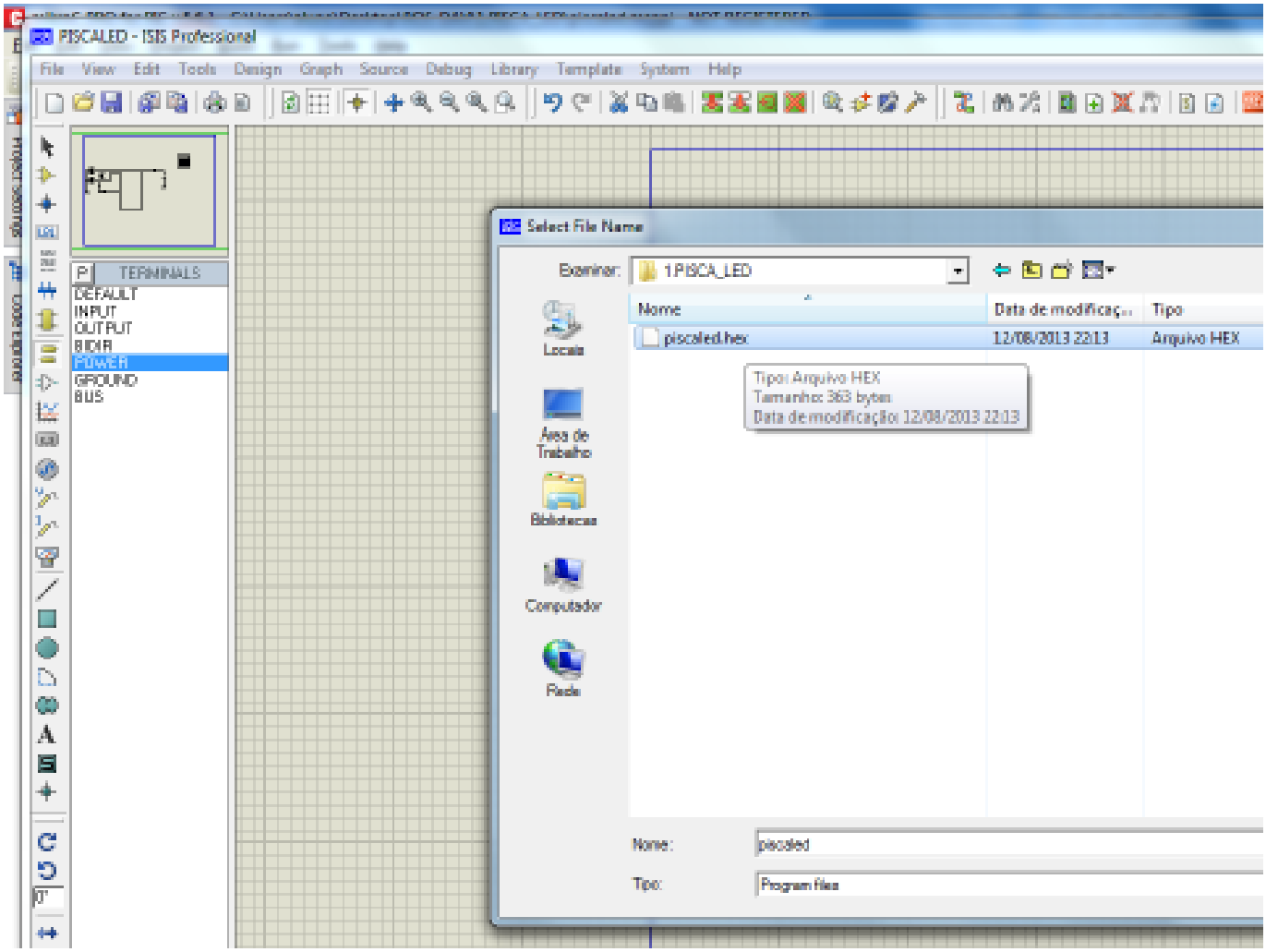
Com o software pronto, podemos simular o software no Isis (proteus) ou outro simulador.

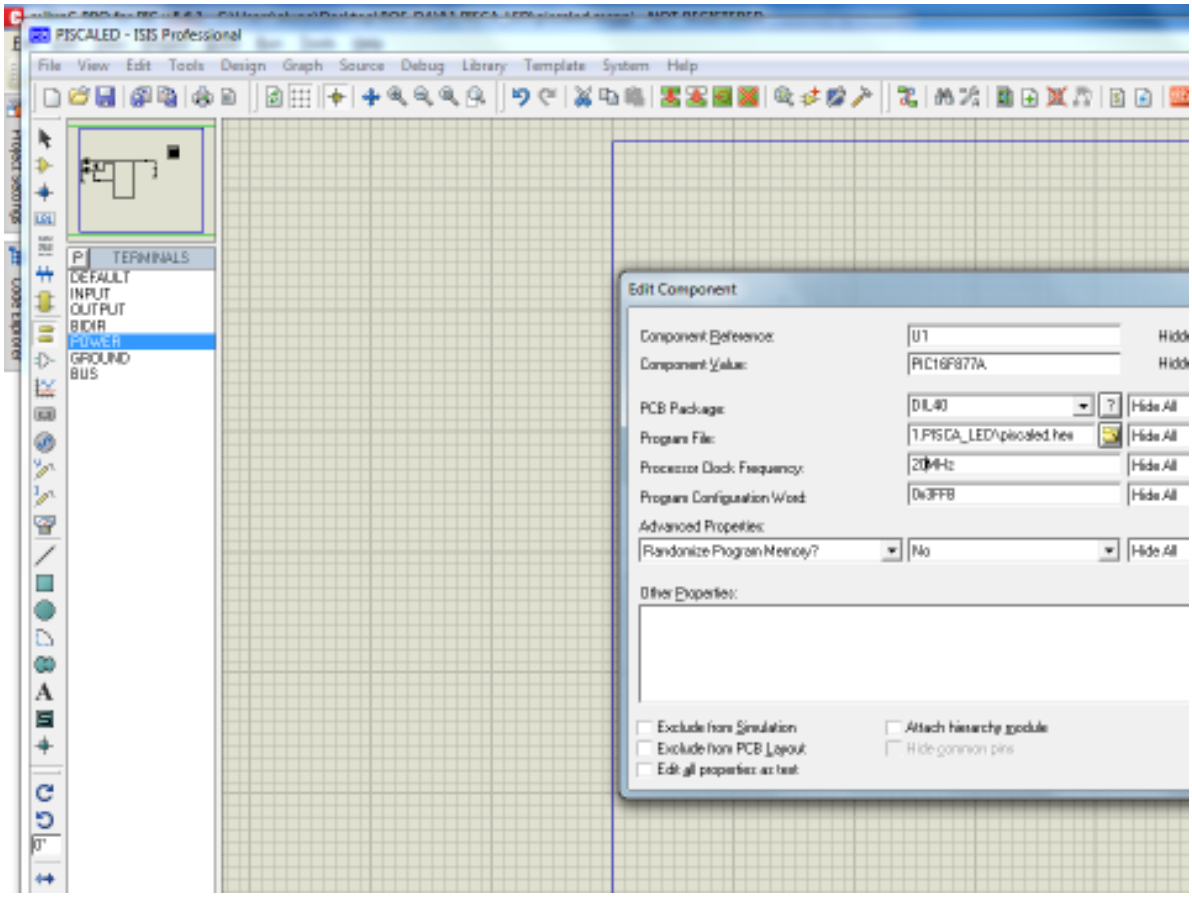


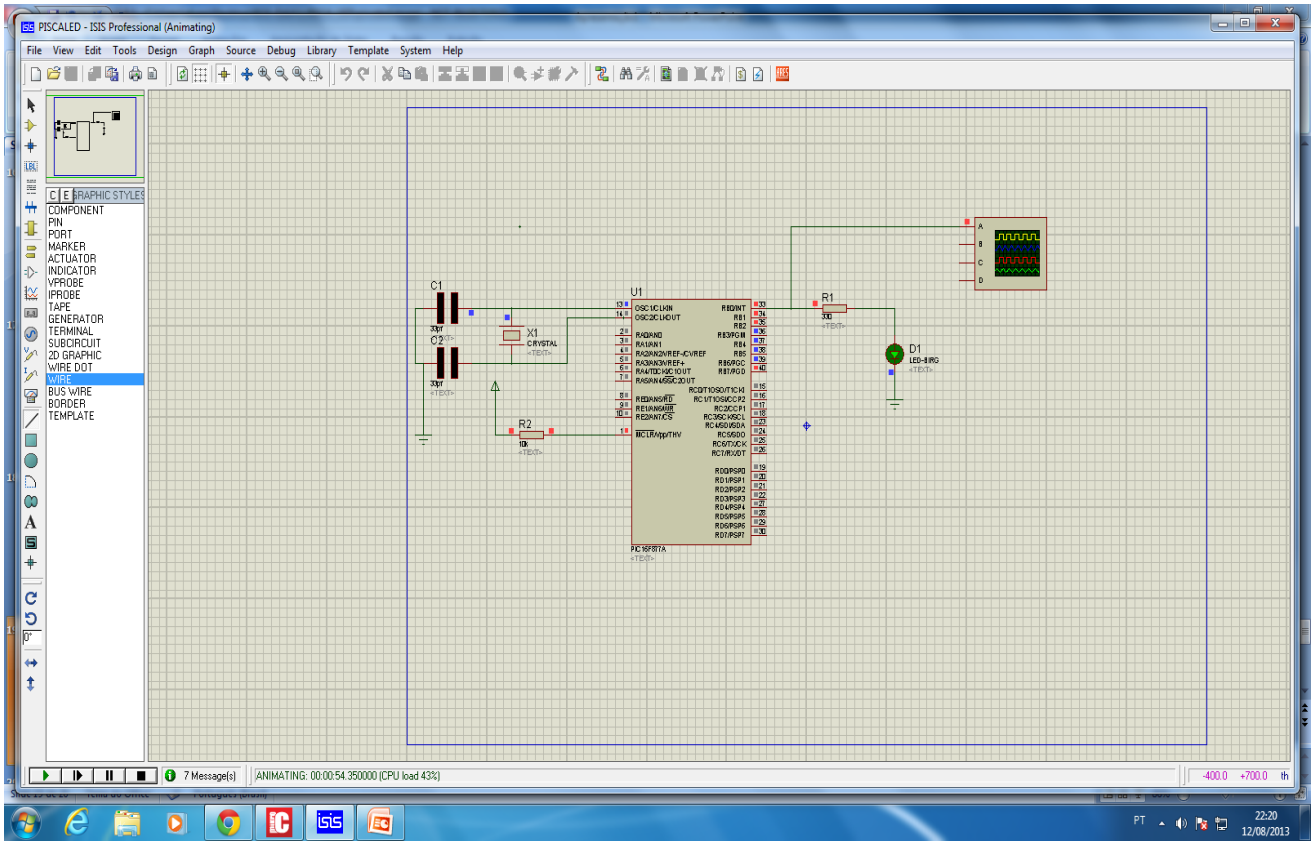
Clicar no microcontrolador para abrir a edição do componente.



No campo Program File , deve ser feito o link do seu software, e no campo Processor Clock Frequency, deve ser colocado o mesmo Clock que foi colocado no projeto.

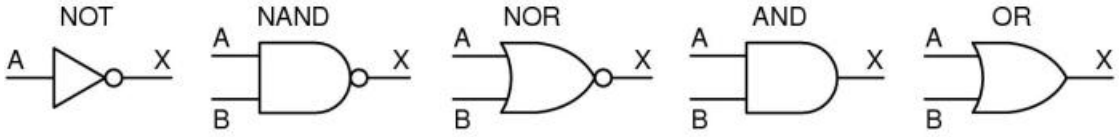

PISCALED - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
TERMINALS
DEFAULT
INPUT
OUTPUT
BIDIR
POWER
GROUND
BUS
Select File Name
Examinar:
1.PISCA_LED
Nome
Data de modificaç...
Tipo
piscaled.hex
12/08/2013 22:13
Arquivo HEX
Tipo: Arquivo HEX
Tamanho: 383 bytes
Data de modificação: 12/08/2013 22:13
Locais
Área de Trabalho
Bibliotecas
Computador
Rede
Nome:
piscaled
Tipo:
Program files





Com o software inserido no microcontrolador, clique no símbolo Play, para iniciar a simulação. Caso faça alterações no software mas mantenha o mesmo projeto, não é necessário refazer o link , basta clicar em Stop e depois em Play, para simular o software já atualizado.

8 Portas Lógicas



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

9 Comandos relacionais

`==`	**Igual a**
`!=`	**Diferente**
`>=`	**Maior ou igual**
`>`	**Maior que**
`<`	**Menor que**
`<=`	**Maior ou igual**

Operadores Lógicos em linguagem C

||	**OU lógico**
&&	**E lógico**
!	**Negação**

10 Comandos condicionais.

O comando IF é um comando condicional, ou seja ele verifica se o resultado da expressão é verdadeira ou falta, e executa ou não a operação relacionado a essa condição.

```
if (expressão lógica )
   {   comando 1;
       comando 2;
   }
```

Nesse caso verifica-se se a expressão é verdadeira, e caso seja falsa nenhuma ação é feita.

O comando if também tem a opção de executar a ação cada seja verdadeira, e outra ação caso a expressão lógica do if seja falta.

```
if (expressão lógica )
     {comando1; // executado se expressão lógica for verdadeira

      }
else {  // executado se a expressão lógica for falsa.

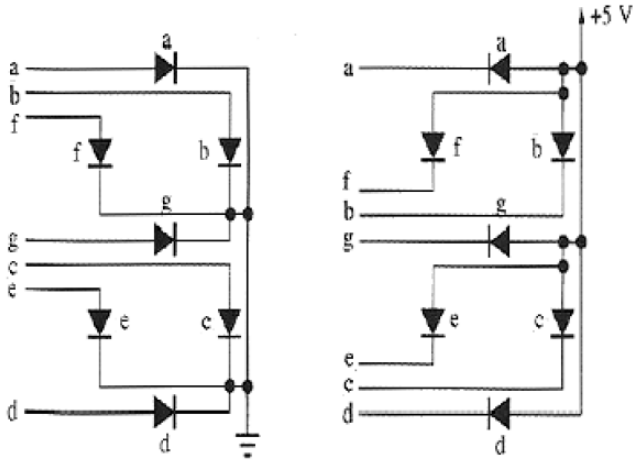
comando2; // executado se "expr_log" for falsa
comando3; // executado sempre, independente
      }
```

comando while

```
Iniciar a variável de controle
 Enquanto (condição) faça
 Início
    Instruções;   // dentro das instrução a variável de controle relacionado a condição
                   // pode ser alterada ou não.
 Fim;
```

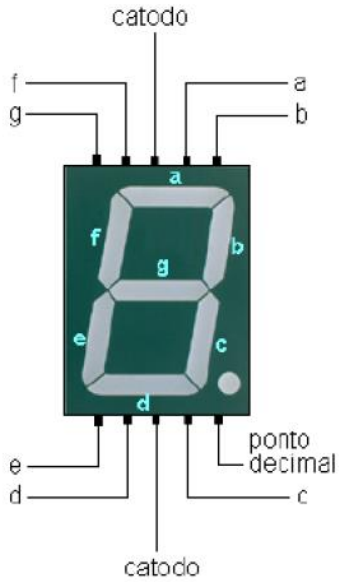
11 Display de 7 segmentos

O display pode ser do tipo **ânodo comum**, ou seja os terminais ânodo de todos os segmentos estão interligados internamente e para o display funcionar, este terminal comum deverá ser ligado em Vcc, enquanto que o segmento para ligar precisa de estar ligados no GND.
Já o display **cátodo comum**, é o contrário, ou seja, o terminal comum, deverá ser ligado ao GND e para ligar o segmento é necessário aplicar Vcc ao terminal.
Actualmente, o display mais comercializado é o do tipo ânodo comum.



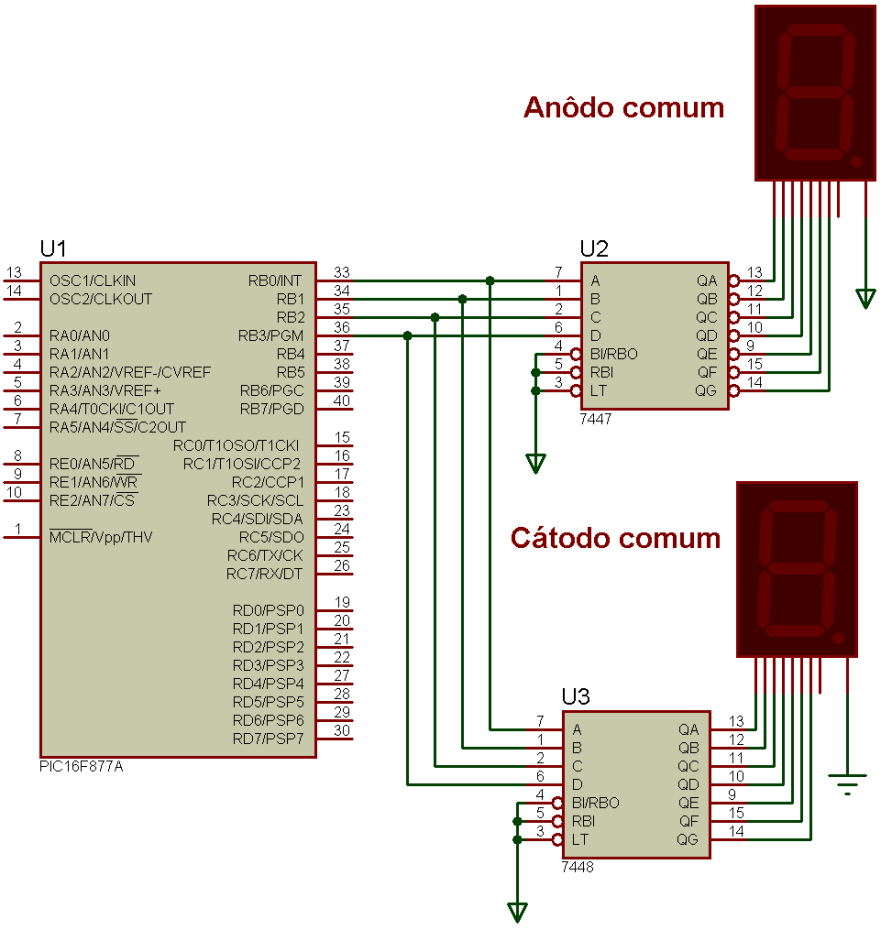
Cátodo comum Ânodo comum

O display de sete segmentos é formado com sete leds, posicionados de modo a possibilitar a formação de números decimais. A figura representa uma unidade do display genérica, com a nomenclatura de identificação dos segmentos.



11.1 Display de 7 segmentos utilizando o conversor 7447 ou 7448.

Uma das formas de ligação do display, é utilizar um Ci que faz a conversão do código BCD para o display de 7 segmentos. Como temos duas opções de display, Cátodo e Anodo comum, temos um ci próprio para display anodo, e outro próprio para display catodo conforme figura abaixo.

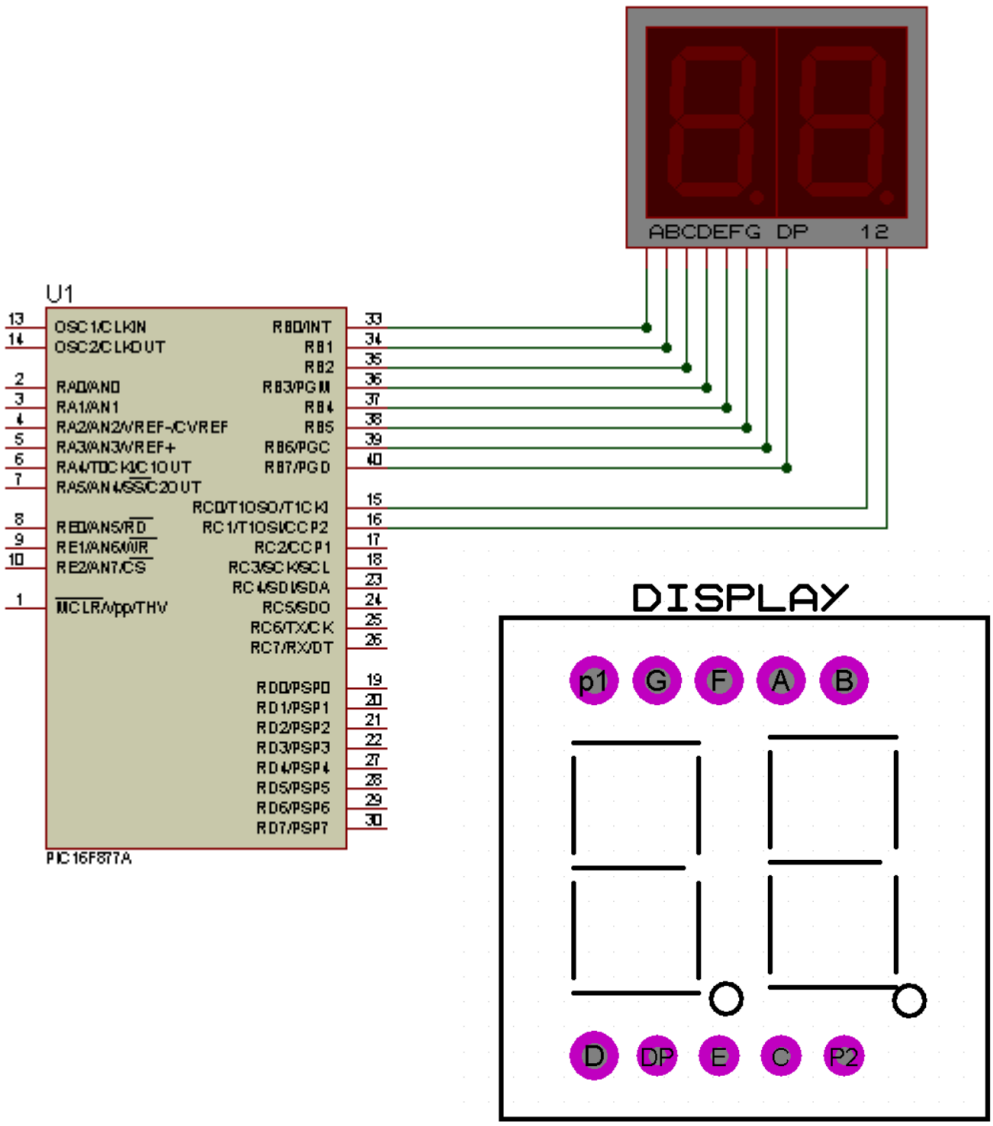


Código MikroC para uso do display com ci de conversão.

```
void main() {
 int a=0;     trisb=0;
     while(1)
                {portb=a;
                 a++;
                 delay_ms(500);
                 if(a>9) { a=0; }
                }
}
```

11.2 Display de 7 segmentos sem o conversor.

11.3 Display de 7 segmentos multiplexado




```
void main() {
int a=23, u,d,k;
trisb=0;
trisc=0;
while(1)
{   d=a/10;          u=a%10;
portc=0b00000001;    // seleciona o display da unidade
if(u==0){ portb=63; }
```

```
if(u==1){portb=6; }
if(u==2){portb=91; }
if(u==3){portb=79; }
if(u==4){portb=102; }
if(u==5){portb=109; }
if(u==6){portb=125; }
if(u==7){portb=7; }
if(u==8){portb=127; }
if(u==9){portb=111;}
delay_ms(10);

portc=0b00000010; // seleciona o display da dezena
if(d==0){ portb=63; }
if(d==1){portb=6; }
if(d==2){portb=91; }
if(d==3){portb=79; }
if(d==4){portb=102; }
if(d==5){portb=109; }
if(d==6){portb=125; }
if(d==7){portb=7; }
if(d==8){portb=127; }
if(d==9){portb=111;}
delay_ms(10);
k++;
if(k>50) { a++; k=1; }
}   }
```

12 Vetor – exemplo com display de 7 segmentos.

No exemplo anterior o comando if foi utilizado para selecionar o número correto a ser enviado ao display, contudo existe uma forma mais eficiente de construir esse mesmo código utilizando um vetor.

Um vetor possui uma estrutura que pode armazenar uma determinada quantidade de valores do mesmo tipo, ou seja posso criar um vetor do tipo int, que pode armazenas variáveis somente do tipo int.

Sua representação comum é

Tipo nomedoVetor[tamanho] ;

Podemos também inicializar valores no vetor

Int x[2]={3,4};

Nesse caso a variável x[0] é igual a 3, e a variável x[1] é igual a 4.

Adaptando o exemplo do display multiplexado no vetor temos o exemplo abaixo.

```
// início do código
int unidade,dezena,x;

int seg7[10]={63,6,91,79,102,109,125,7,127,111};
// o vetor seg7 possui 10 posições,e já foi inicializado com
// os 10 valores.

void display7seg(int numero);  // protótipo da função

void main()
{
    trisd=0b00000000; // configura portd como saída digital
    trisc.rc4=0; // nível lógico 0 no pino c4
    trisc.rc5=0; // nível lógico 0 no pino c5

    while(1)        // loop infinito
            {
            X=10;     // conversão AD
            display7seg(x);  // chamada da função display7seg chamando
                             // utilizando a variável x
```

```
            }
}
  /// inicio das funções
  void display7seg(int numero)
      { // a função começa separando a unidade da dezena.
       unidade=numero%10;//resto da divisão por 10, resulta na unidade
       dezena=numero/10;   // divisão por 10 , resulta na dezena.
       portc.rc4=1; portc.rc5=0; // seleciona o display da unidade
       portd=seg7[unidade]; //a variável seg7 foi inicializado com os
                            // binários compatíveis com a formação do
                            // número no displa7 de 7 segmento.
        delay_ms(20);
        portc.rc4=0; portc.rc5=1;  // seleciona o outro display
        portd=seg7[dezena];        // envia o número da dezena.
        delay_ms(20);
      }
```

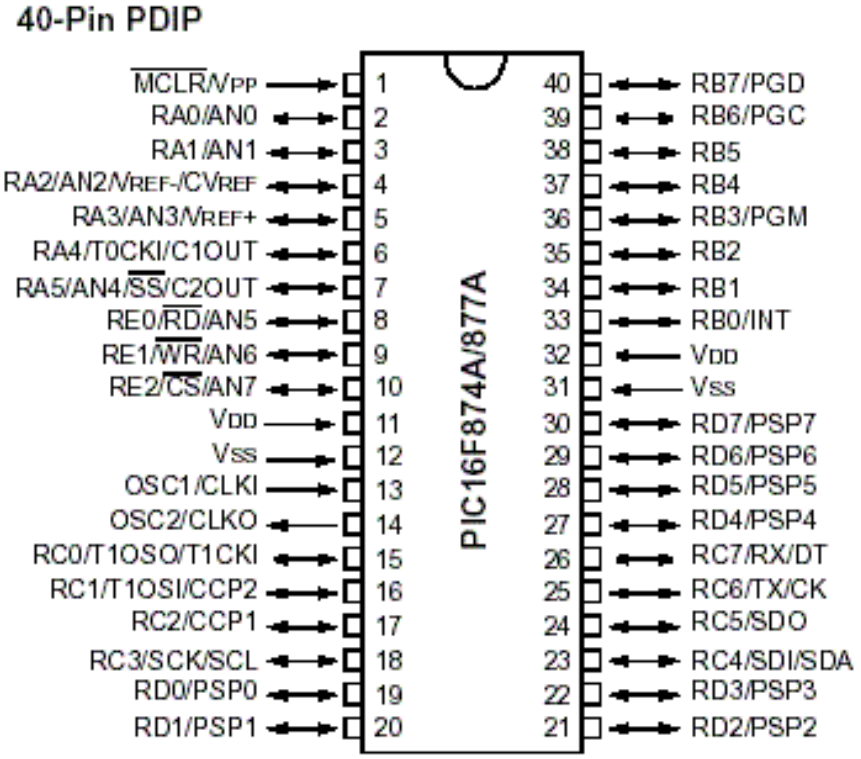
13 Quadro Resumo , entrada e saída digital.

Configuração (entrada e saída)		
Trisb=0;	Configura todo o port b como saída	0 entrada 1 saída
Trisb.rb0=0	Configura o pino rb0 como saída	0 saída
Trisb=1;	Configura todo o port b como entrada	0 entrada 1 saída
Trisb.rb0=1	Configura o pino rb0 como entrada	1 entrada
Trisb=0b00001111; ou Trisb=15;	Configura rb0 a rb3 como saída e rb4 a rb7 como entrada.	

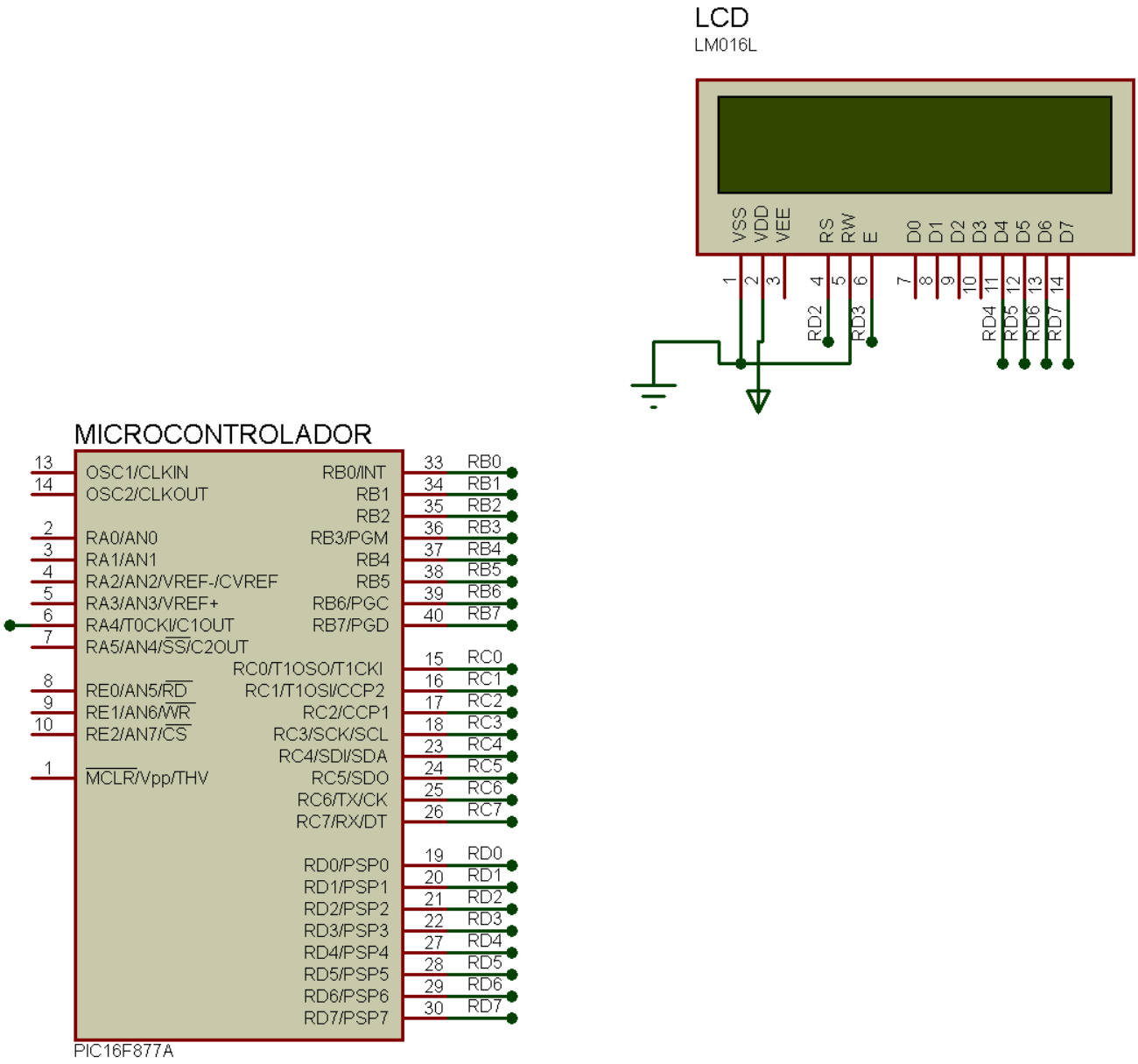
Controle dos pinos		
Controla todos os pinos do Portb com um número decimal	Portb=0;	Todo port b =0
Controla todos os pinos do Portb com um número binário	Portb=0b00000011; ou Portb=3;	Rb0=1 Rb1=1
Controla apenas um pino	Portb.rb0=1; saída alta Portb.rb0=0; saída baixa	
Controle de tempo		
Delay_ms(500);	Tempo em milisegundo Aceita somente números	
Vdelay_ms(variavel);	Aceita variáveis de números inteiros	

14 Exercícios entrada e saída digital

1. Faça o software que gere uma frequência de 1 Hz.
2. Faça o software que gere uma frequência de 10hz.
3. Faça o software que gere uma frequência de 1hz e aumente a frequência de 1 em 1 até 10 hz.
4. Faça um contador binário crescente de 0 a 10 .
5. Faça um contador binário decrescente de 10 a 0.
6. Faça um contador utilizando display de 7 segmentos (utilize o display 7segBCD que converte o código binário BCD no código de 7 segmentos.
7. Faça um contador crescente de 0 a 99 e apresente em dois displays de 7 segmentos multiplexado.
8. Faça um contador decrescente 99 a 0 e apresente em dois displays de 7 segmentos multiplexado e acenda um led quando chegar em 0.
9. Faça gerador de frequência com variação de 0 a 10hz, controlado por dois botões, um para subir o valor e outro para descer.



15 Display de LCD



Para utilizar um display de LCD, primeiro é necessário saber onde está ligado cada pino,

Para esse circuitos, a ligação é feita conforme tabela abaixo.

Pino do Microcontrolador	Pino do LCD
RD2	RS
RD3	E
RD4	RD4
RD5	RD5
RD6	RD6
RD7	RD7

No código deverá ser informado essa configuração para que a biblioteca do LCD no MiKroC possa se comunicar corretamente com o LCD.

Isso é feito através da tabela abaixo.

```
// LCD module connections
sbit LCD_RS at RD2_bit;  // pino RS do display está ligado no pino D2 do pic.
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connections
```

Os as configurações a direita de cada linha devem ser configuradas conforme ligação no Hardware e devem estar antes da função main(), no código em C.

Os principais comandos utilizados são:

```
Lcd_Cmd(_LCD_CLEAR);               // Limpa o display
Lcd_Cmd(_LCD_CURSOR_OFF);          // Desliga o cursor do display
Lcd_Out(1,1," Cronometro");        // Escreve um texto na linha 1, coluna 1
```

Outras funções podem ser encontradas na tabela abaixo.

Lcd Command	Purpose
_LCD_FIRST_ROW	Move o Cursos para Primeira Linha
_LCD_SECOND_ROW	Move cursor to the 2nd row
_LCD_THIRD_ROW	Move cursor to the 3rd row
_LCD_FOURTH_ROW	Move cursor to the 4th row

_LCD_CLEAR	Limpa o Display
_LCD_RETURN_HOME	Return cursor to home position, returns a shifted display to its original position. Display data RAM is unaffected.
_LCD_CURSOR_OFF	Desliga o Cursos
_LCD_UNDERLINE_ON	Underline cursor on
_LCD_BLINK_CURSOR_ON	Blink cursor on
_LCD_MOVE_CURSOR_LEFT	Move cursor left without changing display data RAM
_LCD_MOVE_CURSOR_RIGHT	Move cursor right without changing display data RAM
_LCD_TURN_ON	Turn Lcd display on
_LCD_TURN_OFF	Turn Lcd display off
_LCD_SHIFT_LEFT	Shift display left without changing display data RAM
_LCD_SHIFT_RIGHT	Shift display right without changing display data RAM

Exemplo de código, cronometro no lcd

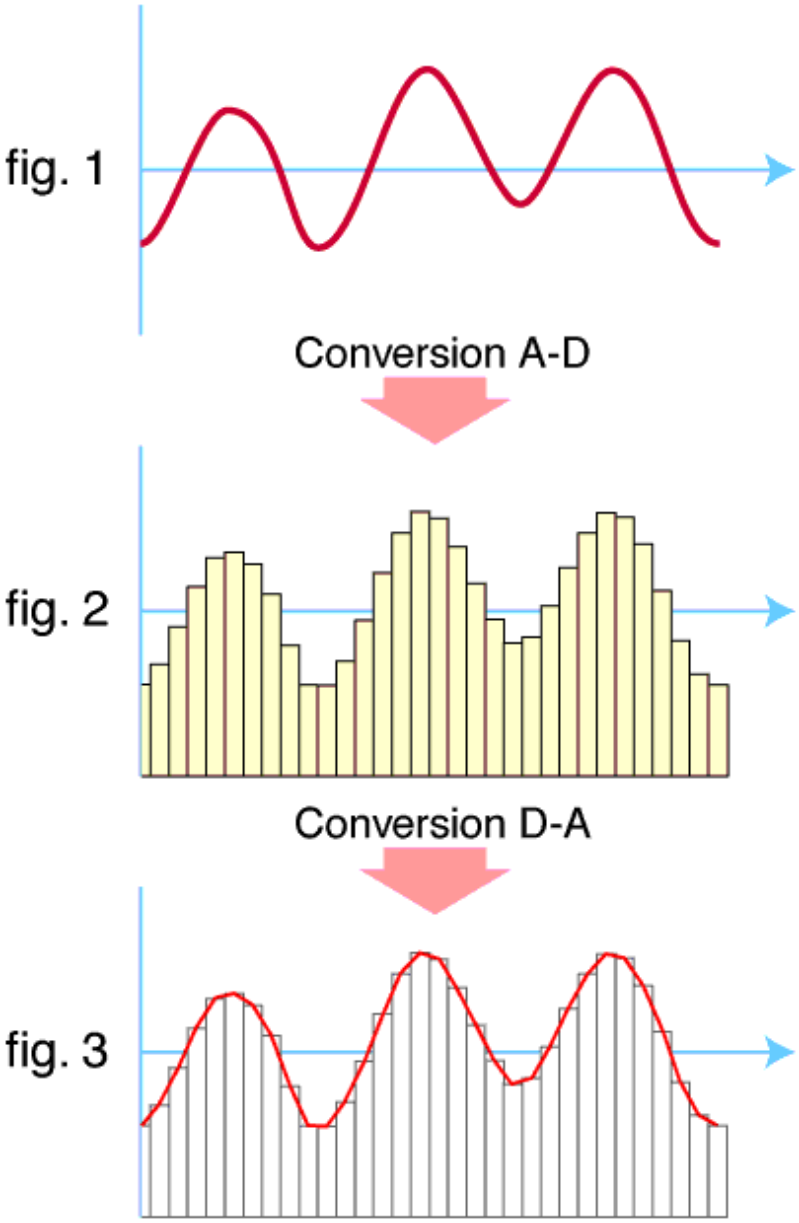
```
// LCD module connections
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connections
```

```
void main(){
  int minuto=0,segundo=0;
  char txt[16];
  Lcd_Init();  // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);               // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);          // Cursor off
  Lcd_Out(1,1," Cronometro");                // Write text in first row
  delay_ms(1000);
  trisb=0;
while(1)
 {
 portb.rb0=~portb.rb0; // teste do período do loop do programa
 wordtostr(minuto,txt); //txt=a
 Lcd_Out(2,1,txt);
 delay_ms(1000); //altera o valor do delay caso deseje calibrar o cronometro
 wordtostr(segundo,txt);
 Lcd_Out(2,6,txt);
 segundo++;
 if(segundo>59) { segundo=0; minuto++; }
  }
}
```

16 Conversor Analógico Digital

Na conversão do sinal analógico em digital pode ser utilizado um Circuito Integrado para fazer a conversão do sinal analógico em um sinal digital, limitado a um número de bits, e um limite inferior e superior de tensão.



Para configuração do conversor analógico digital disponível no PIC16f877A , é necessário o uso de uma tabela de configuração para esse modelo de dispositivo.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	Vss	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	Vss	7/1
0010	D	D	D	A	A	A	A	A	VDD	Vss	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	Vss	4/1
0100	D	D	D	D	A	D	A	A	VDD	Vss	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	Vss	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	Vss	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	Vss	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	Vss	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

Configuração das Portas Analógicas

Adcon1=0b00000000; // todas portas analógicas , com vdd e vss de referência

Adcon1=0b00000110; // todas portas digitais

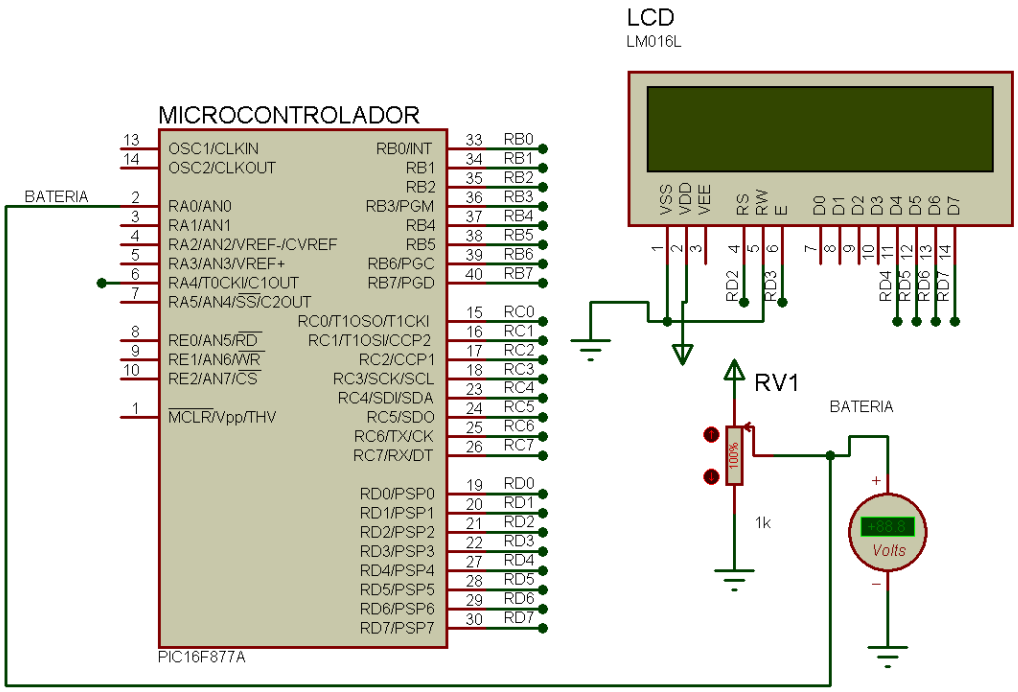
Adcon1=0b00001001 // pinos an7 e an6 digitais, e os demais analógicos VDD e VSS como referência

Variavel=adc_read(1); // leitura do canal ADC no canal AN1.

16.1 Exemplo Prático – Leitura de uma tensão de 0 a 5V.

Para a leitura de uma tensão, primeiro é preciso configurar uma entrada analógica conforme circuito.

O potenciômetro vai variar a tensão de entrada permitindo medir a tensão através do microcontrolador.




```
// LCD module connections
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connections

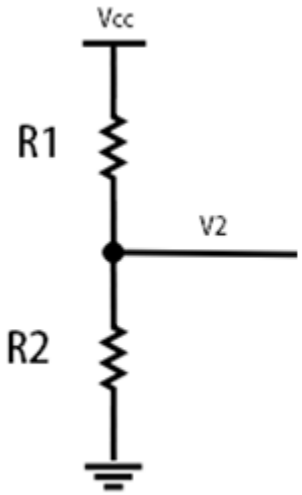
void main(){
  int tensao;    char txt[16];
  Adcon1=0b00000000; // todas as portas analógicas
```

```
  Lcd_Init();  // inicializa LCD
  Lcd_Cmd(_LCD_CLEAR);              // Limpa o display
  Lcd_Cmd(_LCD_CURSOR_OFF);         // Cursor off
  Lcd_Out(1,1,"Tensao Bateria");              // Write text in first row
  trisb=0;
while(1)
{tensao=adc_read(0)*4.888;
 wordtostr(tensao,txt); //txt=a
 Lcd_Out(2,1,txt);
 lcd_out(2,7,"mV");
 delay_ms(200);
 }
                          }
```

17 Divisor de tensão.

Para leitura de tensões acima de 5V, é necessário dividir a tensão para que não ultrapasse o limite do conversor analógico e danifique o dispositivo.

Para isso utilize um divisor de tensão.



Para descobrir o valor de V_2 podemos utilizar a seguinte fórmula:

$$V_2 = \frac{V_{cc}.R_2}{R_{1+}R_2}$$

Cor	1°. Algarismo Significativo	2°. Algarismo Significativo	3°. Algarismo Significativo	Múltiplo	Tolerância
Preto		0	0	x 1	
Marrom	1	1	1	x 10	± 1%
Vermelho	2	2	2	x 10^{2}	± 2%
Laranja	3	3	3	x 10^{3}	
Amarelo	4	4	4	x 10^{4}	
Verde	5	5	5	x 10^{5}	
Azul	6	6	6	x 10^{6}	
Violeta	7	7	7		
Cinza	8	8	8		
Branco	9	9	9		
Ouro				x 10^{-1}	± 5%
Prata				x 10^{-2}	± 10%
Ausência					± 20%

Observe a tabela de resistores, a última faixa é a tolerância (erro máximo) que a resistência pode ter, fazendo que o divisor calculado seja diferente do medido, influenciando diretamente no valor da divisão de tensão, atribuindo um erro na medida da tensão, esse erro pode ser solucionado, utilizando resistores de maior precisão, ou ainda calibrando o equipamento, acertando o valor da equação no software para que seja compatível com o divisor de tensão real.

18 Sensor de Temperatura

Sensor de temperatura em sua maioria são sensores analógicos, podendo variar resistência, corrente com capacitância em função da temperatura.

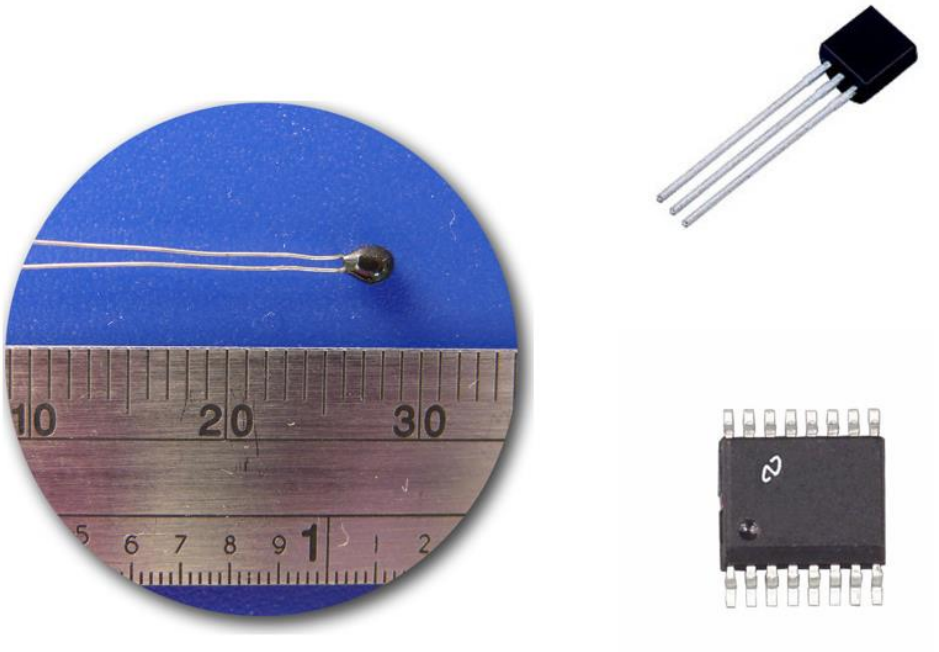


Figura 3 - Sensores de Temperatura

Um dos sensores de temperatura mais comuns são os Termistores, que são semicondutores sensíveis a temperatura, sendo classificados em NTC e PTC.

NTC é um sensor de temperatura com coeficiente negativo, ou seja, sua resistência varia de acordo com a temperatura, ou seja, no caso do NTC; quando a temperatura aumenta sua resistência diminui.

- Os NTC's trabalhão na casa dos -55°C a 300°C
- Possui rápido tempo de resposta.
- Fácil reposição, não necessita de calibração.

O PTC é um sensor de temperatura de coeficiente positivo, funciona ao contrário do NTC, no PTC conforme a temperatura aumenta, maior será sua resistência.

A variação de resistência em sensores PTC e NTC podem ser não lineares, mas em alguns casos dependendo da precisão necessária é possível utilizar uma equação linear aproximada, ou caso necessário encontrar um polinômio que melhor se aproximar da curva obtida.

Nas tabelas temos as tensões e temperaturas medidas sendo que a curva obtida utilizando o software Microsoft Excel foi utilizada para traçar uma reta ou curva de tendência utilizando uma equação linear e um polinômio de grau 02, onde é possível observar o erro em cada situação.

Esses valores medidos não são precisos, mas servem de exemplo para uma aplicação de obtenção de dados de curva de um sensor.

		Aproximação de uma reta		Aproximação polinômio de grau 2	
Tensão V	**Temperatura**	**y=27,871x - 24,006**	**Erro para reta**	**y = 3,61x2 + 5,5523x + 9,3567**	**Erro Polinômio de grau 02**
2,09	36,1	34,24439	-1,85561	36,729848	0,629848
2,2	38,8	37,3102	-1,4898	39,04416	0,24416
2,22	39	37,86762	-1,13238	39,47433	0,47433
2,25	40	38,70375	-1,29625	40,125	0,125
2,29	41	39,81859	-1,18141	41,002668	0,002668
2,34	42	41,21214	-0,78786	42,115998	0,115998
2,38	43	42,32698	-0,67302	43,019658	0,019658
2,42	44	43,44182	-0,55818	43,93487	-0,06513
2,46	45	44,55666	-0,44334	44,861634	-0,138366
2,5	46	45,6715	-0,3285	45,79995	-0,20005
2,54	47	46,78634	-0,21366	46,749818	-0,250182
2,58	48	47,90118	-0,09882	47,711238	-0,288762
2,62	49	49,01602	0,01602	48,68421	-0,31579
2,66	50	50,13086	0,13086	49,668734	-0,331266
2,7	51	51,2457	0,2457	50,66481	-0,33519
2,74	52	52,36054	0,36054	51,672438	-0,327562
2,78	53	53,47538	0,47538	52,691618	-0,308382
2,82	54	54,59022	0,59022	53,72235	-0,27765
2,86	55	55,70506	0,70506	54,764634	-0,235366
2,9	56	56,8199	0,8199	55,81847	-0,18153
2,94	57	57,93474	0,93474	56,883858	-0,116142
2,98	58	59,04958	1,04958	57,960798	-0,039202
3,01	59	59,88571	0,88571	58,776084	-0,223916
3,05	60	61,00055	1,00055	59,87324	-0,12676
3,09	61	62,11539	1,11539	60,981948	-0,018052
3,12	62	62,95152	0,95152	61,82106	-0,17894
3,16	63	64,06636	1,06636	62,949984	-0,050016
3,2	64	65,1812	1,1812	64,09046	0,09046
3,24	65	66,29604	1,29604	65,242488	0,242488
3,27	66	67,13217	1,13217	66,11409	0,11409
3,3	67	67,9683	0,9683	66,99219	-0,00781
3,34	68	69,08314	1,08314	68,173098	0,173098
3,37	69	69,91927	0,91927	69,06636	0,06636
3,41	70	71,03411	1,03411	70,267484	0,267484
3,45	71	72,14895	1,14895	71,48016	0,48016
3,48	72	72,98508	0,98508	72,397248	0,397248

3,5	73	73,5425	0,5425	73,01225	0,01225
3,54	74	74,65734	0,65734	74,250918	0,250918
3,57	75	75,49347	0,49347	75,1875	0,1875
3,6	76	76,3296	0,3296	76,13058	0,13058
3,64	77	77,44444	0,44444	77,398128	0,398128
3,68	78	78,55928	0,55928	78,677228	0,677228
3,71	79	79,39541	0,39541	79,644134	0,644134
3,74	80	80,23154	0,23154	80,617538	0,617538

Tensão V	**Temperatura**	Aproximação de uma reta	**Erro para reta**	Aproximação polinômio de grau 2	**Erro Polinômio de grau 02**
3,77	**81**	81,06767	0,06767	81,59744	0,59744
3,79	**82**	81,62509	-0,37491	82,254318	0,254318
3,81	**83**	82,18251	-0,81749	82,914084	-0,085916
3,84	**84**	83,01864	-0,98136	83,909148	-0,090852
3,86	**85**	83,57606	-1,42394	84,576134	-0,423866
3,89	**86**	84,41219	-1,58781	85,582028	-0,417972
3,93	**87**	85,52703	-1,47297	86,933328	-0,066672
3,95	**88**	86,08445	-1,91555	87,61331	-0,38669
3,97	**89**	86,64187	-2,35813	88,29618	-0,70382
3,99	**90**	87,19929	-2,80071	88,981938	-1,018062

Através da tabela foram obtidos os gráficos de aproximação de reta confirme figura, e curva conforme figura.

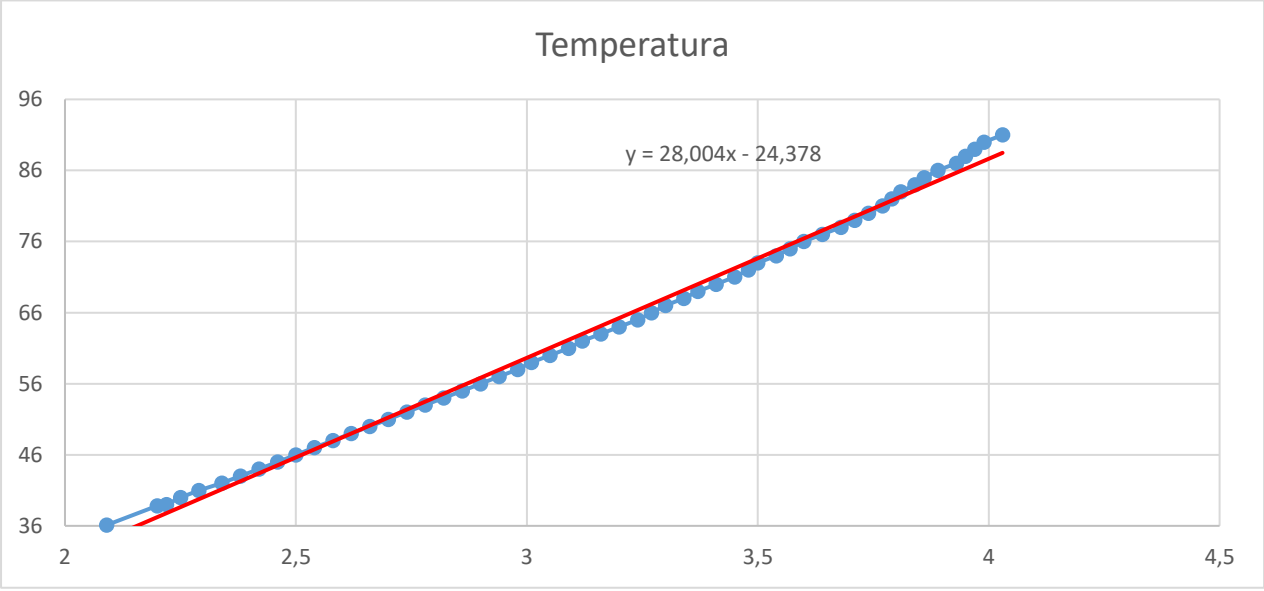


Figura 4 - Gráfico da Temperatura X Tensão, Reta de tendência.

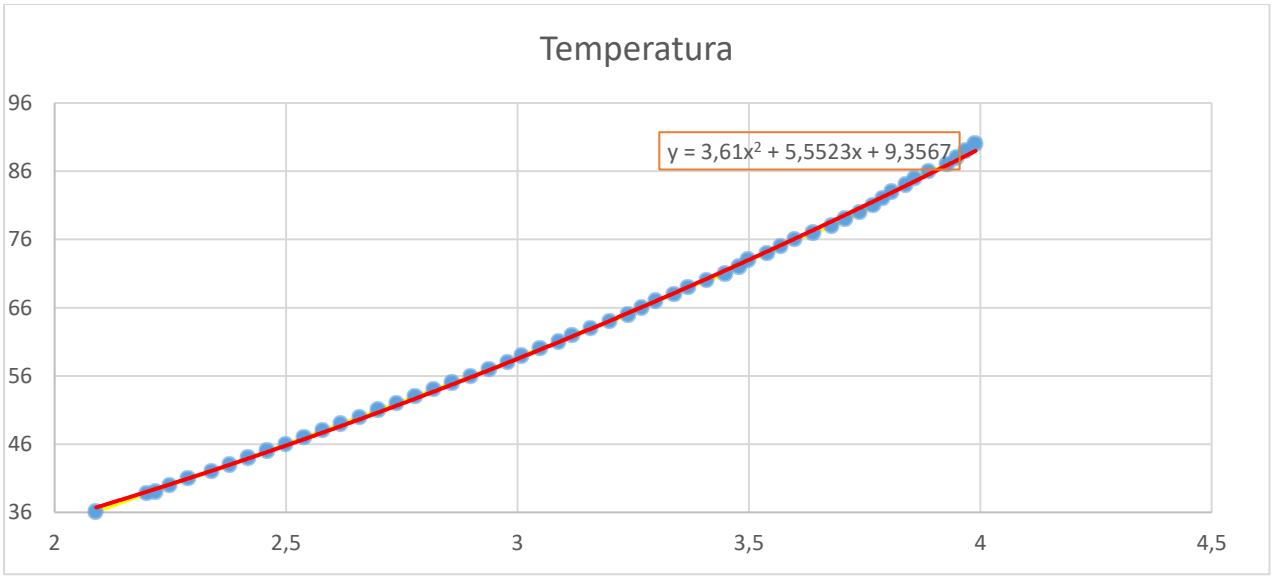


Figura 5 - Gráfico da leitura da temperatura com a aproximação de um polinômio.

19 Exercícios Conversor Analógico Digital

1. Faça a leitura da tensão da bateria do veículo, considere 20V a tensão máxima lida pelo seu circuito e software, apresente a tensão na escala de mV em um LCD, e apresente a interpretação da tensão.
2. (<12v.) Nível Crítico --- Tensão >12v e < 13,5 (carregada, sem estar carregando) – Tensão >13,5 e < 15V (carregando em nível normal) – tensão > 15V, sobrecarga (risco de danificar a bateria)

3. Faça gerador de frequência com variação de 0 a 10hz, controlado por um potenciômetro analógico.

4. Faça gerador de frequência com variação de 0 a 100hz, controlado por um potenciômetro analógico.

5. Faça a leitura de uma bateria de até 15V acenda um led a caba 2 volts.

6. Faça gerador de frequência com variação de 50 a 100hz, controlado por um potenciômetro analógico.

7. Faça a leitura de um sensor de temperatura proporcional a tensão de 0,1V/°C.

8. Faça a leitura de um sensor de posição (potenciômetro) linear, apresenta 4V quando está na posição de 100% e 2V quando está na posição 0.

9. Faça a leitura de um sensor de temperatura proporcional a tensão de 0,1V/°C, e faça leitura de um sensor de posição (potenciômetro) linear, apresenta 4V quando está na posição de 100% e 2V quando está na posição 0. Apresente os dois valores no display de LCD.

10. Mostre a temperatura em ° C utilizando um sensor de temperatura LM35, em um display de 7 segmentos sem conversor BCD.

20 Função

Uma função nada mais é do que uma subrotina usada em um programa.

Na linguagem C, denominamos função a um conjunto de comandos que realiza uma tarefa específica em um módulo dependente de código.

A função é referenciada pelo programa principal através do nome atribuído a ela.

A utilização de funções visa modularizar um programa, o que é muito comum em programação estruturada.

Desta forma podemos dividir um programa em várias partes, no qual cada função realiza uma tarefa bem definida. (1)

A função é composta de 3 partes principais.

20.1 Função sem variável de entrada.

```
void funcaoqualquer(); // protótipo da função , deve estar antes da main()

// variáveis que fazem partes das funções devem ser declaradas como variáveis globais.

// declaradas antes da main().

Int variável;

void main() {

funcaoqualquer();  // chamada da função, chama a função executa e retorna na mesma linha
// ao terminar a execução da função

            }  // fecha main

void funcaoqualquer()

{

 // sequência de comando que a função deve executar.

}
```

20.2 Função com variável de entrada.

```
void piscaled(int p); // protótipo da função

void main() {  // abre a chave main

                trisb=0b00000000;   // configura portb com saída
                portb=0; // coloca todo portb em 0, como valor inicial

                piscaled(3);   // chama a função com o valor 3

             }  // fecha main

void piscaled(int p)     // a variável p foi chamada com o número 3
{  int i;
            for(i=1;i<=p;i++)  // loop de 1 a 3.
            {
             portb.rb0=1;
             delay_ms(500);
             portb.rb0=0;
             delay_ms(500);
             }   // fecha for
}  // fecha função.
```

21 PWM *(Pulse Width Modulation)* ou Modulação de Largura de Pulso.

O PWM é um sinal digital usado para modular a tensão média de uma saída, fazendo isso de forma digital.

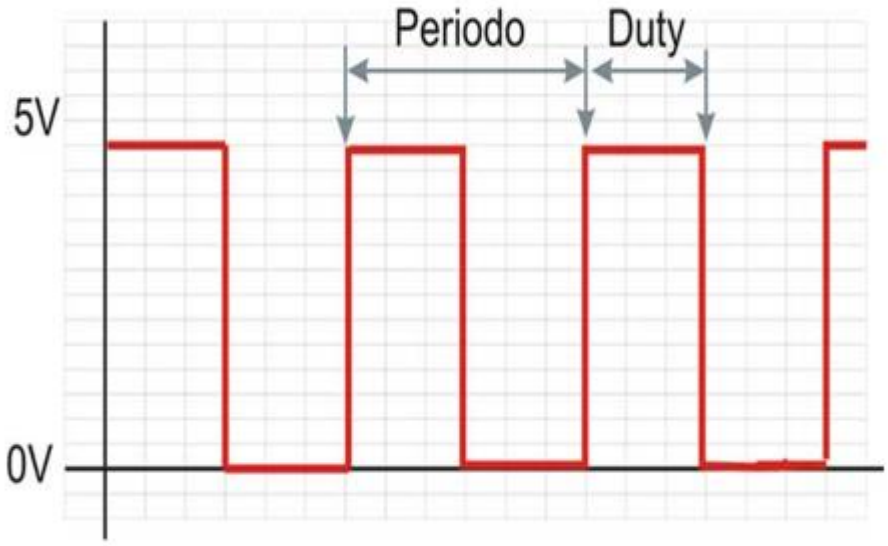
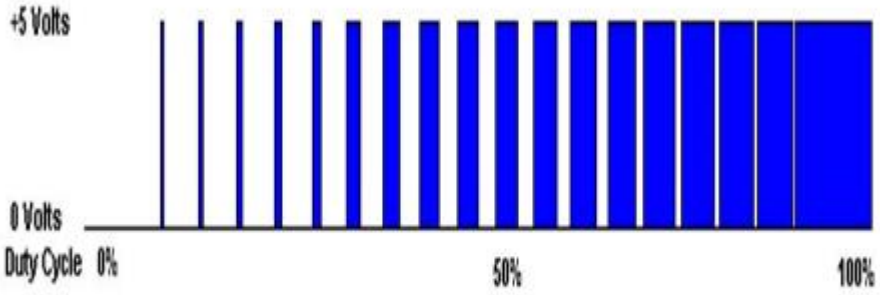
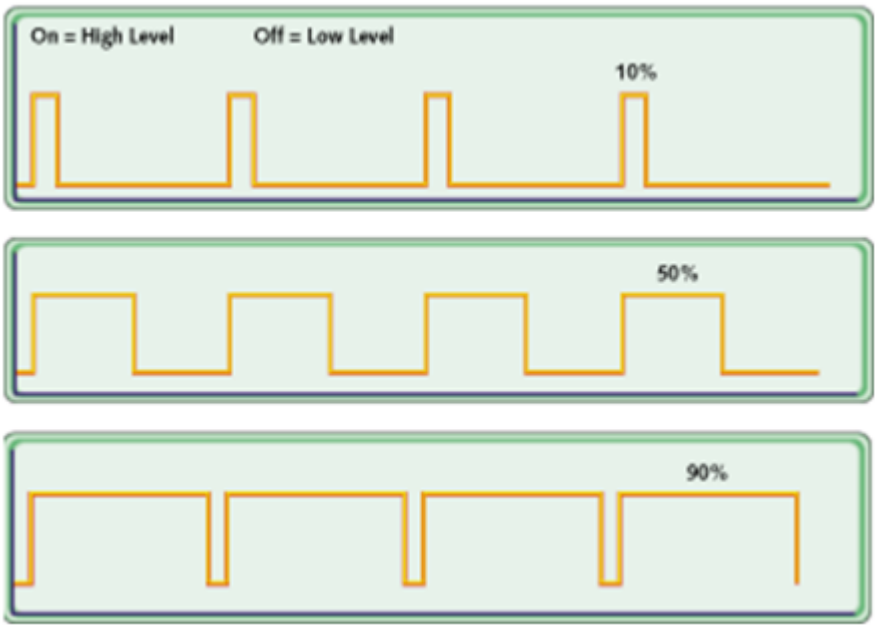


Figura 6 - PWM

O conceito é manter uma frequência fixa, e variar a porcentagem do ciclo alto em relação período total, esse ciclo alto é chamo de duty-cicle. Na Figura 6 - PWM, observa-se o sinal de pwm, com o período fixo e a variação do duty-cicle, que representa a variação da tensão média.





21.1 Exemplo 01 – PWM por Software.

Utilizando o conceito do PWM, vamos gerar um software para gerar um pwm através de um potenciômetro, com uma frequência de 100hz, utilize essa saída para polarizar um transistor e ligue em uma lâmpada incandescente de 12V.

Embora a técnica de gerar um PWM por software possa ocupar muito processamento, não permitindo que o microcontrolador seja utilizado para outras funções, poderá ser melhorada mais adiante quando for apresentado as interrupções por timer.

```
// exemplo 01.
// controle do PWM através de um potênciometro
// Frequencia do PWM  100 hz .
// rotina via software utilizando delay.

void main()
{
int duty,b;
trisc=0;
adcon1=0b00000000;                    // Configure AN pins as digital I/O

while(1)
    {
    duty=adc_read(0)/102.3;
    b=10-duty;
    portc.rc2=1;
    vdelay_ms(duty);
    portc.rc2=0;
    vdelay_ms(b);

    }
}
```

Exercício Proposto.

1. Faça testes com um frequência de 10hz, e perceba o oscilação da lâmpada em alguns valores de PWM.
2. Apresente em um display de LCD o valor do PWM em porcentagem.
3. Apresente em um display de 7 segmentos o valor do PWM em porcentagem.

21.2 PWM por Hardware.

Como vimos anteriormente o sinal de PWM gerado pelo software utiliza muito processamento pois as rotinas de tempo devem ser precisas, não permitindo que o microcontrolador seja utilizado para demais funções.

Para isso muitos microcontroladores possuem um circuito interno gerado de PWM, que após configurado e acionado, mantem o sinal de PWM sem utilizar processamento interno, permitindo o uso do processador para outras funções.

```
Pwm1_Init(5000);          // Inicializa módulo PWM com 5Khz
Pwm1_Start();             // Start PWM
PWM1_Set_Duty(variavel);
```

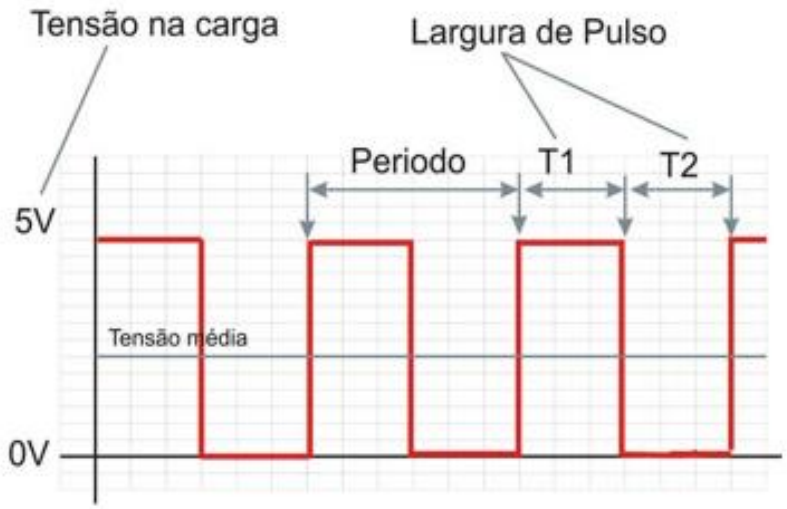


Figura 7 - Sinal PWM

No pic 16f877A, temos dois módulos PWM. Os pinos 16 com o módulo CCP2 ou PWM2, e o pino 17 com o módulo CCP1, PWM1.

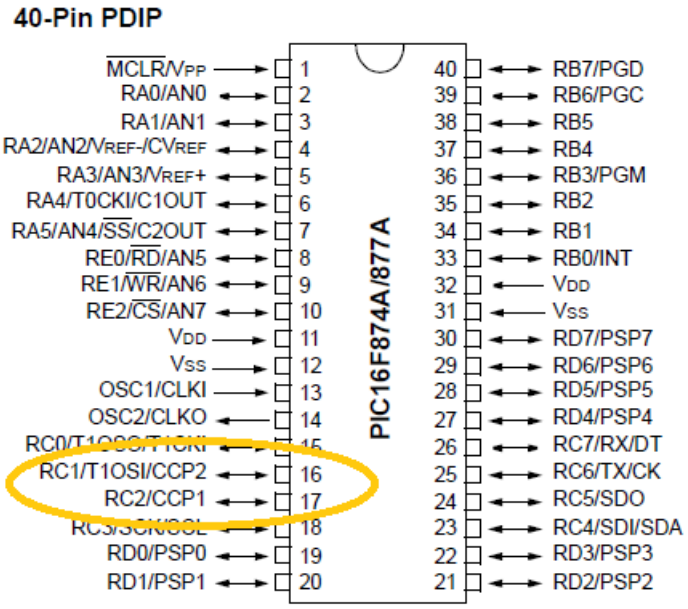


Figura 8 - Pic - módulo PWM.

Pwm1_Init(5000); // Inicializa módulo PWM 1com 5Khz

Pwm2_Init(5000); // Inicializa módulo PWM 2 com 5Khz

Caso utilize os dois módulos PWM, a frequência deve ser as mesma para os dois, embora o compilador aceite o comando com valores diferentes, apenas uma frequência de PWM é estabelecida.

Pwm1_Start(); // Start PWM - inicia o PWM 1

Pwm2_Start(); // Start PWM - inicia o PWM 2

PWM1_Set_Duty(variavel); - 8 ou 10 bits.

PWM2_Set_Duty(variavel); - 8 ou 10 bits.

Utilizando o exemplo Exemplo 01 – PWM por Software., podemos fazer o uso do PWM por Hardware, retirando as funções de acionamento digital e delay, controlando apenas a variável da função PWM1_Set_Duty(variavel);

```
main()
{
int a;
Pwm1_Init(5000);            // Inicializa módulo PWM com 5Khz
Pwm1_Start();               // Start PWM  - inicia o PWM
ADCON1 = 0b00000000; //configuração das portas analógicas.
while(1)
{
a=adc_read(1)/4;   // a divisão por 4 é necessária para transformar a variável do AD de 10 bits
                   //em 8 bits.
```

```
pwm1_set_duty(a); // utiliza a variável a como o duty cicle do PWM, variando de 0 255
 }
}
```

22 Bibliografia

1. **[Online] http://linguagemc.com.br/funcoes-em-c/.**

2. Bosch. [Online] http://br.bosch-automotive.com/pt/internet/parts/parts_and_accessories_2/motor_and_sytems/benzin/more_sensors/sistema_egas_pedal_acelerador_eletronico/sistema_egas__pedal_acelerador_eletronico.html.

3. Tomson, Curso On line MTE. [Online] http://www.cursosonlinemte.com.br/licao/aula-13-valvula-injetora/.

4. Leidecker, Henning. Pasahchenko Lyudmyla. Brusse Jay. Eletrical Failufe of an Accelerator Pedal Position Sensor Caused by a Tin Whisker and Discussion of Investigative Tecchiniques Used for Whisker Detection . ***5th Internacional Tin Whisker Symposium .*** **2011.**

5. DANTAS, SEBASTIÃO JOSE VIEIRA. Notas de Aula de Sensores e Atuadores . 2017.