Centro Universitário *UNIVATES*

Engenharia de Software

Diciplina Arquitetura e Organização de Computadores

Projeto do
Módulo chaveador I/O 8

Plano de Projeto - versão 1.1

Equipe:

Tales Igor Ebert

Professor: Pedro Antonio Madeira de Campos Velho

Índice

Histórico de Alterações

Data	Versão	Descrição	Autor
16/05/14	1.0	Construção do Documento	Tales
30/06/14	1.1	Ajustes do Documento	Tales

1 Introdução

1.1 Motivações e Objetivo

Estamos na era da automação, e estes processos exigem ser comandados de alguma forma, seja por hardware e/ou software, devido a esta característica, microcontroladores com software embarcado estão presentes em várias soluções, principalmente na industria.

O propósito do trabalho é criar um protótipo (interface) para comunicação entre um computador capaz de executar um software que envie comandos para acionar um dispositivo externo para entender como estes são fabricados e qual a complexidade de operação.

1.2 Descrição do projeto Módulo chaveador I/O 8

Este projeto tem como objetivo produzir uma interfaces que receba 8 bits de sinal 5v e 0v de um processador capaz de executar um softwares e tenha 8 saídas para acionamento de diferentes equipamentos/componentes externos.

1.3 Restrições

1.3.1 Restrição de Tempo

O sistema deverá ser entregue até 04/07/2014.

1.3.2 Recursos

Serão alocados um notebook com sistema operacional Linux, componentes eletrônicos necessários para o projeto, internet, um computador com porta serial.

2 Gerenciamento do trabalho

2.1 Declaração de Escopo do Trabalho

2.1.1 Justificativa

Com a crescente industria de automação e mercado de softwares embarcados, vemos a necessidade de termos um contato como montagem de circuitos em *protoboard* para simulações, antedimento dos componentes eletrônicos utilizados e a lógica por traz do circuito eletrônico, e entender como os softwares embarcados funcionam rodando sobre diferentes processadores.

2.1.2 Organização do trabalho

O trabalho será dividido em duas partes, parte um, contendo a explicação, conceitos de hardware e parte dois, contendo a explicação do software;

3 Trabalho

3.1 Hardware

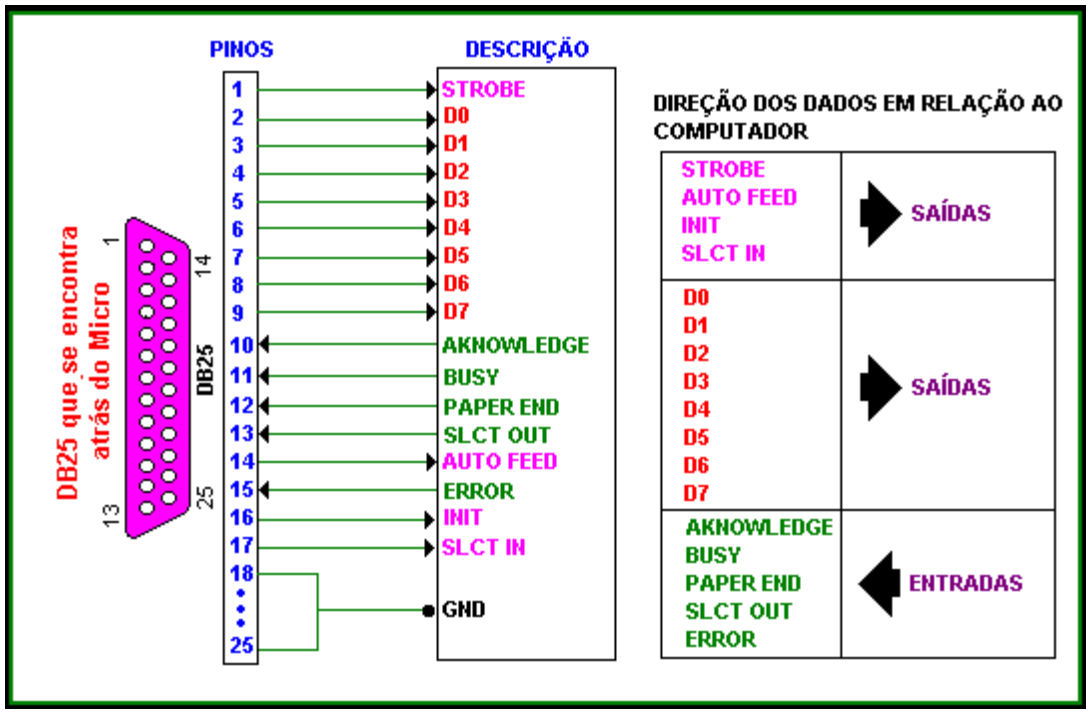
3.1.1 Porta paralela

A porta paralela é uma interface de comunicação (entrada e saída) entre o computador e um periférico, originalmente criada para impressoras, mas atualmente, são vários os periféricos que utilizam-se desta porta para enviar e receber dados para o computador.

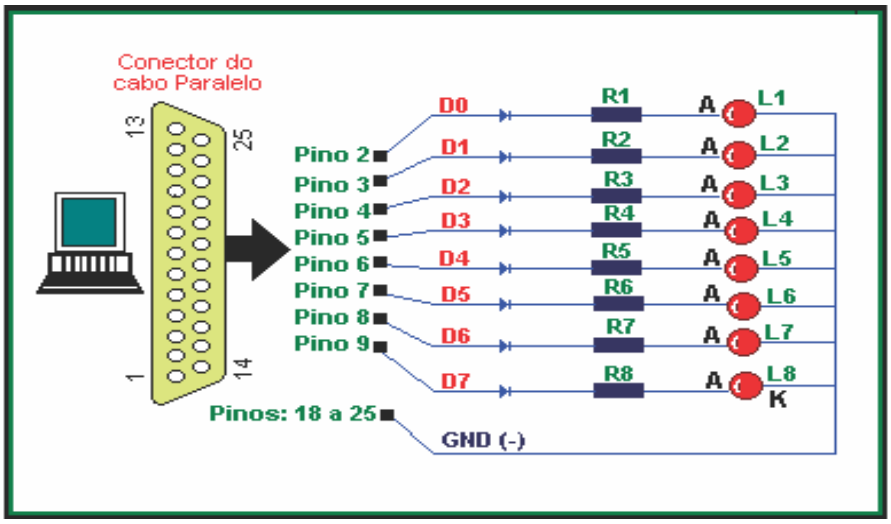
A porta paralela por trabalhar em transmissão bidirecional, modo EPP (Enhanced Parallel Port) e comunica-se com a CPU utilizando um BUS de dados de 32 bits e para a transmissão de dados entre periféricos são usado 8 bits por vez.

O DB25 é um conector que fica na parte de trás do gabinete do computador, e é através deste, que o cabo paralelo se conecta ao computador para poder enviar e receber dados.

Na porta paralela, um pino está em nível lógico 0 quando a tensão elétrica no mesmo está entre 0 à 0,4v. Um pino se encontra em nível lógico 1 quando a tensão elétrica no mesmo está acima de 3.1 e até 5v.



3.1.2 Esquema para ligação de simples LEDs:

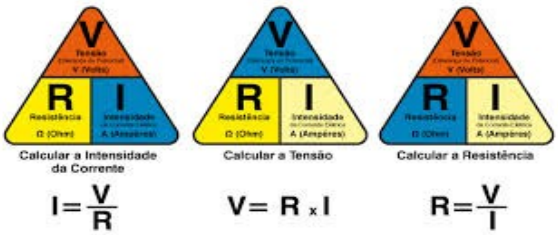


Calculando o resistor para resistir a passagem de tensão.

Tabela de cores: tabela_de_cores_calculadora

Calculadora de resistor: calculadora_online

Lei de OHM: lei_de_ohm



Saída 5V – 1.5v (voltagem de operação do led) = 3.5v, aplicando a lei de ohm temos:

R= v/i (R= 3,5v/5ma R= 3,5/0,005A) R= **700 ohm**

OBS: utilizei então um resistor de 1K (1000ohm) que resiste ainda mais a voltagem.

3.1.3 Display de 7-segmentos

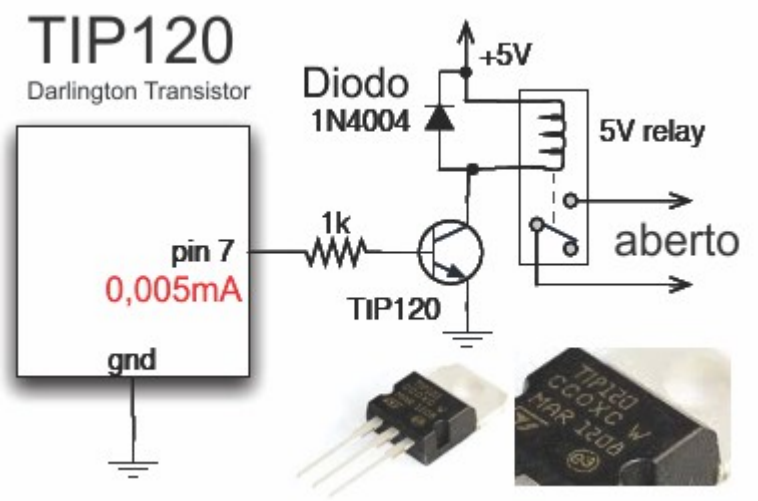
Um display de 7 segmentos podem ser considerados como 7 leds agrupados de forma ordenada, para que, quando aplicada tensão em algum determinado terminal, acenda seu respectivo LED. Se uma certa combinação de LEDs for acesa simultaneamente, é possível exibir algarismos numéricos sendo mostrados pelo display.

Logo após a montagem do circuito, isolamos o display e buscamos compreender quais eram os pinos do display responsáveis pelo acionamento de cada segmento, seguem as figuras representando os resultados desse procedimento.

			POSIÇÃO								
			8	7	6	5	4	3	2	1	
DISPLAY	BIN	DEC	g	f	a	b	e	d	c	h	
0	01111110	126	0	1	1	1	1	1	1	0	0
1	00010010	18	0	0	0	1	0	0	1	0	1
2	10111100	188	1	0	1	1	1	1	0	0	2
3	10110110	182	1	0	1	1	0	1	1	0	3
4	11010010	210	1	1	0	1	0	0	1	0	4
5	11100110	230	1	1	1	0	0	1	1	0	5
6	11001110	206	1	1	0	0	1	1	1	0	6
7	00110010	50	0	0	1	1	0	0	1	0	7
8	11111110	254	1	1	1	1	1	1	1	0	8
9	11110010	242	1	1	1	1	0	0	1	0	9

É importante dizer que esta tabela corresponde ao nosso circuito (pinagem que foi usada na ligação do display x porta-paralela).

3.1.4 Darlington Transistor



Na eletrônica, o transistor Darlington é um dispositivo semicondutor que combina dois transístores bipolares no mesmo encapsulamento (as vezes chamado par Darlington).

A configuração (originalmente realizada com dois transistores separados) foi inventada pelo engenheiro Sidney Darlington do Bell Labs. A ideia de por dois ou três transistores em um mesmo chip foi patentada por ele, mas não a ideia de por um número arbitrário de transistores, o que originaria o conceito moderno de circuitos integrados.

Esta configuração serve para que o dispositivo seja capaz de proporcionar um grande ganho de corrente e requer menos espaço do que o dos transistores normais na mesma configuração. O Ganho total do Darlington é produto do ganho dos transistores individuais. Um dispositivo típico tem um ganho de corrente de 1000 ou superior.

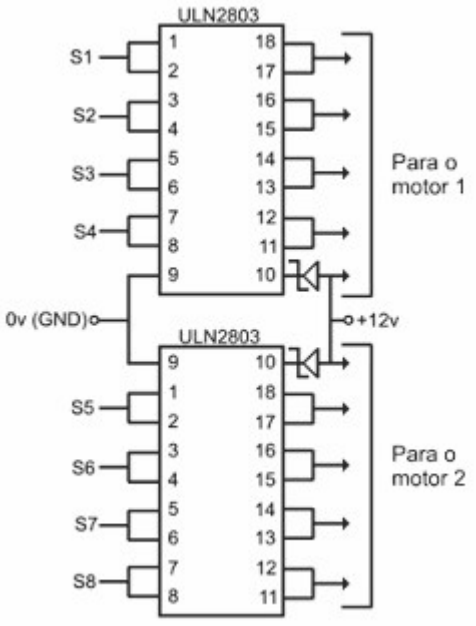
Darlington Transistor: video_externo e artigo

3.1.5 CI ULN2803

O ULN2803 é um CI que contém um conjunto de oito transistores darlington e funciona como um amplificador de sinais, tem 8 entradas que podem controlar até 8 saídas

As entradas são ativadas com 5v e baixa corrente, mas as saídas podem ter até 50v. Além disso, ele eleva um consumo de corrente de até 500mA. Existe também o ULN2003, que tem as mesmas características, com a diferença que tem 7 entradas e 7 saídas.

Ainda segundo o datasheet do ULN2803, é possível ligar as entradas e saídas em paralelo para aumentar o consumo máximo de corrente suportado pelo chip. Então ligando as entradas e saídas, de duas a duas, conseguimos uma capacidade máxima de 1A.



3.1.6 Motor de Passo

Motores de passos são dispositivos mecânicos eletro-magnéticos que podem ser controlados digitalmente através de um hardware específico e/ou através de softwares. Motores de passos são encontrados em aparelhos onde a precisão é um fator muito importante. São usados em larga escala em impressoras, plotters, scanners, drivers de disquetes, discos rígidos e muitos outros aparelhos. Existem vários modelos de motores de passos disponíveis no mercado que podem ser utilizados para diversos propósitos. Poderemos utilizá-los para mover robôs, câmeras de vídeo, brinquedos ou mesmo uma cortina.

Abaixo segue a tabela como exemplo das diferentes configurações de passos.

Passo completo 1 (Full-step)

- Somente uma bobina é energizada a cada passo;
- Menor torque;
- Pouco consumo de energia;
- Maior velocidade.

Nº do passo	B3	B2	B1	B0	Decimal
1-->	1	0	0	0	8
2-->	0	1	0	0	4
3-->	0	0	1	0	2
4-->	0	0	0	1	1

Passo completo 2 (Full-step)

- Duas bobinas são energizadas a cada passo;

-Maior torque;

-Consome mais energia que o Passo completo 1;

-Maior velocidade.

Nº do passo	B3	B2	B1	B0	Decimal
1-->	1	1	0	0	12
2-->	0	1	1	0	6
3-->	0	0	1	1	3
4-->	1	0	0	1	9

Meio passo (Half-step)

- A combinação do passo completo 1 e do passo completo 2

-Consome mais energia que os passo anteriores;

-É muito mais preciso que os passos anteriores;

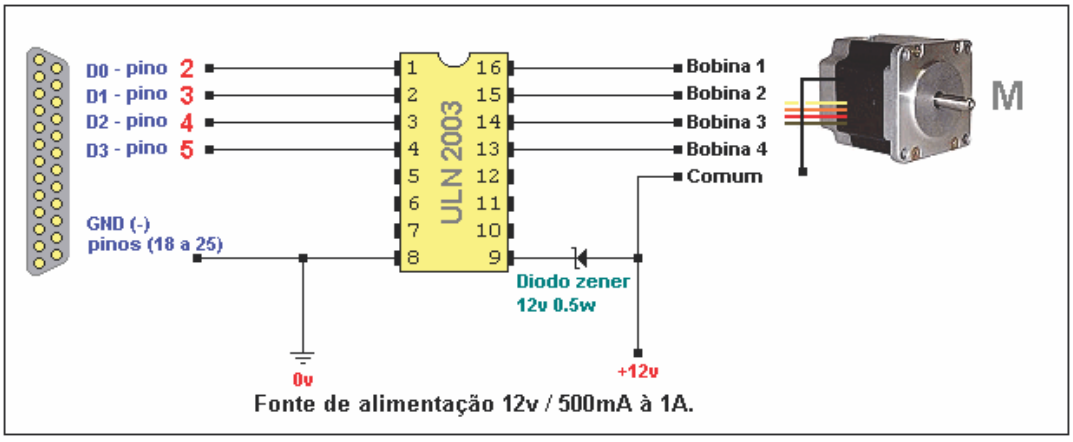
-O torque é próximo ao do Passo completo 2;

-A velocidade é menor que as dos passos anteriores.

Nº do passo	B3	B2	B1	B0	Decimal
1-->	1	0	0	0	8
2-->	1	1	0	0	12
3-->	0	1	0	0	4
4-->	0	1	1	0	6
5-->	0	0	1	0	2
6-->	0	0	1	1	3
7-->	0	0	0	1	1
8-->	1	0	0	1	9

3.1.7 Esquema para ligação de um ULN2803 a um motor de passo

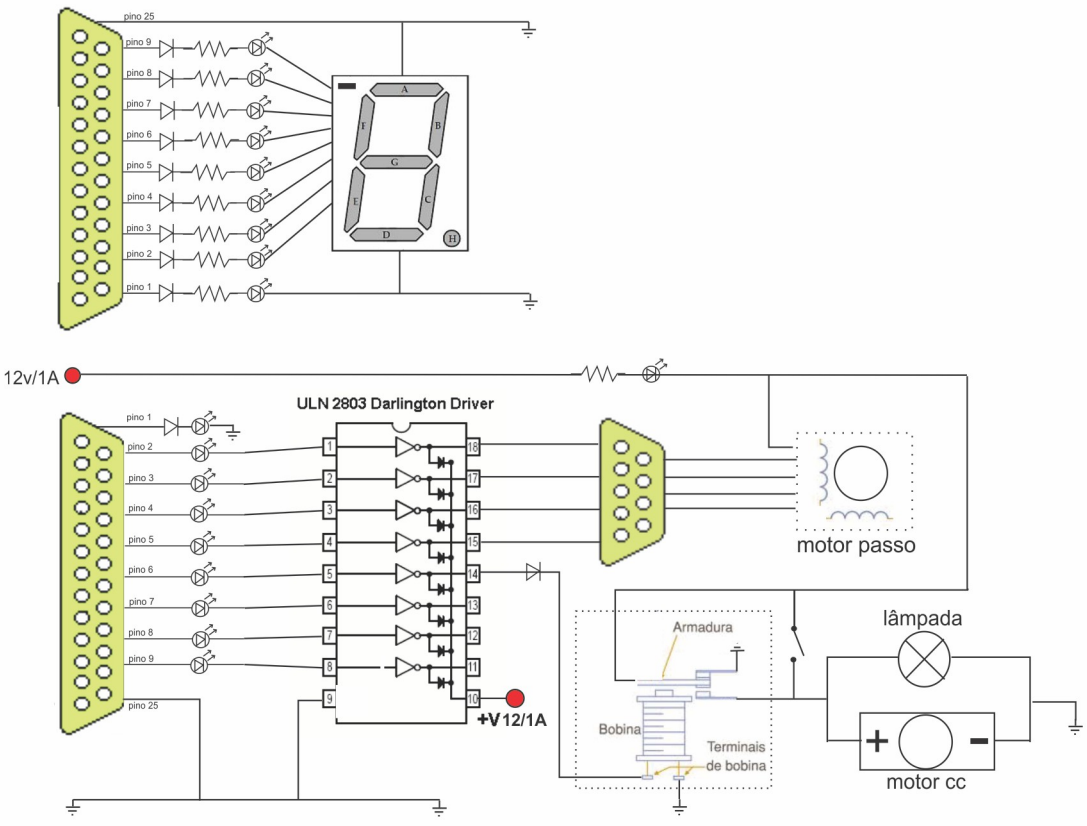
Nesta esquema as entradas de um ULN 2803 são alimentadas de forma a atender a lógica digital TTL, recebendo diretamente de um computador capaz de processar um software especifico para nossa aplicação. Cada enrolamento do motor de passo é energizado quando a entrada corresponde é levada ao nível alto. Assim, no software que será utilizado deve-se programar tanto o tempo de acionamento de cada saída quanto sua sequência, ou ainda ordem para a aplicação desejada. Observe que há uma alimentação independente de 12 V para o motor de passo, que operará com uma corrente máxima de 500 mA.



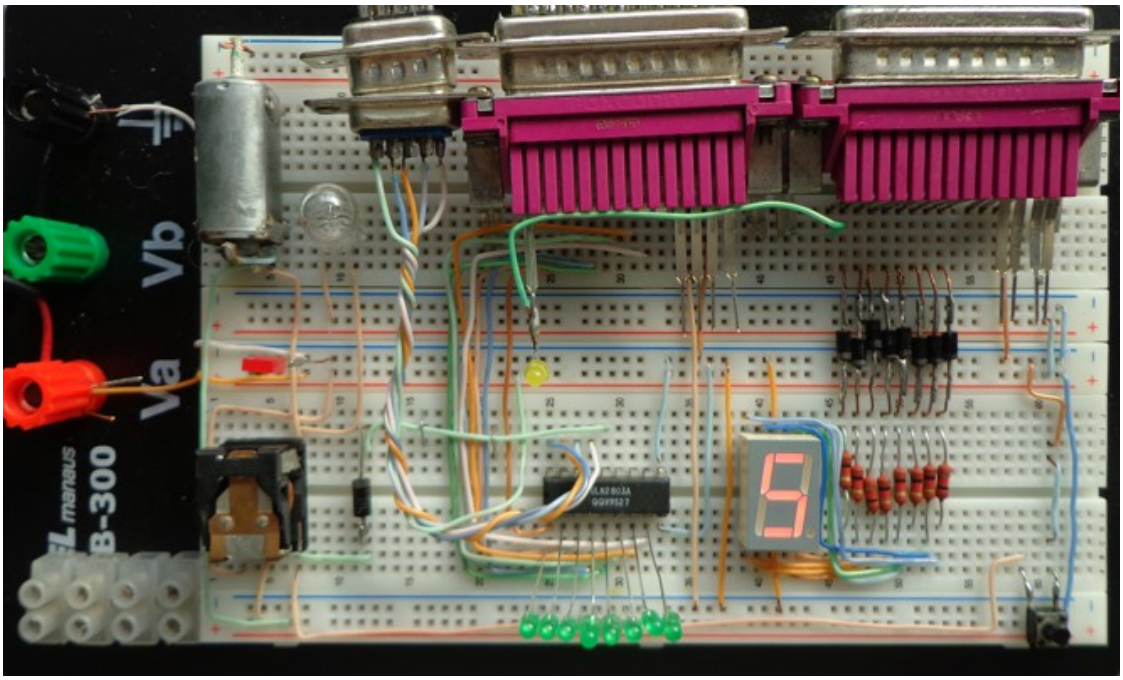
Os drivers comerciais, como ULN2003, ULN2803, UDN6118 e BA618 entre outros, tem a característica de servirem como protetores, pois eles tem em seu integrado também um diodo que permite a passagem de corrente somente em um sentido.

3.1.8 Esquema técnico usado na protoboard

Foi utilizado uma fonte de 12V externa para alimentação do motor de passo, motor contínuo e lâmpada encandeceste. O driver ULN 2803 é alimentado por um conjunto de 8bits (lógica TTL) de entrada que aciona um relé no pino 5 e um motor de passo com os pinos 1,2,3 e 4, estes que vão energizar as bobinas do motor na ordem programada.



Representação do circuito montado sobre a protoboard.

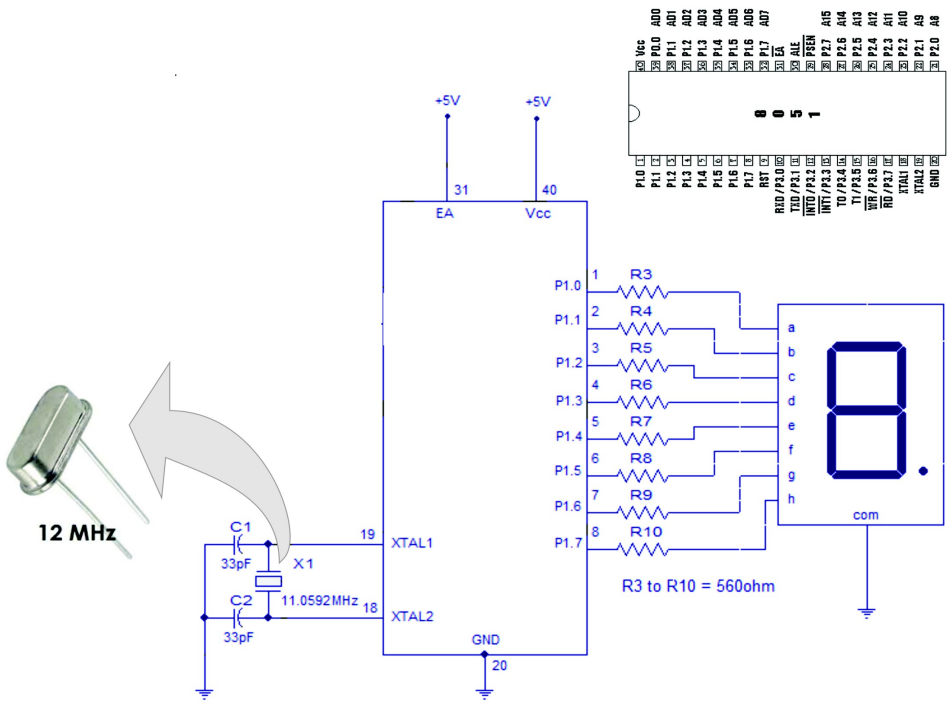


Circuito final montado sobre a protoboard.

3.1.9 Como seria fazer o circuito do display com o 8051?

A ligação do 8051 é simples, como não tivemos acesso a uma gravadora de 8051, não foi possível fazer a montgem em uma protoboard, porem a ligação do 8051 seria dos seguintes pinos:

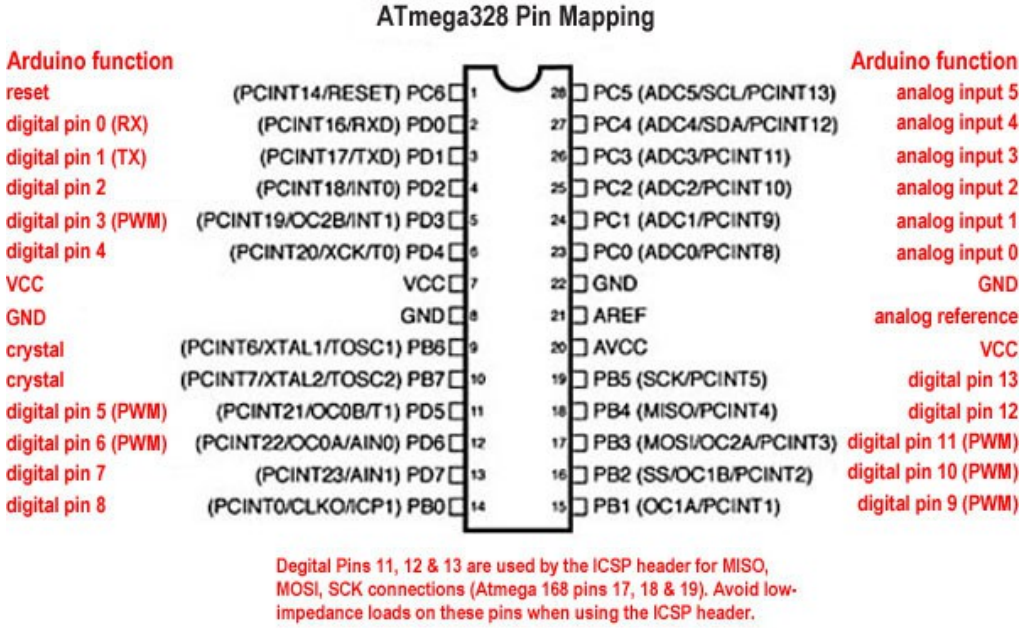
- Pino EA: É um sinal de entrada, para escolher se será utilizada a memória ROM interna;
- Pinos P1.0 ao P1.7: São os pinos de saída conetados diretamente ao nosso display;
- Pino GND que será conectado ao terra/negativo;
- Pino Xtal1 e Xtal2: são conectados ao pequeno circuito oscilador, usando dois capacitores e um cristal.



3.1.10 Como seria fazer o circuito do display com o Arduino?

Os apaixonados por tecnologia certamente já pensaram em prover soluções eletrônicas que resolvessem probleminhas do dia a dia. Com o Arduino, uma placa fabricada na Itália utilizada como plataforma de prototipagem eletrônica que torna a robótica mais acessível a todos. Projeto italiano iniciado em 2005 tinha primeiramente cunho educacional e interagia com aplicações escolares. A fonte de alimentação recebe energia externa por uma tensão de, no mínimo, 7 volts e máximo de 35 volts com corrente mínima de 300m. Em termos de software, o Arduino pode ter funcionalidades

desenvolvidas por meio da linguagem C/C++. As funções IDE do Arduino permitem o desenvolvimento de software que possa ser executado pelo dispositivo.



Circuito para usar o ATMega328 sem a placa Arduino: link_externo

3 Sotfware

3.2.1 Acesso a porta paralela em alto-nível

Escrever programas para acessar a porta paralela foi muito fácil no tempo do DOS e em Win95/98. Nós poderíamos usar Inporb e outportb ou _inp () ou funções _outp em nosso programa sem qualquer problema. Mas entrar para a nova era de sistemas operacionais como: Win NT4, WIN2000, WINXP, toda essa simplicidade acabou. Quando estamos tentando executar um programa que é escrito usando as funções do software convencionais como Inporb, outportb, _inp () ou _outp em um sistema WINNT

ou WIN2000 ou superior, ele irá mostrar uma mensagem de erro similar a "A instrução privilegiada exceção ocorreu na aplicação na localização .... ".

A solução foi introduzir a biblioteca ***inpout32.dll*** para WIN 98/NT/2000/XP. Esta dll tem as seguintes características:

- Usa um driver de modo kernel que não gera erro de execução privilegiada;
- Nenhum software especial ou instalação de driver é necessário;
- Não usa APIs especiais necessárias, usa apenas duas funções Inp32 e Out32;
- Pode ser facilmente usado com VC + + e VB;

Sobre a DLL: link_externo

Click here *for a program written for Borland C++ compiler, by Douglas Beattie Jr.*

3.2.2 Programando em C#

O acesso a porta usando a DLL inpout32.dll é extremamente simples, logo que entendido seu funcionamento, segue abaixo trechos do código fonte do aplicativo produzido para este trabalho.

Abaixo a forma como criamos uma classe para instanciar a DLL, no caso ela precisa estar dentro do diretorio do programa ou dentro do diretorio padrão de bicliotecas do ambiente de programação, isto varia conforme a linguagem que escolher.

```
namespace ModuloChaveadorIO8{
  class PortControl{
    [DllImport("Inpout32.dll")]
    public static extern short Inp32(int address);
    [DllImport("inpout32.dll", EntryPoint = "Out32")]
    public static extern void Output(int adress, int value); // valores em decimal
```

```
    }
}
try {
            PortControl.Output(decAdd, decData);
    } catch(System.DllNotFoundException) {
        MessageBox.Show("A inpout32.dll não existe!!!","Erro da DLL");
       System.Environment.Exit(-1);
    }
```

Variáveis para dado_decimal e endereco_decimal são usadas para chamar a função Output da DLL que recebe dois valores decimais como parâmetro:

```
int dado_decimal = 0;
int endereco_decimal = 888; // one 888 = 378h endereço físico da porta paralela
```

Para exibirmos um numero no display, é necessário enviar um conjunto de 8bits(1Byte), onde um bit sinalizado como 0, corresponde a 0V a 0,4V e quando sinalizado como 1, corresponde de 3,3V a 5V.

```
 private void button9_Click(object sender, EventArgs e)
      {
        this.numero(7);//desligado, onde 0 = 00000000
        System.Threading.Thread.Sleep(200);
        this.numero(10);//parâmetro 10 para decimal 0
      }
public void numero(int num) {
        if (num == 10)
        {
           dado_decimal = 0;//desligado, onde 0 = 00000000
           PortControl.Output(endereco_decimal, dado_decimal);
        }
       if (num == 7)
```

```
        {
           dado_decimal = 0;//desliga, onde 0 = 00000000
           PortControl.Output(endereco_decimal, dado_decimal);
           dado_decimal += 50;//ligando agrupados, onde 50 = 00110010
           PortControl.Output(endereco_decimal, dado_decimal);
        }

if (tboxD4.BackColor == Color.Red)
  private void btnD4_Click(object sender, EventArgs e) {
        getAdd();  // pega o valor de endereço da porta-paralela na combolist

        if (tboxD4.BackColor == Color.Red) {
           dado_decimal += 16;
           tboxD4.BackColor = Color.Green;
        } else {
           dado_decimal -= 16;
           tboxD4.BackColor = Color.Red;
        }
        PortControl.Output(endereco_decimal, dado_decimal);
        converterValor();  // função para exibir o valor em bit/hex
     }
```

Anexo: codigo completo em cssharp desenvolvido para este trabalho: link_externo

Tamanho do programa: 40kbytes;

3.2.3 Programando em DSL para Arduino

O arduino usa uma linguagem de programação DSL (Linguagem de programação específica) baseada em C/C++. É uma plataforma de desenvolvimento onde o desenvolvedor sente-se a vontade em programar. Abaixo o link externo da espeficicação

da linguagem:

Espeficicação de referencia da Linguagem: link_externo

Programa usado para controlar a protoboard:

```
#include <Stepper.h>
int passos = 50;
int estado_bootaoPino13 = LOW; //pino usado para ligar o botão
int contador = 0;
const int passos_por_volta = 50;
Stepper motor_passo(passos_por_volta,9,10,11,12);
int velocidade_motor = 50;

void setHalfStep();
const int sinalPino0 =  0;
const int sinalPino1 =  1;
const int sinalPino2 =  2;
const int sinalPino3 =  3;
const int sinalPino4 =  4;
const int sinalPino5 =  5;
const int sinalPino6 =  6;
const int sinalPino7 =  7;
const int sinalPino8 =  8;
const int sinalPino9 =  9;
const int sinalPino10 = 10;
const int sinalPino11 = 11;
const int sinalPino12 = 12;
const int botaoPino13 = 13;

void setup(){
        pinMode(sinalPino0,  OUTPUT); // declara o pino do led como saída
```

```
        pinMode(sinalPino1,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino2,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino3,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino4,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino5,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino6,  OUTPUT); // botdeclara o pino do led como saída
        pinMode(sinalPino7,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino8,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino9,  OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino10, OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino11, OUTPUT); // declara o pino do led como saída
        pinMode(sinalPino12, OUTPUT); // declara o pino do led como saída
        pinMode(botaoPino13, INPUT);
        digitalWrite(botaoPino13, HIGH);
        motor_passo.setSpeed(velocidade_motor); //Determina a velocidade do motor
}
void loop(){
        verifica_botao();
        delay(300);
        digitalWrite(sinalPino8, LOW);
        PORTD=0b01111110; // 0
        verifica_botao();
        delay(1000);
        PORTD=0b00010010; // 1
        verifica_botao();
        delay(1000);
        PORTD=0b10111100; // 2
        verifica_botao();
        delay(1000);
        PORTD=0b10110110; // 3
        verifica_botao();
```

```
            delay(1000);
            PORTD=0b11010010; // 4
            verifica_botao();
            delay(1000);
            PORTD=0b11100110; // 5
            verifica_botao();
            delay(1000);
            PORTD=0b11101110; // 6
            verifica_botao();
            delay(1000);
            PORTD=0b00110010; // 7
            verifica_botao();
            delay(1000);
            PORTD=0b11111110; // 8
            verifica_botao();
            delay(1000);
            PORTD=0b11110010; // 9
            verifica_botao();
            delay(1000);
            digitalWrite(sinalPino8, HIGH);
}

void loop_motor(){
            motor_passo.step(passos);
}

void verifica_botao(){
          estado_bootaoPino13 = digitalRead(botaoPino13);
          while(estado_bootaoPino13 == LOW) {
                        motor_passo.setSpeed(velocidade_motor);
                        loop_motor();
```

```
                velocidade_motor = velocidade_motor-20;
                estado_bootaoPino13 = digitalRead(botaoPino13);
                if (velocidade_motor < 40){
                velocidade_motor = 100;
        }
        digitalWrite(sinalPino9, LOW);
        digitalWrite(sinalPino10, LOW);
        digitalWrite(sinalPino11, LOW);
        digitalWrite(sinalPino12, LOW);
 }
}
//PORTD - The Port D Data Register – read/write
//digitalWrite(pin, value) returns null
//Stepper Library
//Stepper(steps, pin1, pin2)
//Stepper(steps, pin1, pin2, pin3, pin4)
//setSpeed(rpm) //step(steps)
```

Anexo: codigo completo em DSL Arduino desenvolvido para este trabalho : link_externo

Tamanho do programa: 2,5Kbytes;

4 Vídeo final do projeto rodando

link externo em http://tirs.com.br/universidade/arquitetura/ArOrgComp-trabalho-final_001.php

5 Conclusão

Este trabalho teve como enfoque o estudo, pesquisa e desenvolvimento de sistemas para controle de componentes externos. Foram explanados todos os passos necessários para a realização de desenvolvimento e/ou integração de sistemas embarcados, conhecimento de eletronica e ferramentas e/ou as IDE em que trabalhamos.

Com relação ao nosso protótipo, consluiu-se que podemos desenvolver soluções em automação usando hardwares e software de alto nível, e o objetivo do trabalho foi atingido ao controlar a interfaces desenvolvida com uma aplicação de alto-nivel rodando em um computador doméstico.

Ao longo do desenvolvimento e pesquisa, foi possível concluir que é muito mais vantajoso usar um hardware que suporte programação embarcada pelos seguintes motivos:
- Embora baixo nível, é fácil seu entendimento e sua programação, pois é específica para o hardware/processador;
- Não existem fatores externos que comprometam o funcionamento, é livre de originados por um sistema operacional como linux/windows, drivers etc;
- O processador/hardware não se modifica, ele sempre irá funcionar da mesma forma;
- Dependendo da finalidade de um produto final, com programação embarcada e processadores de baixo custo, podemos contemplar tudo em único hardware;

Em relação ao aplicativo inicialmente desenvolvido, podemos concluir que ao utilizar em um computador que usa Windows ou Linux, temos muitas outras preucupaçãoes que vão alem do aplicativo em si, tais como:
- Preucupação com compatibilidade entre diferentes versões de hardware;

- O lançamento de uma nova versão de sistema operacional pode fazer com que nosso aplicativo pare de funcionar;
- É necessário ser compatível com diferentes portas de comunicação;
- O projeto pode ou não ter que se preucupar com drivers e compatibildade;

Já quando trabalhamos com um codigo embarcado em um processador "simples", é muito mais fácil adquirir conhecimento total sobre o funcionamento do hardware e software, desta forma ter total confiança se for usado em um produto final.

Em relação ao conteudo da disciplina de arquitetura, este foi fundamental para os principios de funciomaneto de um processar com linguagens de baixo-nível; e somado ao interesse na pesquisa para a execução do trabalho, tivemos um resultado satisfatório.

6 Bibliografia

6.1 Paginas Eletrônicas

©2014 Wikipédia. "Display de sete segmentos", http://pt.wikipedia.org/wiki/Display_de_sete_segmentos [visitada em 11 de maio de 20014]

©DataSheet datasheetcatalog.com. "Octal High Voltage, High Current Darlington Transistor Array", http://pdf.datasheetcatalog.com/datasheet/motorola/ULN2804A.pdf [visitada em 11 de maio de 2014]

©DataSheet datasheetcatalog.com. "Microcontroller InSystem Programmable Flash", http://pdf.datasheetcatalog.com/datasheets2/90/90688_1.pdf [visitada em 11 de maio de 2014]

©DataSheet datasheetcatalog.com. "Agilent Slim Font Seven Segment Displays", http://pdf.datasheetcatalog.com/datasheet/atmel/2503S.pdf [visitada em 11 de maio de 2014]

©2014 Universiade Paulista. "Primeira Lei de Ohm ", http://adm.online.unip.br/img_ead_dp/20409.PDF [visitada em 17 de junho de 2014]

©2011 JumperOne. "Interfacing Microcontrollers: Darlington Transistors ", http://jumperone.com/2011/09/darlington/ [visitada em 17 de junho de 2014]

©2014 Electronica PT. "Código de cores das resistências-Online", http://www.electronica-pt.com/codigo-cor-resistores-online [visitada em 17 de junho de 2014]

©LOGIX4U.NET Logix4u. "A versatile Dynamic Link Library for parallel port interfacing", http://logix4u.net/parallel-port/16-inpout32dll-for-windows-982000ntxp [visitada em 20 de junho de 2014]

©LOGIX4U.NET Logix4u. "A tutorial on Parallel port Interfacing ", http://logix4u.net/parallel-port/15-a-tutorial-on-parallel-port-interfacing [visitada em 20 de junho de 2014]

©LOGIX4U.NET Logix4u. "Parallel port interfacing in Win32, using C/C++", http://www.hytherion.com/beattidp/comput/pport.htm [visitada em 20 de junho de 2014]

©HB Brand Electronic Components from China Manufacturer. "Resistor Calculator", http://www.hebeiltd.com.cn/?p=zz.led.resistor.calculator#series [visitada em 21 de junho de 2014]

©2005 Hytherion. "Parallel port interfacing in Win32, using C/C++", http://www.hytherion.com/beattidp/comput/pport.htm [visitada em 21 de junho de 2014]

©1999-2006 Rogercom - Antonio Rogério Messia. "Introdução a porta paralela" http://www.rogercom.com/pparalela/introducao.htm [visitada em 21 de maio de 2014]

©2014 Wikipédia. "Lei de Ohm**"**, http://pt.wikipedia.org/wiki/Lei_de_Ohm[visitada em 10 de junho de 2014]

©João Alexandre da Silveira. "Cartilha para Programação em Arduino**"**, http://ordemnatural.com.br/pdf-files/CartilhadoArduino_ed1.pdf [visitada em 30 de junho de 2014]

©2002 - 2014 Jameco. "Build Your Own Arduino Circuit on a Breadboard", http://www.jameco.com/jameco/workshop/jamecobuilds/arduinocircuit.html [visitada em 30 de julho de 2014

©2014 Arduino. “Language Reference”,
http://arduino.cc/en/Reference/HomePage [visitada em 30 de junho de 2014]