Pedro Paulo Vezzá Campos

Trabalho Prático 1: Interconexão de Computadores Usando Interface RS-232

Santa Catarina - SC, Brasil

3 de novembro de 2010

Pedro Paulo Vezzá Campos

Trabalho Prático 1: Interconexão de Computadores Usando Interface RS-232

Trabalho apresentado para avaliação na disciplina INE5414, do curso de Bacharelado em Ciências da Computação, turma 04208, da Universidade Federal de Santa Catarina, ministrada pelo professor Carlos Becker Westphall

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Santa Catarina - SC, Brasil

3 de novembro de 2010

Objetivos

Para este primeiro trabalho prático foi proposto que fosse desenvolvido um software que possibilitasse a comunicação através da interface serial RS-232 entre os dois microcomputadores, de tal forma que tudo que for digitado em um dos dois teclados apareça, simultaneamente, na tela de ambos microcomputadores.

Neste realatório serão apresentados uma introdução ao protocolo RS-232, uma descrição das atividades realizadas para cumprir os requisitos impostos e por fim será apresentado o código fonte documentado do programa concluído.

Interface RS-232

RS-232 (Conhecido ainda por EIA RS-232C ou ITU V.24) é um protocolo de comunicação serial binária entre um Terminal de Dados, DTE e um Comunicador de Dados, DCE. [**?**]

Sua utilização original era conectara um teletipo (TTY) a um modem. Atualmente o RS-232 está na terceira revisão (RS-232C), publicadada em 1969 para adequar-se às características elétricas desses dispositivos. Posteriormente, ele foi utilizado com outros propósitos, tais como conectar equipamentos já existentes. Até os anos 90 o a porta RS-232 foi onipresente nos IBM-PCs sendo a maneira padrão de conectar um modem a um PC.

Apesar de ter sido substituída em computadores modernos por barramentos mais velozes e compactos, tais como o USB e o Firewire, o RS-232 ainda possui grande importância como conexão para manutenção de equipamentos eletrônicos como televisões, decodificadores, no-breaks, dentre outros. Ainda, é uma maneira simples de conectar dispositivos remotos, tais como sensores externos que transmitem dados em baixa taxa de transferência.

Protocolo

No protocolo de comunicação RS-232 comumentemente os caracteres são enviados um a um através de uma comunicação assíncrona utilizando bits de start e stop. Nesse estilo de codificação há o uso de um bit de início, seguido por sete ou oito bits de dados, possivelmente um bit de paridade, e um, 1,5 ou dois bits de parada. Assim, normalmente são necessários $1+8+1=10$ bits para enviar um único caractere. Portanto a relação entre dados úteis e dados transmitidos é de $7/10=0,7$. O padrão define os níveis elétricos correspondentes aos níveis lógicos um e zero, a velocidade de transmissão padrão e os tipos de conectores.[**?**]

O padrão especifica 20 diferentes sinais de conexão, e um conector com formato de “D” (fig. 1) é comumente usado. São utilizados conectores machos e fêmeas - geralmente os conectores dos cabos são machos e os conectores de dispositivos são fêmeas - e estão disponíveis adaptadores m-m e f-f. Há também os chamados "null modems", que removem a necessidade de um modem interligando os dois terminais ao realizar uma ligação cruzada no cabeamento. Para a realização do trabalho prático foi utilizado um null modem, como será explicado adiante.

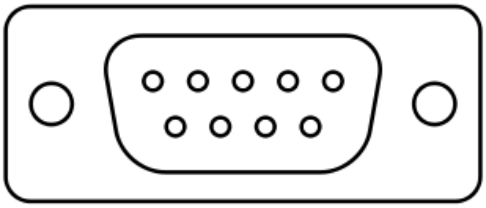

Figura 1: Diagrama de um conector DE9

O RS-232 é recomendado para conexões curtas (quinze metros ou menos) devido à grande capacitância gerada por cabos comuns de maior extensão. Os sinais variam de 3 a 15 volts positivos ou negativos, valores próximos de zero não são sinais válidos. O nível lógico um é definido por ser voltagem negativa, a condição de sinal é chamada "marca"e tem significado funcional de OFF (desligado). O nível lógico zero é positivo, a condição de sinal é chamda "espaço", e a função é ON (ligado). Níveis de sinal +-5, +-10, +- 12 e +-15 são vistos comumente, dependendo da fonte elétrica disponível.[**?**]

Três são os tipos de sinais nesses fios: terra, transmissão/recepção e "handshake". A tabela 1 (obtida de [**?**]) descreve de forma detalhada a função de cada um desses pinos:

Pino	Sinal	In/Out	Descrição
1	DCD	In	Data Carrier Detect
2	RxD	In	Receive Data
3	TxD	Out	Transmit Data
4	DTR	Out	Data Terminal Ready
5	GND	-	Ground
6	DSR	In	Data Set Ready
7	RTS	Out	Request to Send
8	CTS	In	Clear to Send
9	RI	In	Ring Indicator

Tabela 1: Pinos do conector DE-9

O sinal de terra tem a função de aterrar as outras conexões e é necessário para identificar o 0V. Se os equipamentos estiverem muito longe, o terra poderá mostrar-se diferente em cada

uma das pontas do cabo, podendo gerar falhas na comunicação. Em conectores de 25 pinos, o pino 7 geralmente é o terra (pino 1 e terra do chassis são raramente usados). Neste mesmo conector, os pinos 2 e 3 são os pinos de transmissão e recepção, um dispositivo deve enviar no 2 e receber no 3; o outro deve ser o contrário (se não, essa inversão deve ser feita no fim do cabo, como num cabo para null modem, também chamado de crossover).

Há várias configurações de software para conexões seriais. As mais comuns são velocidade e bits de paridade e parada. A velocidade é a quantidade de bits por segundo transmitida de um dispositivo para outro. Taxas comuns de transmissão são 300, 1200, 2400, 9600, 19200, etc. Tipicamente ambos os dispositivos devem estar configurados com a mesma velocidade, alguns dispositivos, porém, podem ser configurados para auto-detectar a velocidade. A paridade pode ser ímpar (todos os caracteres enviados terão um número ímpar de 1's), par ou não estar ativada. Normalmente também pode-se escolher nos softwares de comunicação serial entre usar 1 ou 2 bits de parada. [**?**]

Implementação do software de comunicação serial

Recursos utilizados

Windows XP Versão 32 bits com SP3

com0com Software livre que implementa diversos null modens virtuais, criando portas seriais virtuais utilizadas pelo programa criado.

Python 2.7 Linguagem de programação adotada

PySerial Biblioteca de interfaceamento com portas seriais fornecidas pelo sistema operacional.

Detalhamento do experimento

Diante da incapacidade de conectar fisicamente dois computadores por falta de portas DE-9 disponíveis e adaptadores DE-9/USB, foi adotada a opção de simular virtualmente uma porta serial. Para cumprir esse propósito foi adotado o com0com, um software livre exclusivo para Windows que atua como um null modem virtual criando portas seriais virtuais e interconectando-as. Através de uma interface gráfica (Fig 2) intuitiva é possível alterar as configurações de pinagem além de outras opções que permitem simular situações do mundo real como taxas de transferência, ruído, etc.

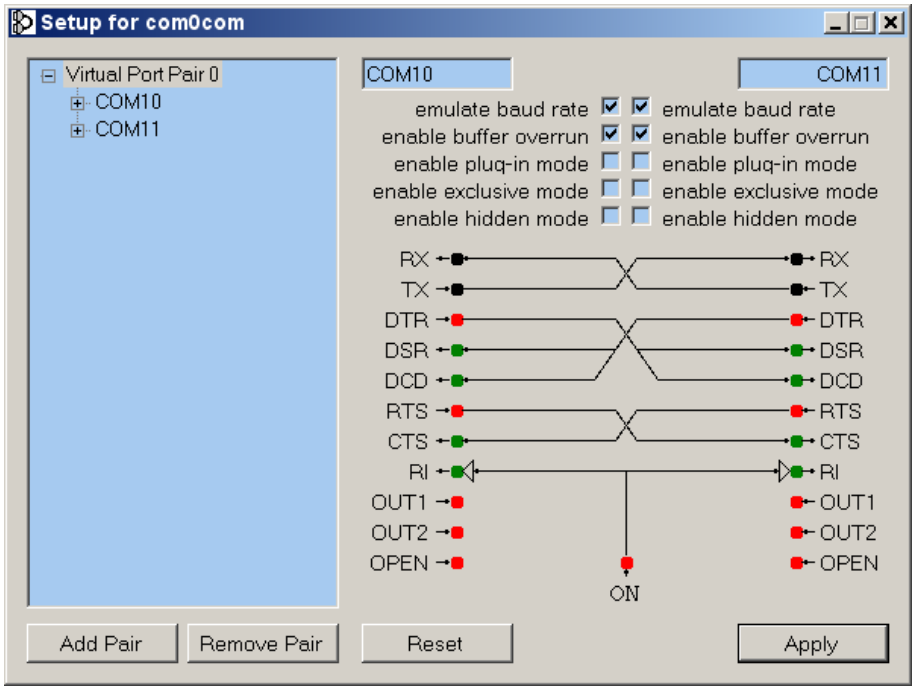

Figura 2: Interface do com0com

Após a correta configuração do com0com e teste através do HyperTerminal, ferramenta embutida no Windows XP para comunicação entre dispositivos, procedeu-se para a codificação do programa propriamente dito. Neste momento, Python apresentou-se como uma solução conveniente devido à facilidade de programação e rápida prototipação. Aliado ao Python há o PySerial, uma biblioteca de funções para interfaceamento entre programas Python e interfaces seriais. Seu funcionamento básico é através da criação de um objeto do tipo "Serial" que conecta-se a uma porta serial especificada e obedecendo diversas configurações que abrangem desde taxa de transferência, paridade, número de bits de stop, tamanho do byte, etc. O envio de dados é através da função "write()" que recebe como parâmetro uma string que é enviada ao terminal remoto, já o recebimento de dados ocorre através da função "read()" que retorna os caracteres armazenados no buffer da porta.

O funcionamento básico do programa é o seguinte:

- Apresentação do programa ao usuário;
- Pedido da porta desejada para conexão;
- Pedido das configurações desejadas de taxa de transmissão, tamanho do byte e número de bits de stop;
- Início da conexão

- Acionamento de duas “threads”, ambas realizando polling, uma para o recebimento de dados da porta e outra para o envio de dados

Código fonte

Abaixo segue o código fonte documentado do programa. Os pontos críticos do programa encontram-se nas threads, onde é realizado o polling de recebimento e envio de dados, ainda, a função “iniciarConexao()” é responsável por coletar as configurações desejadas e iniciar a conexão na porta escolhida.

```
1  # coding: utf-8
2  from serial import Serial
3  from threading import Thread
4  from Getch import getch
5  import sys
6
7  def main():
8      """
9      Funcao incial do programa. Invoca
10     uma funcao que introduz o usuario ao
11     programa e inicia a conexo.
12     """
13     exibirIntroducao()
14     iniciarConexao()
15
16 def exibirIntroducao():
17     """
18     Exibe uma pequena introducao ao usuario do programa
19     """
20
21     print "INE5414 - REDES DE COMPUTADORES I"
22     print "PEDRO PAULO VEZZ CAMPOS"
23     print "TRABALHO PRTICO 1: INTERCONEXO DE COMPUTADORES USANDO RS
24
25 def iniciarConexao():
26     """
```

```
27      Fun  o principal do programa. Questiona o usu rio das
28      configura  es desejadas, inicia a conex o com a porta
29      serial definida e inicia duas threads: Uma respons vel
30      por ler os caracteres escritos pelo usu rio e outra
31      respons vel por escrever os caracteres recebidos do
32      terminal remoto.
33      """
34      print "Forne a a porta serial a conexao utilizada: "
35      porta = raw_input()
36      if porta.startswith("CNC"):
37          porta = "\\\\.\\" + porta
38
39      print "Forne a a taxa de transfer ncia (2400 bits/s): "
40      taxa = geraValorDefault(raw_input(), 2400)
41
42      print "Forne a o tamanho do byte (8 bits): "
43      tamByte = geraValorDefault(raw_input(), 8)
44
45      print "Forne a o n mero de stop bits (1 bit): "
46      stopBits = geraValorDefault(raw_input(), 1)
47
48      conexao = Serial(port=porta, baudrate=taxa, bytesize=tamByte, stopb
49
50      print "Conex o iniciada. Escreva sua mensagem, ela ser  exibida n
51      Sender(conexao).start()
52      Receiver(conexao).start()
53      conexao.close
54
55  def geraValorDefault(valor, default):
56      """
57      Gera um inteiro parseado de valor se valor
58          diferente de vazio sen o retorna o valor
59      passado em default.
60      """
61      if valor == "":
```

```
62          valor = default
63      else:
64          valor = int(valor)
65      return valor
66
67  class Sender(Thread):
68      """
69      Classe responsável por realizar um polling
70      capturando uma letra por vez, exibindo-a
71      ao usuário e enviando-a via
72      interface serial.
73      """
74      def __init__(self, conexao):
75          Thread.__init__(self)
76          self.conexao = conexao
77
78      def run(self):
79          while True:
80              ch = getch()
81              sys.stdout.write(ch)
82              self.conexao.write(ch)
83
84  class Receiver(Thread):
85      """
86      Classe responsável por realizar um polling
87      lendo uma letra por vez da interface serial
88      e exibindo-a ao usuário.
89      """
90      def __init__(self, conexao):
91          Thread.__init__(self)
92          self.conexao = conexao
93
94      def run(self):
95          while True:
96              sys.stdout.write(self.conexao.read())
```

```
97
98 if __name__ == "__main__":
99     main()

1 ## {{{ http://code.activestate.com/recipes/134892/ (r2)
2 class _Getch:
3     """Gets a single character from standard input.  Does not echo to t
4 screen."""
5     def __init__(self):
6         try:
7             self.impl = _GetchWindows()
8         except ImportError:
9             self.impl = _GetchUnix()
10
11     def __call__(self): return self.impl()
12
13
14 class _GetchUnix:
15     def __init__(self):
16         import tty, sys
17
18     def __call__(self):
19         import sys, tty, termios
20         fd = sys.stdin.fileno()
21         old_settings = termios.tcgetattr(fd)
22         try:
23             tty.setraw(sys.stdin.fileno())
24             ch = sys.stdin.read(1)
25         finally:
26             termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
27         return ch
28
29
30 class _GetchWindows:
31     def __init__(self):
32         import msvcrt
```

```
33
34      def __call__(self):
35          import msvcrt
36          ch = msvcrt.getch()
37          if ch == "\r":
38              return "\r\n"
39          return ch
40
41
42  getch = _Getch()
43  ## end of http://code.activestate.com/recipes/134892/ }}}
```