BỘ GIÁO DỤC VÀ ĐÀO TẠO

HUTECH
ĐẠI HỌC KỸ THUẬT CÔNG NGHỆ TP.HCM

TÀI LIỆU THỰC HÀNH

TRUYỀN SỐ LIỆU

Biên soạn: Thái Hồ Phú Hào

Thực Hành Truyền Số Liệu

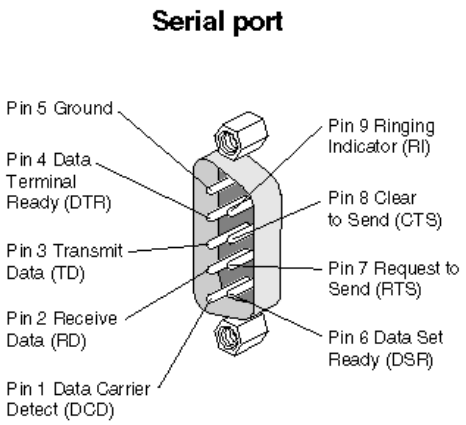
Ấn bản 2017

MỤC LỤC

Mục Lục

BÀI 1: KẾT NỐI SERIAL TRONG C#

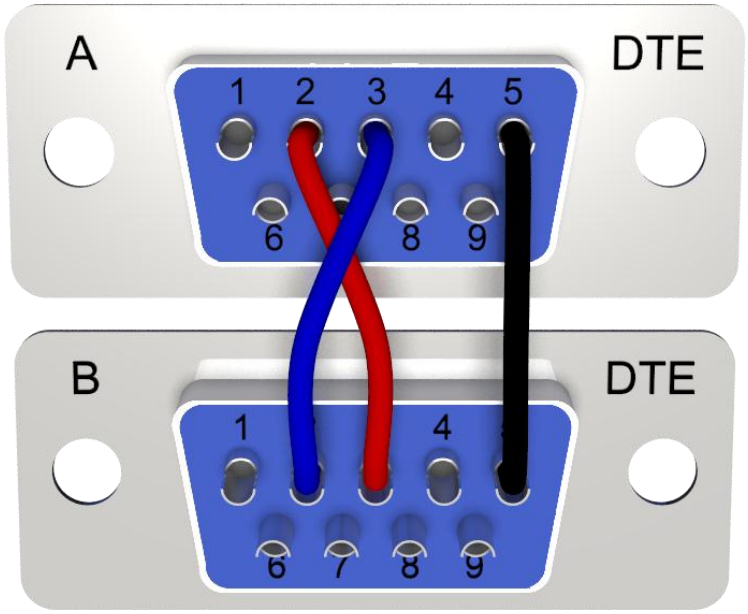
1.1 GIỚI THIỆU



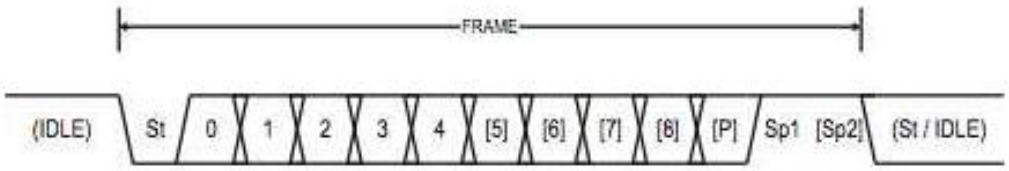
- Là cổng kết nối rất phổ biến trong các thiết bị công nghiệp, dân dụng.
- Loại cổng kết nối: DB9 – Male.
- Chuẩn truyền thông RS232.
- Ý nghĩa chân kết nối:
    - **RXD** (receive Data): Đường nhận dữ liệu.
    - **TXD** (Transmit Data): Đường gửi dữ liệu.
    - DTR (Data Terminal Ready): Báo DTE sẵn sàng .Chân DTR thường ở trạng thái ON khi thiết bị đầu cuối sẵn sàng thiết lập kênh truyền thông (tự động quay số hay tự động trả lời). DTR ở trạng thái OFF chỉ khi thiết bị đầu cuối không muốn DCE của nó chấp nhận lời gọi từ xa.
    - DSR (Data Set Ready): Báo DCE sẵn sàng, ở chế độ trả lời, 1 tone trả lời và DSR ON sau 2 giây khi Modem nhấc máy.

- DCD (Data Carrier Detect): Tín hiệu này tích cực khi Modem nhận được tín hiệu từ trạm từ xa và nó duy trì trong suốt quá trình liên kết.
- RTS (Request To Send): Đường RTS kiểm soát chiều truyền dữ liệu. Khi một trạm cần gửi dữ liệu, nó đóng mạch RTS sang ON để báo hiệu với modem của nó.
- CTS (Clear To Send): Khi CTS chuyển sang ON, Modem xác nhận là DTE có thể truyền số liệu. Quá trình ngược lại nếu đổi chiều truyền số liệu
- RI (Ring Indicator): Khi modem nhận được tín hiệu chuông, RI chuyển ON/OFF một cách tuần tự với chuông điện thoại để báo hiệu cho trạm đầu cuối. Tín hiệu này chỉ thị rằng một modem xa yêu cầu thiết lập liên kết dial-up.

- Sơ đồ kết nối đơn giản:
  - Chỉ sử dụng 2 đường tính hiện: RXD – TXD nối chéo giữa 2 thiết bị.



  - Đường tín hiệu GND nối giữa 2 thiết bị.
- Chuẩn truyền thông: RS232
  - Phương thức truyền thông nối tiếp không đồng bộ.
  - Dữ liệu được truyền/nhận theo từng byte.
  - Khung dữ liệu:

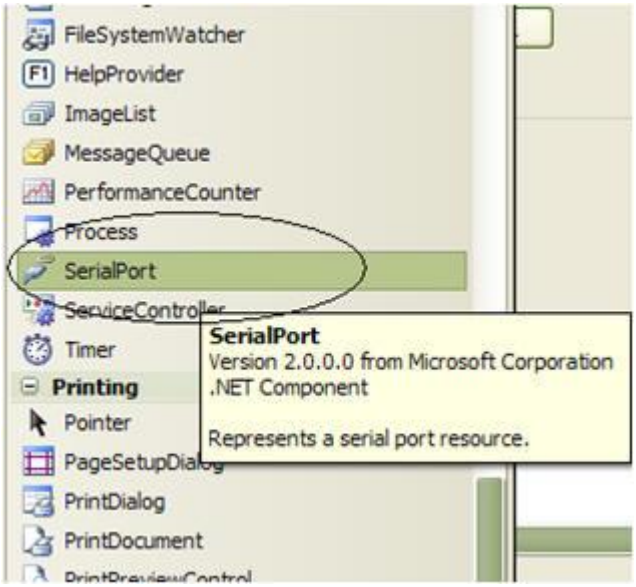


St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

- Các tham số:
  - Baudrate: tốc độ truyền/nhận dữ liệu.
  - Data bit: số bit dữ liệu có trong khung dữ liệu.
  - Parity: kiểm tra chẵn/lẻ khi truyền.
  - Buffer Size: bộ đệm dữ liệu.
  - Stop bit: số bit stop có trong khung dữ liệu.

1.2 Đối tượng Serial Port trong VC#

- Là đối tượng kết nối truyền thông nối tiếp của máy tính

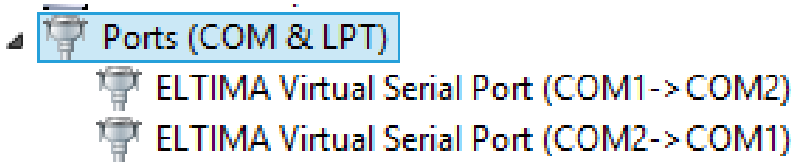


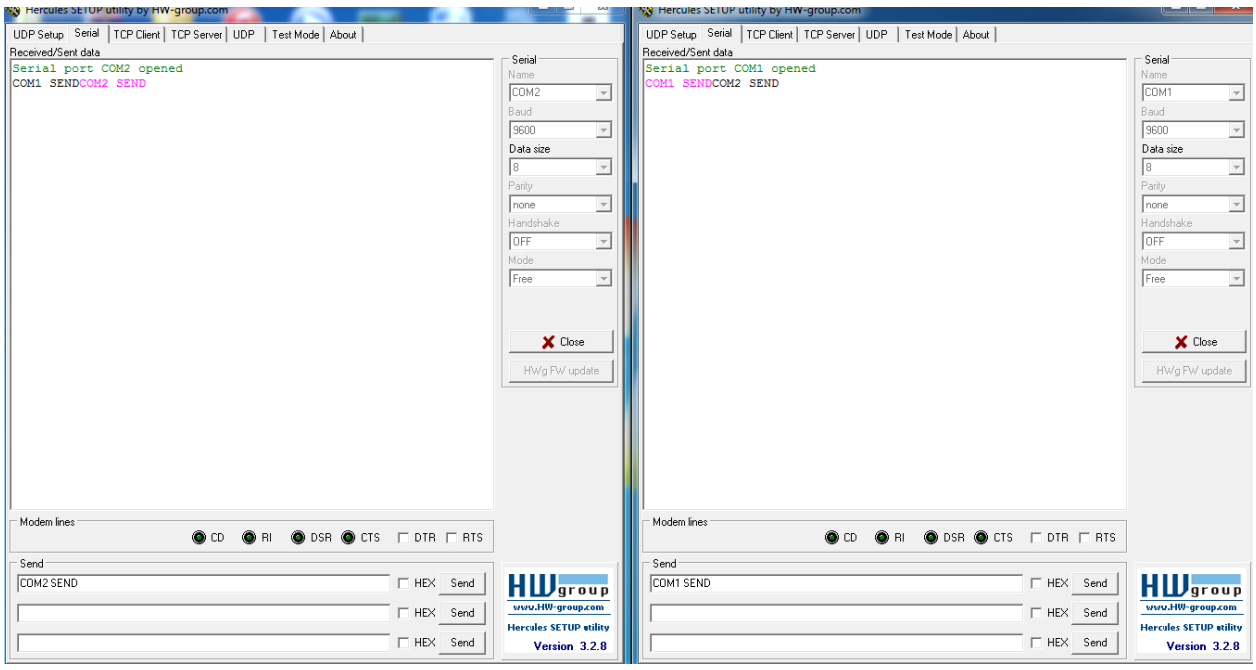
- Công cụ thực hành:
    - Phần mềm kiểm tra hoạt động:
        - ***http://www.hw-group.com/products/hercules/index_en.html***
    - Phần mềm tạo cổng COM ảo:
        - ***http://www.eltima.com/products/vspdxp/***
    - Link tham khảo:
        - ***http://www.codeproject.com/Tips/361285/Serial-Port-Communication***

1.3 Ví dụ

Ví dụ:

- Tạo cổng COM ảo trên máy tính
- Kiểm tra trao đổi dữ liệu giữa các cổng COM.





Một số hàm trong SerialPort thường dùng:

- SerialPort.IsOpen(): Trả lại trang thái của cổng là đang đóng hay mở.
- SerialPort.Open(): Mở cổng với thông số đã cài đặt.
- SerialPort.Close(): Đóng cổng.
- SerialPort.WriteLine(String data): Truyền một string xuống bộ đệm cổng để truyền đi.
- SerialPort.ReadExisting (): Đọc một string từ bộ đệm cổng.
- SerialPort.ReadChar(): Đọc một giá trị kiểu char từ bộ đệm cổng.
- SerialPort.ReadByte(): Đọc một giá trị kiểu Byte từ bộ đệm cổng.

1.4 Bài tập

- Xây dựng chương trình đóng/mở cổng COM.
- Kiểm tra trạng thái cổng COM đang đóng (close) hay mở (open).
- Cho phép cài đặt các tham số của cổng COM. (home)
- Cho phép truyền dữ liệu qua cổng COM.
- Cho phép nhận dữ liệu qua cổng COM:
  - Sử dụng nút nhận.
  - Sử dụng Timer. (home)
  - Sử dụng sự kiện nhận dữ liệu. (home)

BÀI 2: LẬP TRÌNH ANALOG TO DIGITAL CONVERTER

2.1 Lý thuyết

Để thực hiện chuyển đổi từ tín hiệu tương tự sang tín hiệu số, thiết bị chuyển đổi phải thực hiện thông qua 4 bước:

- Lấp mẫu (Sampling).
- Lượng tử hóa (Quantion).
- Mã hóa (Encoding).
- Né (Compression).

- **Bước 1: Lấy mẫu**

Tín hiệu tương tự được biểu diễn dưới dạng đồ thị hàm Sin. Nhìn vào hình 2.1 phía dưới sẽ hình dung được lấy mẫu là gì: Lấy mẫu có nghĩa là chia nhỏ tín hiệu tương tự thành nhiều phần bằng nhau và đúng thời gian theo chu kỳ các tín hiệu tương tự chia nhỏ đó sẽ được lấy.

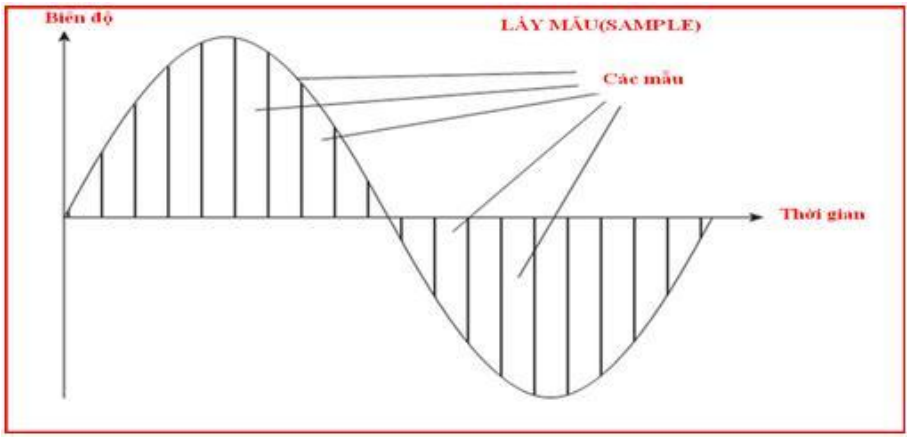
Định luật lấy mẫu Nyquist: Một tín hiệu có biên độ giới hạn có thể được biểu diễn và khôi phục chính xác bằng tín hiệu rời rạc nếu tần số lấy mẫu của tín hiệu phải đảm bảo điều kiện: $\boldsymbol{F_s \geq 2F_{max}}$

Trong đó:

- F_s (sample): Tần số lấy mẫu.
- F_{max}: Tần số lớn nhất của tín hiệu cần lấy mẫu.

(Tần số là số lần dao động trên một đơn vị thời gian và nó bằng f=1/T. Còn chu kỳ là khoảng thời gian để thực hiện đủ một dao động)

- Tần số lấy mẫu: Là số mẫu lấy được trong vòng một giây.
- Tần số của tín hiệu: Là số tín hiệu giao động trong vòng một giây.



Hình 2.1: Lấy mẫu tín hiệu

Ví dụ:

- Tần số con người có thể nghe nằm trong khoảng: 20 – 20.000Hz
- Tần số giọng nói của con người: 200 – 9.000Hz
- Tần số con người thường nói nhất: 300 – 4.000Hz

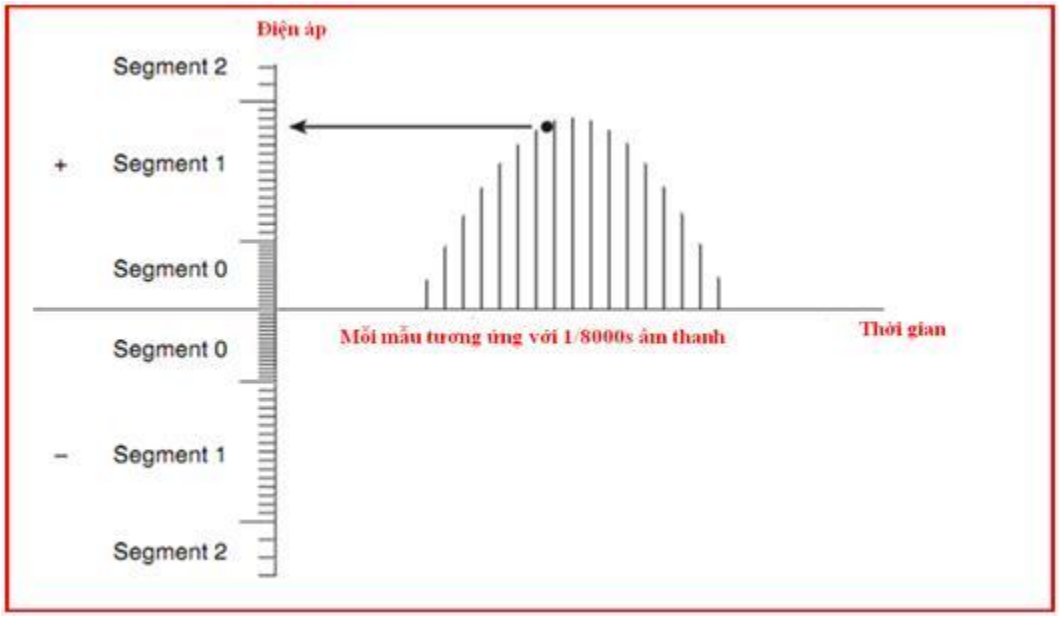
Vậy theo định luật *Nyquist* thì để con người có thể nghe cần phải lấy mẫy là 2 x 9.000 = 18.000Hz. Tức là phải lấy 18.000 mẫu trong vòng một giây. Và việc lấy nhiều mẫu như thế sẽ chiếm băng thông của cuộc gọi, trong khi đó thì dãi băng tần mà con người thường nói nhất chỉ 300 – 4.000Hz. Do đó, *Nyquist* mới quyết định cắt giãm tần số từ 9.000Hz còn 4.000Hz, theo định luật sẽ tính được F_s = 2 x 4.000 = 8.000Hz. Phải lấy 8.000 mẫu trong vòng một giây.

Tuy nhiên, việc cắt giảm tần số như trên sẽ ảnh hưởng:

- Chất lượng cuộc gọi, chất lượng âm thanh nhưng vẫn còn đủ để có thể nghe được, để cảm nhận được giọng nói qua đường thoại.
- Có một số âm thanh sẽ không được lấy mẫu, và đầu dây đang nghe điện thoại sẽ không nghe được đầu phát nói cái gì, nhưng những âm thanh này chiếm tỷ lệ rất nhỏ trong cuộc gọi hằng ngày. Do đó, có thể hoàn toàn bỏ qua những âm thanh đó.

- **Bước 2: Lượng tử hóa**

Sau khi đã có 8.000 mẫu từ bước trên,tiến hành gán từng mẫu này với những mức điện áp (voltage) tương ứng. Đó chính là quá trình lượng tử hóa.



Hình 2.2: Lượng tử hóa mẫu tính hiệu

Như hình 2.2, mỗi mẫu sẽ ứng với một mức điện áp tương ứng (nửa trên trục thời gian là điệp áp dương, nửa dưới trục thời gian là điện áp âm). Điện áp được chia làm Segment chạy từ "-7" đến "+7". Mỗi Segment được chia làm 16 phần bằng nhau. Mỗi mẫu thì có 8 bit nhị phân để biểu diễn. Từ đó có thể tính được 256 khả năng có thể xảy ra.

- **Bước 3: Mã hóa**

Với việc lấy mẫu 8.000 mẫu/giây, sử dụng 8 bit nhị phân để mã hóa cho từng mẫu. Cứ mỗi mẫu được lượng tử hóa ở trên sẽ được biểu diễn dưới dạng nhị phân (bit "0" hoặc bit "1"). Tiến trình này còn được biết với cái tên Pulse-Code Modulation (PCM). Bằng cách chuyển đổi số nhị phân 8 bit cho mỗi mẫu trong 8.000 mẫu. Tính được băng thông yêu cầu tối thiểu cho thoại là 8.000 mẫu × 8 bit/mẫu = 64.000 bps= 64 kbps.

- **Bước 4: Nén**

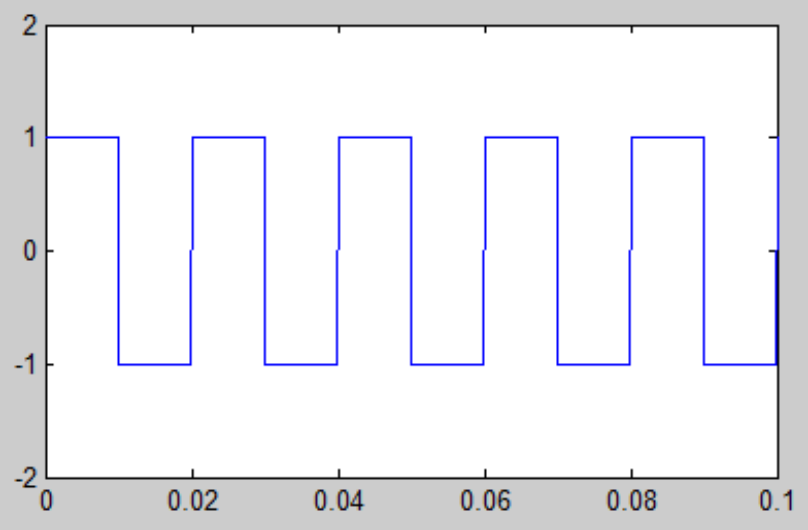
Mục đích của việc nén tín hiệu là để tiết kiệm đường truyền khi băng thông không cho phép. Phương pháp nén có thể làm giảm chất lượng cuộc điện thoại nhưng phải nằm trong tiêu chuẩn cho phép. Có một số tiêu chuẩn giọng nói quốc tế ITU (International Telecommunication Union). Tùy thuộc vào các yếu tố sau đây mà thuật toán dùng để nén giọng nói thoại khác nhau:

- Băng thông trên đường truyền.
- Chất lượng tín hiệu trên đường truyền.
- Độ trễ của tín hiệu.
- Việc xử lý của CPU cho các thuật toán phức tạp.

2.2 Lập trình trên matlab

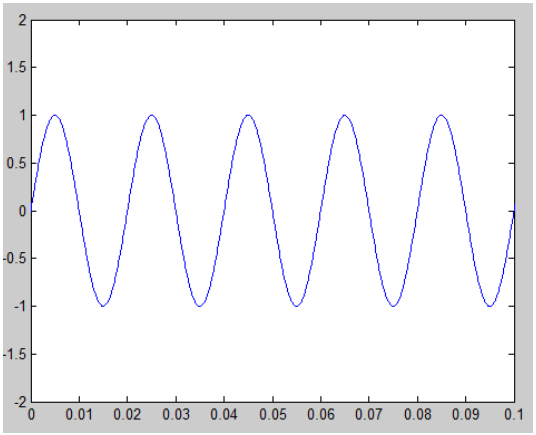
- **Tạo sóng vuông**

```
f = 10000

t = 0:1/f:1.5

x = square(2*pi*50*t,50)

plot(t,x)

axis([0 0.1 -2 2])
```



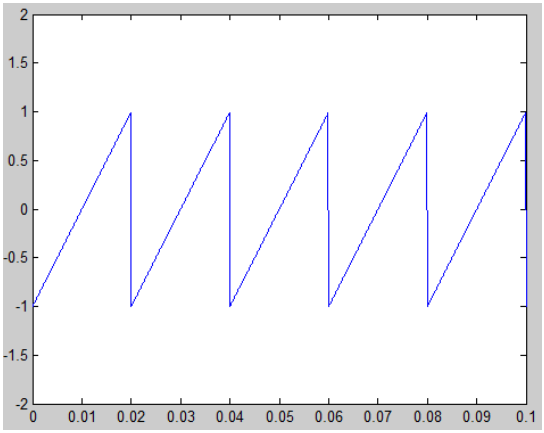
- **Tạo sóng sin**

```
f = 10000

t = 0:1/f:1.5

x = sin(2*pi*50*t)

plot(t,x)

axis([0 0.1 -2 2])
```



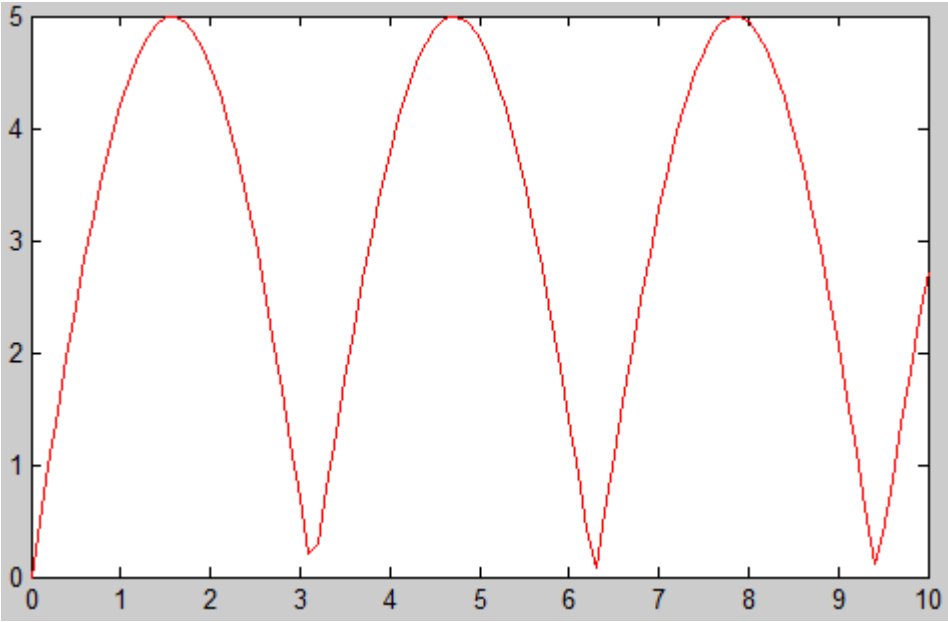
- **Tạo sóng răng cưa**

```
f = 10000

t = 0:1/f:1.5

x = sawtooth(2*pi*50*t)

plot(t,x)

axis([0 0.1 -2 2])
```



Ví dụ : cho tín hiệu analog xác định bởi hàm :

```
y = abs(5*sin(t))
```



Chuyển đổi trên thành tín hiệu digial với độ rộng 3 bit.

```
f=10
n=3 %số bit
q=f/(2^n-1) % lượng tử hóa
t=0:0.1:10 % trục thời gian
y = abs(5*sin(t)) % tín hiện analog
x0=fix(y/q) %làm tròn điểm lấy mẫu
y0=dec2bin(x0,n) %chuyển thành nhị phân n bit
y1=x0*q
plot(t,y,'r') % vẽ đồ thị của hàm y theo t với màu đỏ
hold on
plot (t,y1,'b') % vẽ đồ thị hàm y1 theo t với màu xanh
hold off
```

BÀI 3: MÔ PHỎNG TRUYỀN SỐ LIỆU BẰNG MATLAB

3.1 Mục đích

- Cung cấp những kiến thức cơ bản về Matlab Simulink như:
  - Bộ tạo sóng sin.
  - Scope.
  - Bộ tạo chuỗi bit.
  - Bộ tạo nhiễu.
  - Bộ đếm lỗi bit.

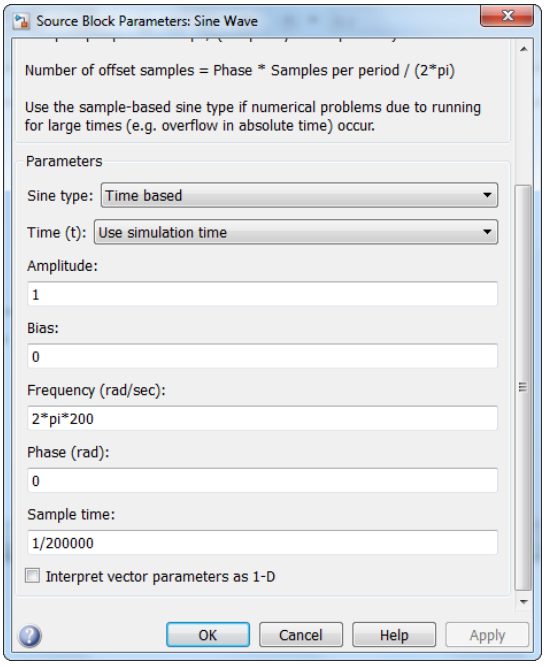
3.2 Các khối cơ bản

- Bộ tạo sóng sin:

Vào Simulink/Sources/Sin Wave  Sine Wave

Tạo tín hiệu $s(t) = 1\sin(2\pi 200t)$

Ta cấu hình bộ Sin Wave như sau:



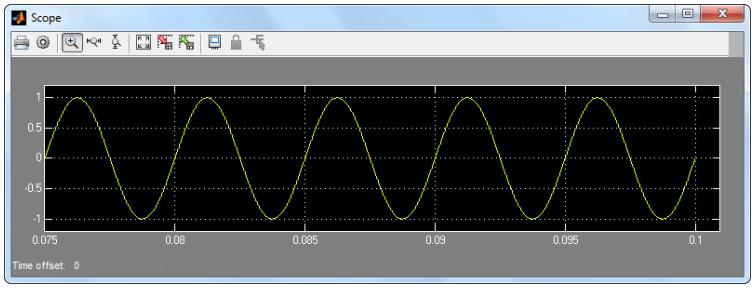
Chú ý:

Sample time (T_s) là thời gian lấy mẫu. $T_S = \frac{1}{f_s}$, f_s phải chọn sao cho $f_s \geq 2.f_m$, với f_m là tần số của tín hiệu.

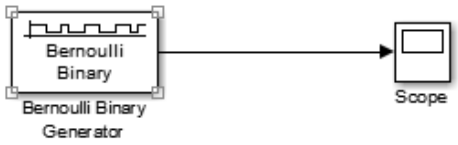
- Scope: Vào Simulink/Sinks/Scope  Scope

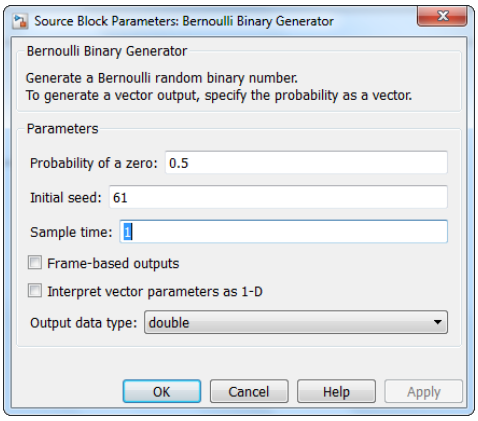
Scope hiển thị tịn hiệu trong miền thời gian. Hiển thị tín hiệu

$$s(t) = 1\sin(2\pi 200t)$$



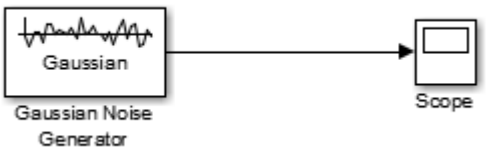
- Bộ tạo dữ liệu số: Vào Communications System Toolbox\Comm Surces\Random Data Sources
  - Bernoulli Binary Generator: tạo chuỗi bit ngẫu nhiên

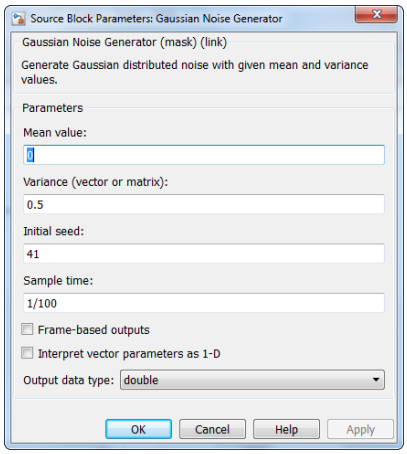


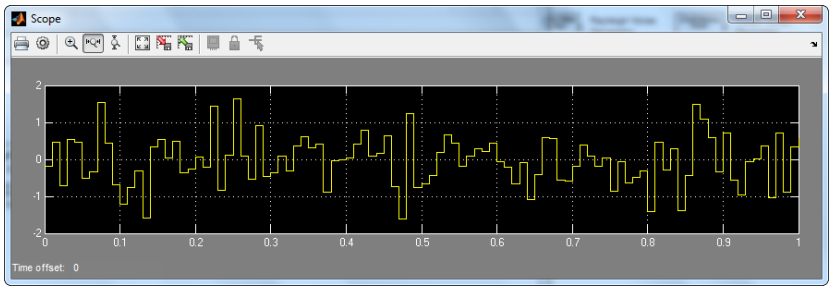


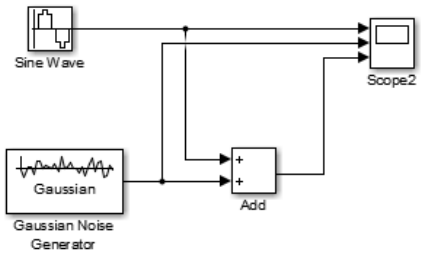
Tốc độ là 1 bps

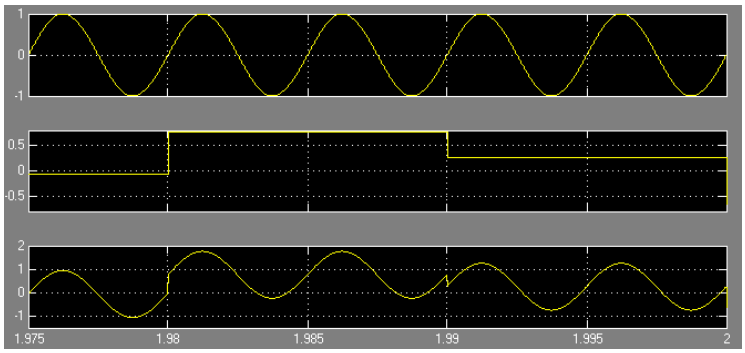
  - Gaussian Noise Generator: Bộ tạo nhiễu (vào Communications System Toolbox/Comm Sources/Noise Generators)



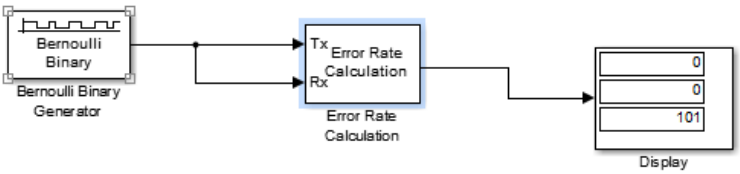








- Bộ đếm lỗi bit:



BÀI 4: HAMMING CODE

4.1 Thuật toán

Mã Hamming là một mã sửa lỗi tuyến, mã này có thể phát hiện một bit hoặc hai bit bị lỗi. Mã Hamming còn có thể sửa các lỗi do một bit bị sai gây ra.

- Các vị trí trong một khối được bắt đầu đánh số từ 1: Bit 1, 2, 3…
- Tất cả các bít ở vị trí số mũ của 2 được dùng làm bít chẵn lẻ. Vd:2^0, 2^1, 2^2, 2^3…
- Tất cả các vị trí khác vị trí còn lại sẽ được dùng cho các bit dữ liệu.

Mỗi bit chẵn lẻ tính giá trị chẵn lẻ cho một số bit trong từ mã theo quy tắc như sau:

- Bít chẵn lẻ ở vị trí số 1 (2^0) sẽ kiểm tra tính chẵn lẻ của các vị trí mà số thứ tự của vị trí đó trong hệ nhị phân có bit ngoài cùng là 1. Vd: 1 ($\underline{1}_2$), 3 ($1\underline{1}_2$), 5 ($10\underline{1}_2$), 7 ($11\underline{1}_2$), 9 ($100\underline{1}_2$) ..
- Bít chẵn lẻ ở vị trí số 2 (10_2) sẽ kiểm tra tính chẵn lẻ của các vị trí mà số thứ tự của vị trí đó trong hệ nhị phân có bit thứ 2 là 1. Vd: 2 ($\underline{1}0_2$), 3 ($1\underline{1}1_2$), 6 ($1\underline{1}0_2$), 7 ($1\underline{1}1_2$), 10 ($10\underline{1}0_2$)…
- Bít chẵn lẻ ở vị trí số 4 (100_2) sẽ kiểm tra tính chẵn lẻ của các vị trí mà số thứ tự của vị trí đó trong hệ nhị phân có bit thứ 3 là 1. Vd: 4($\underline{1}00_2$), 5($\underline{1}01_2$), 6($\underline{1}10_2$), 7($\underline{1}11_2$), 12($\underline{1}100_2$), 13($\underline{1}101_2$), 14($\underline{1}110_2$), 15($\underline{1}111_2$)…
- Tương tự với các bít chẵn lẻ ở vị trí cao hơn: 8, 16, 32…

Mã chẵn lẻ thêm một bit vào trong dữ liệu, và bit cho thêm này cho biết số lượng bit có giá trị 1 của đoạn dữ liệu nằm trước là một số chẵn hay một số lẻ. Nếu một bit bị thay đổi trong quá trình truyền dữ liệu, giá trị chẵn lẻ trong thông điệp sẽ thay đổi và do đó có thể phát hiện được lỗi (Chú ý rằng bit bị thay đổi có thể lại chính là bit kiểm tra). Theo quy ước chung, bit kiểm tra có giá trị bằng 1 nếu số lượng bit có giá

trị một trong dữ liệu là một số lẻ, và giá trị của bit kiểm tra bằng 0 nếu số lượng bit có giá trị một trong dữ liệu là một số chẵn. Nói cách khác, nếu đoạn dữ liệu và bit kiểm tra được gộp lại cùng với nhau, số lượng bit có giá trị bằng 1 luôn luôn là một số chẵn.

4.2 Ví dụ

Lấy ví dụ chúng ta có một từ dữ liệu dài 7 bit với giá trị là "0110101". Để chứng minh phương pháp các mã Hamming được tính toán và được sử dụng để kiểm tra lỗi, xin xem bảng liệt kê dưới đây. Chữ d (data) được dùng để biểu thị các bit dữ liệu và chữ p (parity) để biểu thị các bit chẵn lẻ (parity bits).

Đầu tiên, các bit của dữ liệu được đặt vào vị trí tương thích của chúng, sau đó các bit chẵn lẻ cho mỗi trường hợp được tính toán dùng quy luật bit chẵn lẻ số chẵn[1].

Thứ tự bit	1	2	3	4	5	6	7	8	9	10	11
Vị trí bit chẵn lẻ và các bit dữ liệu	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7
Nhóm dữ liệu (không có bit chẵn lẻ):			**0**		**1**	**1**	**0**		**1**	**0**	**1**
p_1	**1**		0		1		0		1		1
p_2		**0**	0			1	0			0	1
p_3				**0**	1	1	0				
p_4								**0**	1	0	1
Nhóm dữ liệu (với bit chẵn lẻ):	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**

Dữ liệu truyền đi (bao gồm các bit chẵn lẻ) là "**10001100101**". Nếu ta thử cho rằng bit cuối cùng bị truyền sai và bị lộn ngược từ 1 sang 0. Dữ liệu nhận được sẽ là "**10001100100**".

Phân tích quy luật kiến tạo mã Hamming bằng cách cho bit chẵn lẻ giá trị 1 khi kết quả kiểm tra dùng quy luật số chẵn bị sai.

Thứ tự bit	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**		
Vị trí bit chẵn lẻ và các bit dữ	**p_1**	**p_2**	**d_1**	**p_3**	**d_2**	**d_3**	**d_4**	**p_4**	**d_5**	**d_6**	**d_7**	**Kiểm chẵn**	**Bit chẵn**
Nhóm dữ liệu nhận được:	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	
p_1	**1**		0		1		0		1		0	**Sai**	**1**
p_2		**0**	0			1	0			0	0	**Sai**	**1**
p_3				**0**	1	1	0					Đúng	0
p_4								**0**	1	0	0	**Sai**	**1**

Định giá trị của các bit chẵn lẻ (nên nhớ bit nằm dưới cùng được viết về bên phải - viết ngược lại từ dưới lên trên). Giá trị số nguyên của các bit chẵn lẻ là $11_{(10)}$, và như vậy có nghĩa là bit thứ 11 trong nhóm dữ liệu (data word) - bao gồm cả các bit chẵn lẻ - là bit có giá trị không đúng, và bit này cần phải đổi ngược lại.

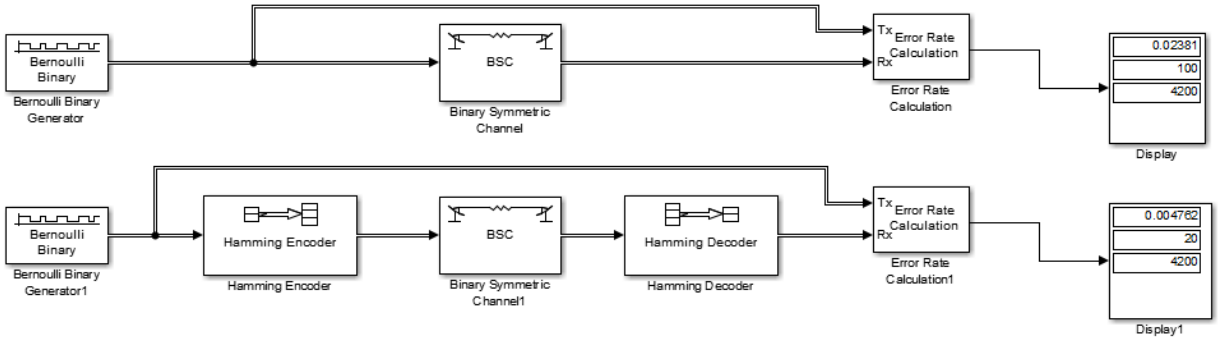
	p_3	**p_2**	**p_1**		
Nhị phân	1	0	1	1	
Thập phân	**8**		**2**	**1**	**Σ = 11**

Việc đổi ngược giá trị của bit thứ 11: **10001100100** trở lại thành **10001100101**.

Bằng việc bỏ đi phần mã Hamming, chúng ta lấy được phần dữ liệu gốc với giá trị là **0110101**.

4.3 Bài tập

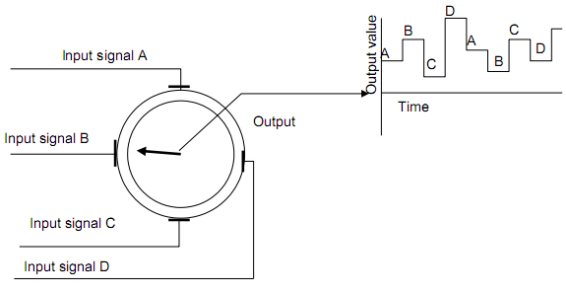
Câu 1: Dùng công cụ Matlab Simulink mô phỏng quá trình sửa lỗi bằng hamming code với tỷ lệ lỗi trên đường truyền là 2%, so sánh quá trình truyền áp dụng thuật toán sửa lỗi hamming code và không áp dụng thuật toán sửa lỗi.


Bernoulli Binary
Bernoulli Binary Generator
BSC
Binary Symmetric Channel
Tx
Rx
Error Rate Calculation
Error Rate Calculation
0.02381
100
4200
Display
Bernoulli Binary
Bernoulli Binary Generator1
Hamming Encoder
Hamming Encoder
BSC
Binary Symmetric Channel1
Hamming Decoder
Hamming Decoder
Tx
Rx
Error Rate Calculation
Error Rate Calculation1
0.004762
20
4200
Display1

BÀI 5: TIME DIVISION MUTIPLEXING (TDM)

5.1 MÔ HÌNH TDM

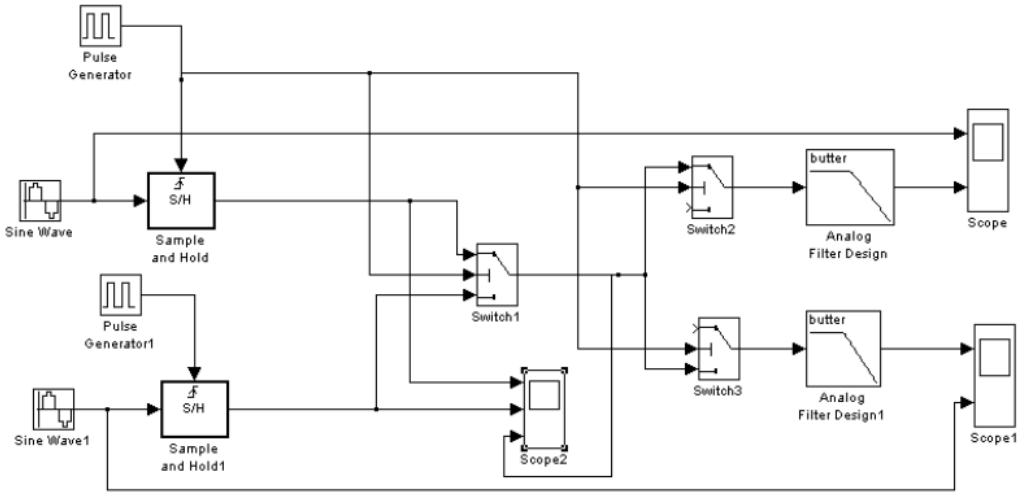
Mô hình trên dẫn TDM được minh họa như hình 5.1. Các luồng dữ liệu được truyền trên từng khe thời gian.



Hình 5.1: Mô hình TDM

5.2 Mô Phỏng TDM

Thiết kế bộ TDM để truyền hai luồng dữ liệu có tần số là 100Hz và 200Hz. Bộ phát xung pulse có tần số 2KHz, lệch pha 180 độ.



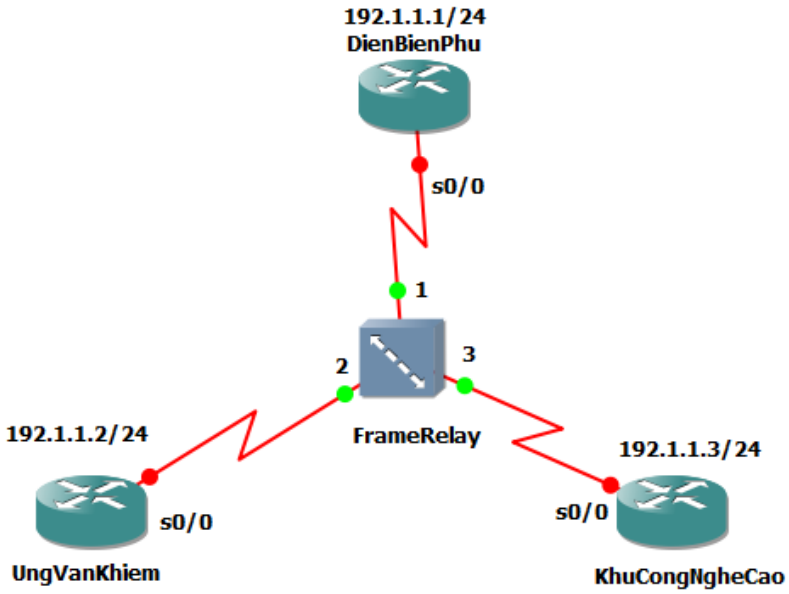
BÀI 6: FRAME RELAY

6.1 Yêu Cầu

Frame Relay là dịch vụ nối mạng dữ liệu theo phương thức chuyển mạch gói, hoạt động ở mức liên kết (link level) và rất thích hợp với truyền số liệu dung lượng lớn.

Frame Relay tiết kiệm đáng kể so với đường thuê riêng (private line) nhờ tính năng dồn kênh cho phép thiết lập nhiều kết nối (hoặc cuộc thoại) trên cùng một đường dây vật lý. Trên đường vật lý kết nối duy nhất, Frame Relay hỗ trợ nhiều ứng dụng khác nhau của khách hàng như TCP/IP, NetBIOS, SNA..., cho cả các ứng dụng thoại. Nhờ vậy tiết kiệm chi phí băng thông, đường dây cũng như thiết bị truyền dẫn và thiết bị kết nối.

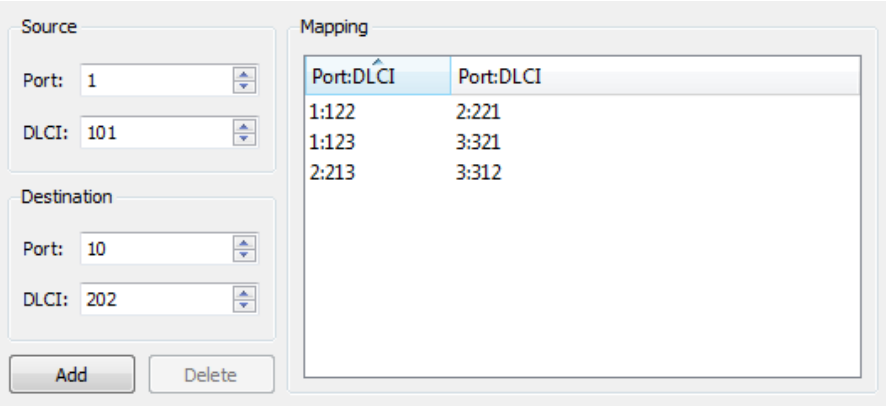
Yêu cầu: thiết kế mô hình truyền dữ liệu frame relay tại 3 chi nhánh của 1 công ty, dựa trên đường truyền có sẵn.



6.1 Hướng Dẫn

Thực hiện cấu hình Frame-Relay switch theo sơ đồ map DLCI như sau:

- Port 1 DLCI 122 → Port 2 DLCI 221
- Port 1 DLCI 123 → Port 3 DLCI 321
- Port 2 DLCI 213 → Port 3 DLCI 312



- Start các thiết bị và cấu hình từng thiết bị như sau:
  - Cấu hình tại Router DienBienPhu :

```
R1(config)#int serial 0/0
R1(config-if)#ip address 192.1.1.1 255.255.255.0
R1(config-if)#encapsulation frame-relay
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#exit
R1#write
```

  - Cấu hình tại Router UngVanKhiem :

```
R1(config)#int serial 0/0
R1(config-if)#ip address 192.1.1.2 255.255.255.0
R1(config-if)#encapsulation frame-relay
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#exit
R1#write
```

- Cấu hình tại Router KhuCongNgheCao :

R1(config)#int serial 0/0

R1(config-if)#ip address 192.1.1.3 255.255.255.0

R1(config-if)#encapsulation frame-relay

R1(config-if)#no shutdown

R1(config-if)#exit

R1(config)#exit

R1#write

- Kết quả thiết lập trường đường truyền số liệu giữa các chi nhánh lại với nhau:

```
Router#sho frame-relay map
Serial0/0 (up): ip 192.1.1.2 dlci 122(0x7A,0x1CA0), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.1.1.3 dlci 123(0x7B,0x1CB0), dynamic,
              broadcast,, status defined, active
```

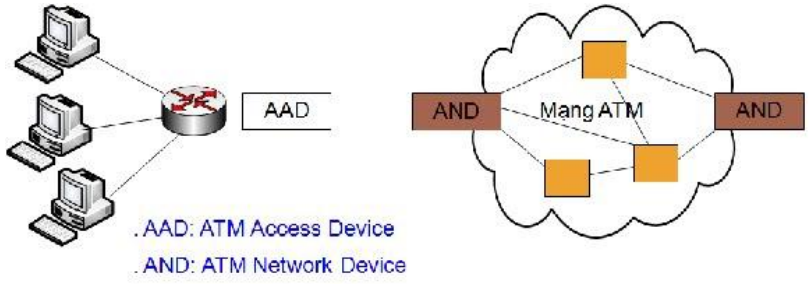
```
Router#ping 192.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/17/60 ms
Router#ping 192.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/19/56 ms
```

BÀI 7: ASYNCHRONOUS TRANSFER MODE

7.1 Yêu Cầu

ATM(Asynchronous Transfer Mode) là công nghệ chuyển mạch gói tương thích với mọi loại hình dịch vụ hiện nay. Nó được dùng trong cả mạng truy nhập lẫn mạng lõi.

Dữ liệu cần gởi được chia thành các gói có độ dài cố định là 53 bytes, được gọi là một tế bào (cell).

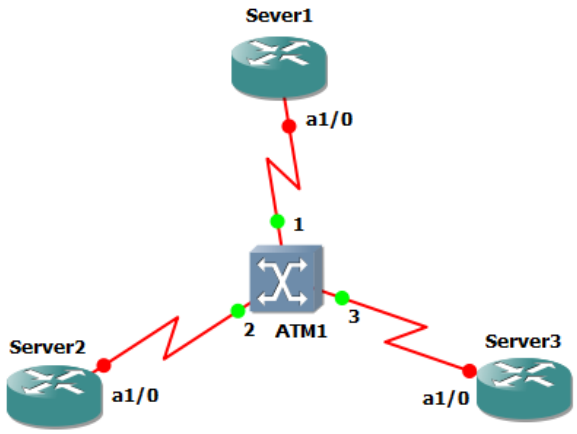


Chuyển mạch ATM

- VPI (Virtual Path Indentifier): nhận dạng đường ảo, dùng để phân biệt đường truyền nào trong số các đường nối tới một nút.
- VCI (Virtual Channel Indentifier): nhận dạng kênh ảo, dùng để phân biệt kênh nào được dùng trong đường truyền trên.
- PT (Payload Type): phân biệt dữ liệu của dịch vụ hay người dùng mà được đóng gói trong cell ATM đang gửi.
- HEC (Header Error Check): Dùng CRC kiểm tra lỗi bit của trường header
- Tại mỗi nút ATM sẽ dựa vào 2 trường VPI và VCI để chuyển mạch gói tin.
- Chỉ thực hiện kiểm tra lỗi của header nên tốc độ rất nhanh

- Vì cấu trúc của 1 cell là cố định 53 bytes nên có thể thiết kế hệ thống chuyển mạch ngay trong thiết bị phần cứng chứ không cần dùng phần mềm như các công nghệ chuyển mạch gói khác. Điều này làm tăng đáng kể tốc độ chuyển mạch
- Đảm bảo chất lượng dịch vụ thông qua việc thiết lập các kênh ảo thường trực PVC (Permanent Virtual Channel) ưu tiên để cấp băng thông cho từng loại dịch vụ hay thông qua thõa thuận với người dùng.

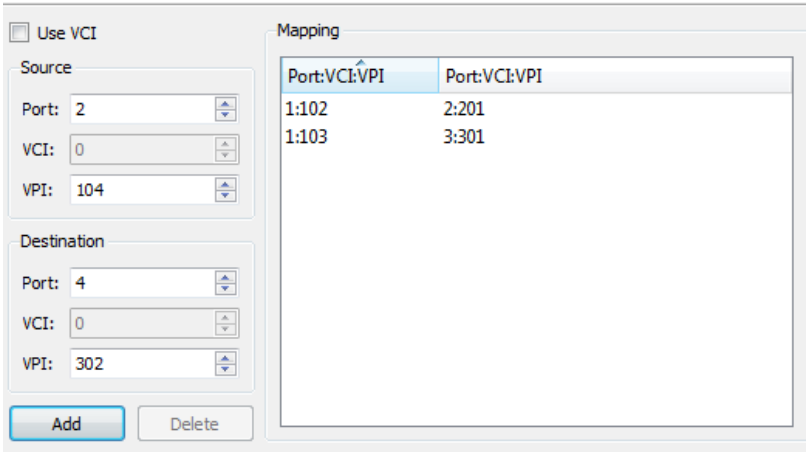
Yêu cầu: Xây dựng mô hình truyền dữ liệu bằng công nghệ chuyển mạch ATM, kết nói các mạng lỗi lại với nhau.



7.2 Hướng Dẫn

Thực hiện cấu hình ATM Switch theo sơ đồ map VCI như sau:

- Port 1 VCI 102 → Port 2 VCI 201
- Port 1 VCI 103 → Port 3 VCI 301



- Start các thiết bị và cấu hình từng thiết bị như sau:

- Cấu hình tại Router S1:

```
R1(config)#interface atm 1/0
R1(config-if)#no shutdown
R1(config-if)#do sho ip int br
R1(config-if)#exit
R1(config)#interface atm 1/0.123 multipoint
R1(config-subif)#ip address 192.168.123.1 255.255.255.0
R1(config-subif)#pvc 0/102
R1(config-if-atm-vc)#encapsulation aal5snap
R1(config-if-atm-vc)#protocol ip 192.168.123.2 broadcast
R1(config-if-atm-vc)#exit
R1(config-subif)#pvc 0/103
R1(config-if-atm-vc)#encapsulation aal5snap
R1(config-if-atm-vc)#protocol ip 192.168.123.3 broadcast
R1(config-if-atm-vc)#exit
R1(config-subif)#exit
R1(config)#do show run
R1(config)#exit
R1#write
```

- Cấu hình tại Router S2:

```
R1(config)#interface atm 1/0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface atm 1/0.123 point-to-point
R1(config-subif)#ip address 192.168.123.2 255.255.255.0
R1(config-subif)#pvc 0/201
R1(config-if-atm-vc)#encapsulation aal5snap
R1(config-if-atm-vc)#exit
R1(config-subif)#exit
R1#write
```

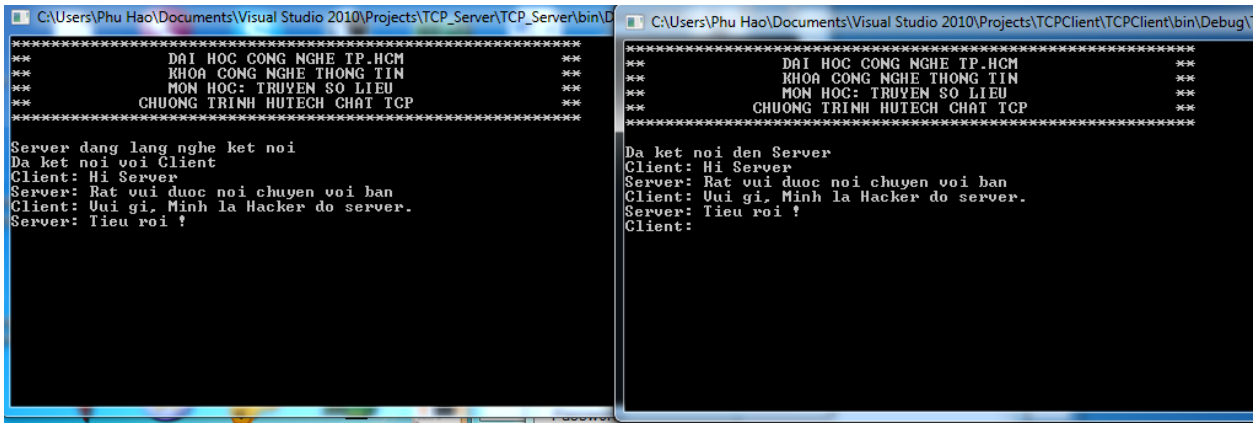
- Cấu hình tại Router S3:

```
R1(config)#interface atm 1/0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface atm 1/0.123 point-to-point
R1(config-subif)#ip address 192.168.123.3 255.255.255.0
R1(config-subif)#pvc 0/301
R1(config-if-atm-vc)#encapsulation aal5snap
R1(config-if-atm-vc)#exit
R1(config-subif)#exit
R1#write
```

BÀI 8: LẬP TRÌNH TCP

8.1 Yêu cầu

Viết chương trình chat bằng console C# với Client và Server.



8.2 Server

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;


namespace TCP_Server
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("************************************************************");
      Console.WriteLine("**       DAI HOC CONG NGHE TP.HCM         **");
      Console.WriteLine("**       KHOA CONG NGHE THONG TIN         **");
      Console.WriteLine("**       MON HOC: TRUYEN SO LIEU          **");
      Console.WriteLine("**      CHUONG TRINH HUTECH CHAT TCP         **");
      Console.WriteLine("************************************************************");
      Console.WriteLine("");
      //tao server
      IPEndPoint ip = new IPEndPoint (IPAddress.Parse("127.0.0.1"),8080);
      TcpListener tcplistener = new TcpListener(ip);
```

```
      //bật server và lắng nghe kết nối
      tcplistener.Start();
      Console.WriteLine("Server dang lang nghe ket noi");

      //chap nhan ket noi
      TcpClient tcpclient = tcplistener.AcceptTcpClient();
      Console.WriteLine("Da ket noi voi Client");

      //tao vung dem xuat nhap
      StreamReader sr = new StreamReader(tcpclient.GetStream());
      StreamWriter sw = new StreamWriter(tcpclient.GetStream());

      while (true)
      {
        //lay data tu client
        string dataReceive = sr.ReadLine();
        Console.WriteLine("Client: " + dataReceive);

        //Gui data den Client
        Console.Write("Server: ");
        string datasend = Console.ReadLine();

        sw.WriteLine(datasend);
        sw.Flush();
      }
      sr.Close();
      sw.Close();
      tcpclient.Close();
    }
  }
}
```

8.3 Client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace TCPClient
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("************************************************************");
      Console.WriteLine("**       DAI HOC CONG NGHE TP.HCM         **");
      Console.WriteLine("**       KHOA CONG NGHE THONG TIN         **");
      Console.WriteLine("**       MON HOC: TRUYEN SO LIEU          **");
      Console.WriteLine("**      CHUONG TRINH HUTECH CHAT TCP         **");
      Console.WriteLine("************************************************************");
      Console.WriteLine("");
      //tao mot Client
      IPEndPoint ipendpoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8080);
      TcpClient tcpclient = new TcpClient();

      //ket noi den server
```

```
        tcpclient.Connect(ipendpoint);
        Console.WriteLine("Da ket noi den Server");
        //tao vung dem nhap xuat
        StreamReader sr = new StreamReader(tcpclient.GetStream());
        StreamWriter sw = new StreamWriter(tcpclient.GetStream());

        while (true)
        {
          //gui du lieu den server
          Console.Write("Client: ");
          string datasend = Console.ReadLine();
          sw.WriteLine(datasend);
          sw.Flush();
          //nhan du lieu
          string dataReceive = sr.ReadLine();
          Console.WriteLine("Server: " + dataReceive);
        }
        sr.Close();
        sw.Close();
        tcpclient.Close();
      }
    }
}
```

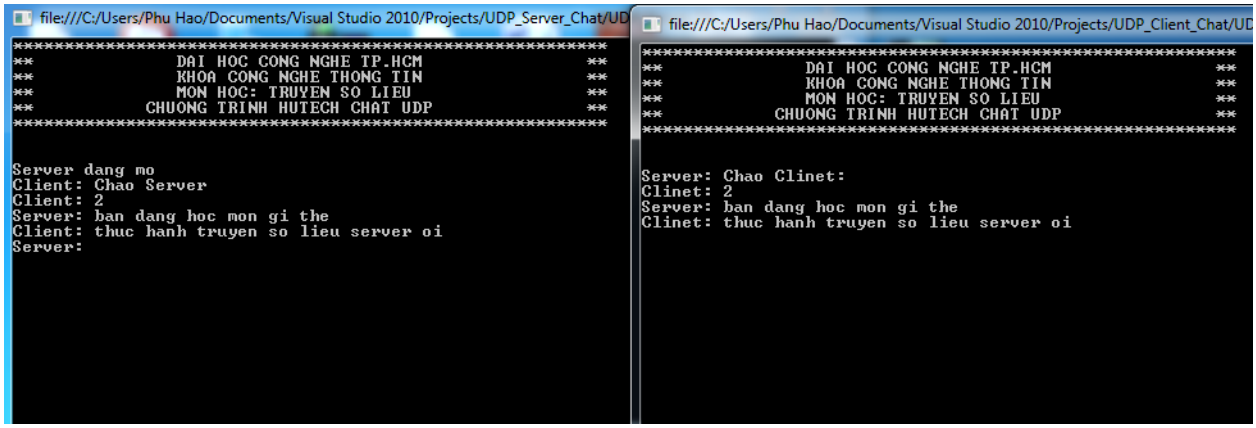
CÂU HỎI VÀ BÀI TẬP

1. Địa chỉ IP 127.0.0.1 là gì ?
2. Thay địa chỉ ip trên thành địa chỉ IP thật trên máy đang thực hành.
3. Dùng wireshark tiến hành bắt gói tin, kiểm tra giao thức được sử dụng trong chương trình. Tìm gói tin Data bắt được.

BÀI 9: LẬP TRÌNH UDP

9.1 Yêu Cầu

Viết chương trình truyền và nhận tin nhắn thông qua giao thức vận chuyển UDP.



Thay địa chỉ ip 172.0.0.1 thành địa chỉ ip trên máy thực hành.

Dùng wireshark bắt và phân tích gói tin UDP.

9.2 Server

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using System.Net;

namespace UDP_Server_Chat
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("**********************************************************");
      Console.WriteLine("**       DAI HOC CONG NGHE TP.HCM         **");
      Console.WriteLine("**       KHOA CONG NGHE THONG TIN         **");
      Console.WriteLine("**       MON HOC: TRUYEN SO LIEU          **");
      Console.WriteLine("**      CHUONG TRINH HUTECH CHAT UDP         **");
      Console.WriteLine("**********************************************************");
      Console.WriteLine("\n");
      //tạo server
```

```
      Socket socketserver = new Socket (AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);

      //tạo một IP Client
      IPEndPoint ipendport = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9999);
      socketserver.Bind(ipendport);

      //tạo ra một EndPoint tu xa de nhan data ve.
      IPEndPoint remoteipendport = new IPEndPoint(IPAddress.Any, 0);
      EndPoint remoteendport = (EndPoint)remoteipendport;
      Console.WriteLine("Server dang mo");

      //nhan du lieu tu Client
      byte[] dataarr = new byte[1024];
      int length = socketserver.ReceiveFrom(dataarr, ref remoteendport);

      string data = Encoding.ASCII.GetString(dataarr, 0, length);
      Console.WriteLine("Client: {0}", data);

      //Gui data cho Client
      dataarr = Encoding.ASCII.GetBytes("Chao Clinet: ");
      socketserver.SendTo(dataarr, remoteendport);

      //nhan du lieu

      while (true)
      {
        //nhận data
        dataarr = new Byte[1024];
        length = socketserver.ReceiveFrom(dataarr, ref remoteendport);
        data = Encoding.ASCII.GetString(dataarr, 0, length);
        if (data.ToUpper().Equals("EXIT"))
          break;
        Console.WriteLine("Client: {0}", data);
        //gui data
        Console.Write("Server: ");
        data = Console.ReadLine();

        dataarr = Encoding.ASCII.GetBytes(data);
        socketserver.SendTo(dataarr, dataarr.Length, SocketFlags.None, remoteendport);
      }
      //dong ket noi
      socketserver.Close();

    }
  }
}
```

9.3 Client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace UDP_Client_Chat
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("**********************************************************");
      Console.WriteLine("**       DAI HOC CONG NGHE TP.HCM        **");
      Console.WriteLine("**       KHOA CONG NGHE THONG TIN        **");
      Console.WriteLine("**       MON HOC: TRUYEN SO LIEU         **");
      Console.WriteLine("**      CHUONG TRINH HUTECH CHAT UDP        **");
      Console.WriteLine("**********************************************************");
      Console.WriteLine("\n");

      //tao socket
      Socket socketclient = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);

      //xac dinh dia chi ip server
      IPEndPoint ipendport = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9999);
      //gui den server
      string data = "Chao Server";
      byte[] dataarr = new byte[1024];
      dataarr = Encoding.ASCII.GetBytes(data);
      socketclient.SendTo(dataarr, ipendport);

      //nhan du lieu
      EndPoint remoteendpoint = (EndPoint)ipendport;
      dataarr = new byte[1024];
      int length = socketclient.ReceiveFrom(dataarr, ref remoteendpoint);
      data = Encoding.ASCII.GetString(dataarr, 0, length);
      Console.WriteLine("Server: {0}", data);
      while (true)
      {
        //gui cho server
        Console.Write("Clinet: ");
        data = Console.ReadLine();
        dataarr = Encoding.ASCII.GetBytes(data);
        socketclient.SendTo(dataarr, data.Length, SocketFlags.None, remoteendpoint);
        if (data.ToUpper().Equals("ECIT"))
          break;
        dataarr = new byte[1024];
        length = socketclient.ReceiveFrom(dataarr, ref remoteendpoint);
        data = Encoding.ASCII.GetString(dataarr, 0, length);
        Console.WriteLine("Server: {0}", data);
      }
      // dong ket noi
      socketclient.Close();
    }
  }
}
```

CÂU HỎI VÀ BÀI TẬP

1. Địa chỉ IP 127.0.0.1 là gì ?
2. Thay địa chỉ ip trên thành địa chỉ IP thật trên máy đang thực hành.
3. Dùng wireshark tiến hành bắt gói tin, kiểm tra giao thức được sử dụng trong chương trình. Tìm gói tin Data bắt được.