Lời Mở Đầu

Trong quá trình công nhgiệp hóa- hiện đại hóa đất nước. Tự động hóa là yếu tố không thể thiếu trong một nền công nghiệp hiện đại. Nói đến tự động hóa thì máy tính là một công cụ hỗ trợ đắc lực nhất và không thể thiếu được trong rất nhiều lĩnh vực, đặc biệt trong đo lường và điều khiển.

Việc ứng dụng máy tính vào kỹ thuật đo lường và điều khiển đã đem lại nhiều kết quả đầy tính ưu việt. Các thiết bị, hệ thống đo lường và điều khiển ghép nối với máy tính có độ chính xác cao, thời gian thu thập dữ liệu ngắn. Nhưng điều đáng quan tâm nhất là mức độ tự động hóa trong việc thu thập và xử lý kết quả đo, kể cả việc lập bảng thống kê, đồ họa, cũng như in ra kết quả. Để đo lường và điều khiển hệ thống thì ngoài các thiết bị ghép nối với máy tính, còn có Smột chương trình nạp vào máy tính để xử lý và điều khiển quá trình hoạt động của hệ thống.

Việc ứng dụng máy tính vào trong các hệ thống truyền động điều khiển tốc độ, vị trí ngày càng phổ biến. Ví dụ như trong các dây truyền lắp ráp các sản phẩm kỹ thuật cao, trong việc gia công sản phẩm có hình dạng, kích thước được vẽ trước trên máy tính, trong cơ cấu truyền động cho tay máy, người máy, cơ cấu ăn dao máy cắt gọt kim loại quay anten, kính viễn vọng, trong các hệ thống bám, tùy động,…

LabVIEW là một ngôn ngữ lập trình chuyên nghiệp trong lĩnh vực tự động hóa, là một môi trường lập trình cho phép tạo ra các chương trình sử dụng kí hiệu đồ họa giúp tạo lên những giao diện chương trình chuyên nghiệp. Nó chứa đựng rất nhiều khả năng, sức mạnh khi phát triển và thực thi các ứng dụng tự động hóa: đo lường, thu thập, phân tích, xử lí dữ liệu... Thế giới thiết bị ảo của labVIEW rất gần gũi và liên kết chặt chẽ với thế giới điều khiển tự động thực.

CHƯƠNG 1:

TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH LAVIEW

1.1 Giới thiệu về LABVIEW

LABVIEW (viết tắt của nhóm từ ***Laboratory Virtual Instrumentation Engineering Workbench***) là một phần mềm máy tính được phát triển bởi công ty National Instruments, Hoa kỳ. LABVIEW còn được biết đến như là một ngôn ngữ lập trình với khái niệm hoàn toàn khác so với các ngôn ngữ lập trình truyền thống như ngôn ngữ C, Pascal. Bằng cách diễn đạt cú pháp thông qua các hình ảnh trực quan trong môi trường soạn thảo, LABVIEW đã được gọi với tên khác là lập trình G (viết tắt của *Graphical*, nghĩa là đồ họa).

- LABVIEW (Virtual Instrument Engineering Workbench) là một môi trường phát triển dựa trên ngôn ngữ lập trình đồ hoạ, thường được sử dụng cho mục đích đo lường, kiểm tra, đánh giá, xử lý, điều khiển các tham số của thiết bị.

- LABVIEW là một ngôn ngữ lập trình đa năng, giống như các ngôn ngữ lập trình hiện đại khác. LABVIEW gồm có các thư viện thu nhận dữ liệu, một loạt các thiết bị điều khiển, phân tích dữ liệu, biểu diễn và lưu trữ dữ liệu. Nó còn có các công cụ phát triển được thiết kế riêng cho việc nối ghép và điều khiển thiết bị.

- LABVIEW khác với các ngôn ngữ lập trình thông thường ở điểm cơ bản là: các ngôn ngữ lập trình khác thường dùng trên cơ chế dòng lệnh, trong khi đó LABVIEW dùng ngôn ngữ lập trình Graphical để tạo ra các chương trình ở dạng sơ đồ khối.

- Trong LABVIEW ta xây dựng giao diện người sử dụng bằng việc thiết lập các công cụ và các đối tượng. Giao diện người sử dụng được hiểu như là một front panel rồi sau đó ta đưa code vào trong sơ đồ khối để điều khiển các đối tượng ở trên front panel. Sơ đồ khối cũng có thể hiểu giống như một lưu đồ thuật toán.

- LABVIEW được tích hợp đầy đủ các chức năng giao tiếp với các phần cứng GPIB, VXI, PXI, RS-232, RS-485, các thiết bị thu nhận dữ liệu. LABVIEW cũng xây dựng các đặc trưng cho việc kết nối các ứng dụng của ta với Web sử dụng LABVIEW Web Server và, chuẩn mạng TCP/IP và Active X.

- LABVIEW được dùng nhiều trong các phòng thí nghiệm, lĩnh vực khoa học kỹ thuật như tự động hóa, điều khiển, điện tử, cơ điện tử, hàng không, hóa sinh, điện tử y sinh,... Hiện tại ngoài phiên bản LABVIEW cho các hệ điều hành Windows, Linux, Hãng NI đã phát triển các mô-đun LABVIEW cho máy hỗ trợ cá nhân (PDA).

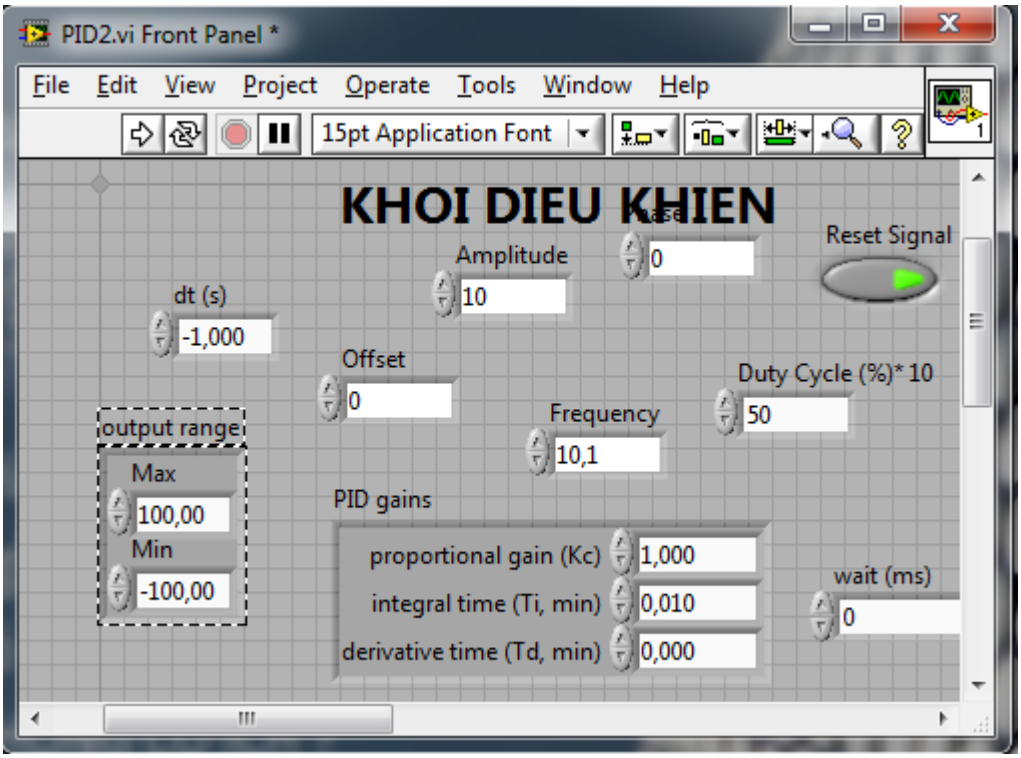
Các chức năng chính của LABVIEW có thể tóm tắt như sau:

- Thu thập tín hiệu từ các thiết bị bên ngoài như cảm biến nhiệt độ, hình ảnh từ webcam, vận tốc của động cơ, ...
- Giao tiếp với các thiết bị ngoại vi thông qua nhiều chuẩn giao tiếp thông qua các cổng giao tiếp: RS232, RS485, USB, PCI, Ethernet
- Mô phỏng và xử lý các tín hiệu thu nhận được để phục vụ các mục đích nghiên cứu hay mục đích của hệ thống mà người lập trình mong muốn
- Xây dựng các giao diện người dùng một cách nhanh chóng và thẩm mỹ hơn nhiều so với các ngôn ngữ khác như Visual Basic, Matlab,..
- Cho phép thực hiện các thuật toán điều khiển như PID, Logic mờ (*Fuzzy Logic*), một cách nhanh chóng thông qua các chức năng tích hợp sẵn trong LABVIEW.
- Cho phép kết hợp với nhiều ngôn ngữ lập trình truyền thống như C, C++, ...

1.2 Giao diện của LABVIEW

a) *Front panel:*

Là giao diện của người sử dụng. Ví dụ sau đây minh họa front panel.



Hình 1 Front panel

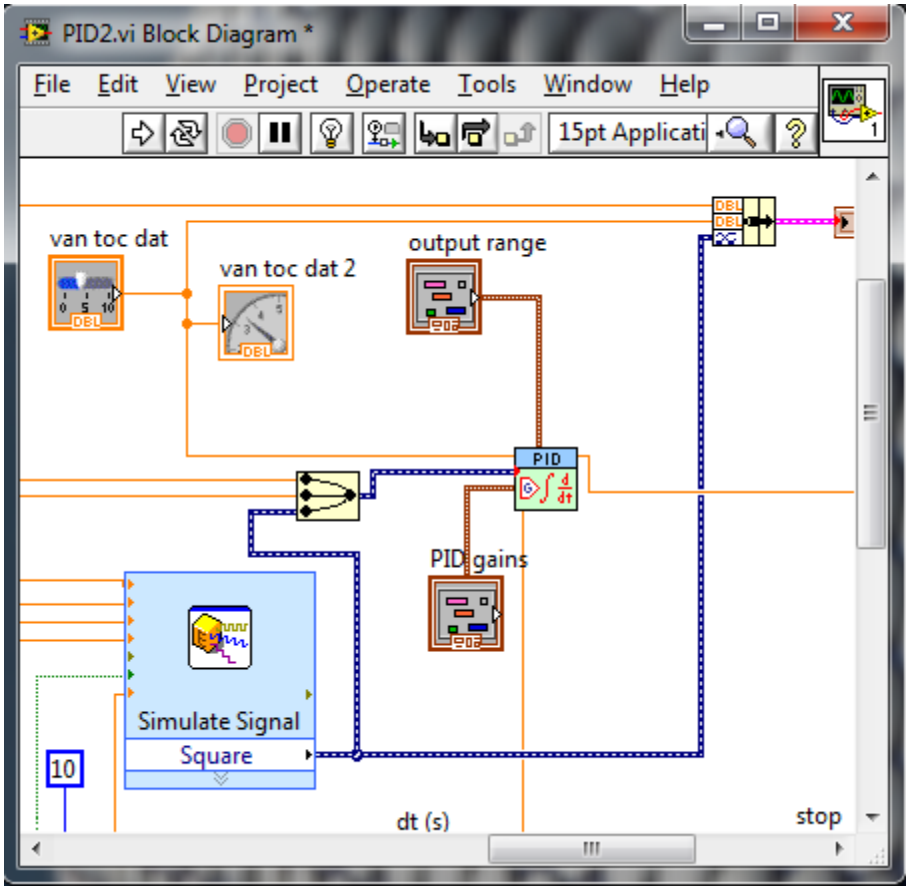
- Xây dựng front panel với các bộ điều khiển (controls) và các hiển thị (Indicators), chúng được sử dụng với các chức năng vào ra dữ liệu. Các điều khiển bao gồm các núm (knobs), nút ấn (push buttons), mặt đồng hồ và các thiết bị vào dữ liệu khác. Control là các đối tượng được đặt trên Front Panel để cung cấp dữ liệu cho chương trình. Nó tương tự như đầu vào cung cấp dữ liệu.

b) *Block Diagram:*

Là sơ đồ khối :Block Diagram của 1 VI là một sơ đồ được xây dựng trên môi trường LABVIEW, nó có thể gồm nhiều đối tượng và các hàm khác nhau để tạo các câu lệnh để chương trình thực hiện. Block Diagram là một mã nguồn đồ hoạ

của 1 VI. Các đối tượng trên Front Panel được thể hiện bằng các thiết bị đầu cuối trên Block Diagram, không thể loại bỏ các thiết bị đầu cuối trên Block

Diagram. Các thiết bị đầu cuối chỉ mất đi sau khi loại bỏ đối tượng tương ứng trên Front panel.



Hình 2 Block Diagram

Cấu trúc của một Block Diagram gồm các thiết bị đầu cuối (Terminal), Nút (Node) và các dây nối (wire).

-Terminal: là các cổng mà dữ liệu truyền qua giữa Block Diagram và Front panel, và giữa các Node trong Block Diagram. Các Terminal nằm ở dưới dạng các Icon của các Function.

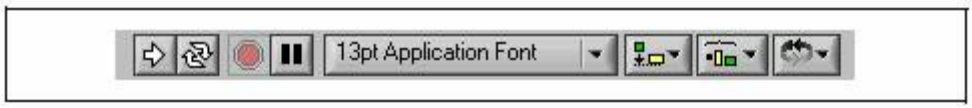
-Nodes: là các phần tử thực hiện chương trình, chúng tương tự như các mệnh đề, toán tử, hàm và các chương trình con trong các ngôn ngữ lập trình thông thường.

-Wires: là các dây nối dữ liệu giữa các node

1.3 Các Thanh công cụ

a)Thanh công cụ front panel:

Sử dụng các nút ấn của thanh công cụ dùng để chạy và tạo ra một VI. Thanh công cụ xuất hiện trên front panel có dạng như sau:



Hình 3 Thanh công cụ trên Front panel

Trong đó:

 Kích nút **Run** dùng để chạy một VI, trong lúc VI chạy thì trạng thái nút ấn thay đổi theo nếu VI không có lỗi gì thì trạng thái có dạng như sau: 

 Khi nút ấn **Run** ở trạng thái này thì có nghĩa VI của ta đang bị lỗi nào đó mà ta cần phải xử lý. Để tìm lỗi ta kích đúp vào nút này để hiển thị danh sách toàn bộ các lỗi trong VI của ta.

 Kích vào nút **Run Continuously** để chạy VI liên tục cho đến khi muốn hủy hay dừng lại. Ta cũng có thể ấn tiếp nút lệnh này để không cho phép chạy liên tục.

 Trong lúc VI chạy, nút hủy bỏ hoạt động xuất hiện và nếu ta ấn vào biểu tượng này thì chương trình đang chạy dừng ngay lập tức. Với một chú ý nên tránh dùng nút lệnh này để dừng một VI, bởi vì ta sẽ không biết trạng thái của VI. Ta nên thiết kế chương trình dừng VI ví dụ ta có thể sử dụng chuyển mạch ở front panel.

Kích vào nút lệnh **Pause** để tạm dừng chương trình VI đang chạy. Khi ta kích vào nút lệnh **Pause** thì LABVIEW sẽ làm sáng vị trí ta dừng hoạt động trong sơ đồ khối. Khi ta muốn chạy tiếp chương trình thì ta ấn lại nút lệnh này.

Dùng để thiết lập font cho VI bao gồm kích thước, kiểu loại, màu sắc.

Dùng để sắp xếp các đối tượng thẳng hàng nhau bao gồm các đường thẳng đứng, mép trên, trái …

Dùng để phân bổ các đối tượng …

Lựa chọn **Reorder** khi ta có các đối tượng gối lên nhau và ta muốn định nghĩa đối tượng là đứng trước hay đứng sau. Việc lựa chọn một trong các đối tượng với việc định vị vị trí của nó rồi sau đó di chuyển lên phía trước hay di chuyển về phía sau…

b)Thanh công cụ Block Diagram

Hình 4 Thanh công cụ Block Diagram

Kích vào nút lệnh **Highlight Execution** ta sẽ thấy luồng dữ liệu chạy trong sơ đồ khối. Khi ta kích lại nút lệnh này quá trình sẽ bị dừng lại.

Kích vào nút lệnh **Step into** dùng để lặp từng bước một trong vòng lặp, subVI.

Kích vào nút lệnh **Step over** dùng để bỏ qua một vòng lặp hoặc một subVI.

 Kích vào nút lệnh **Step out** dùng để nhảy ra ngoài vòng lặp hoặc subVI.

c. Bảng điều khiển Palettes

Việc lập trình trên LABVIEW cần sử dụng các bảng: Tools Palette, Controls Palette, Functions Palette, các bảng đó cung cấp các chức năng để người sử dụng có thể tạo và thay đổi trên giao diện Front Panel và Block Diagram.

Tools Palettes.

Tools Palettes xuất hiện trên cả Front Panel và Block Diagram. Bảng này cho phép người sử dụng có thể xác lập các chế độ làm việc đặc biệt của con trỏ chuột. Khi lựa chọn một công cụ, biểu tượng của con trỏ sẽ được thay đổi theo biểu tượng của công cụ đó

Nếu thiết lập chế độ tự động lựa chọn công cụ và người sử dụng di chuyển con trỏ qua các đối tượng trên Front Panel hoặc Block Diagram, LABVIEW sẽ tự động lựa chọn công cụ phù hợp trên bảng Tools Palette

Để truy cập vào **Tools palette** ta chọn Menu: **View→Tools palette**. Các công cụ trong **Tools palette** gồm có:

Hình 5 Tool panel

 Operating tool: Dùng để thay đổi giá trị điều khiển hoặc lựa chọn văn bản trong điều khiển.

 Positioning tool: Dùng để lựa chọn, di chuyển, thay đổi các kích thước đối tượng.

 Labeling tool: Dùng để soạn thảo văn bản dạng text và tạo ra các nhãn.

 Wiring tool: Dùng để nối các đối tượng lại với nhau trong sơ đồ khối.

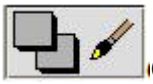
 Object shortcut menu: Dùng để truy cập vào một đối tượng bằng cách kích chuột trái.

 Scrolling tool

 Breakpoint tool: Dùng để thiết lập điểm dừng trên các VI, functions, nút, dây nối, các cấu trúc lệnh để dừng hoạt động ở tại vị trí này.

 Probe tool: Dùng để tạo ra những đầu dò trên các dây nối trong sơ đồ khối. Việc sử dụng Probe tool dùng để kiểm tra các giá trị trung gian trong VI.

 Color copy tool: Dùng để copy các màu cho việc paste bằng việc sử dụng Coloring tool.

 Coloring tool: Dùng để tô màu cho một đối tượng. Nó cũng có thể hiển thị ngay việc thiết lập màu sắc mặt trước và mầu nền.

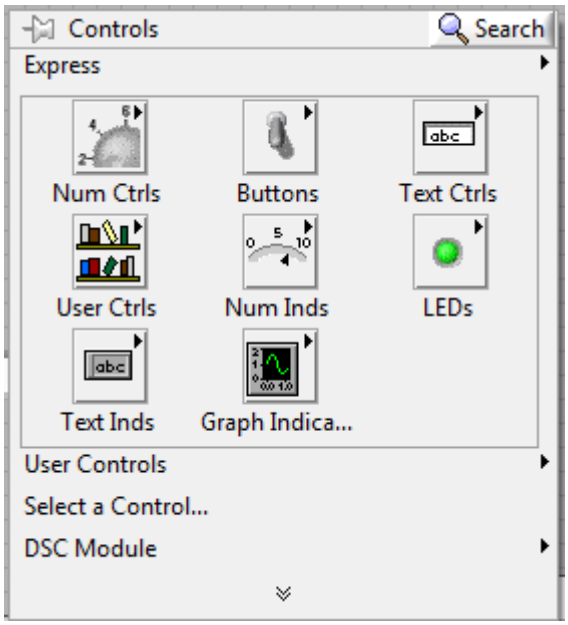
1.4 Các bảng điều khiển và các bảng chức năng.

Các bảng điều khiển và các bảng chức năng là các bảng của đối tượng được sử dụng để tạo ra các VI. Để sử dụng đối tượng trên bảng ta kích vào đối tượng vào đặt nó vào trong front panel hoặc là trong sơ đồ khối.

Sử dụng các nút chỉ dẫn trên bảng **Controls** và bảng **Functions** để xác định và tìm kiếm các điều khiển, các VI và các hàm. Ta cũng có thể làm bằng cách kích chuột phải vào biểu tượng VI ở trên bảng và chọn **Open VI** từ menu phím tắt để mở VI.

e. Bảng điều khiển (Controls Palette).

-Bảng điều khiển chỉ duy nhất xuất hiện trên Front panel. Bảng điều khiển chứa các bộ điều khiển (control) và các bộ hiển thị (Indicator). Bảng điều khiển được minh họa như hình dưới đây.



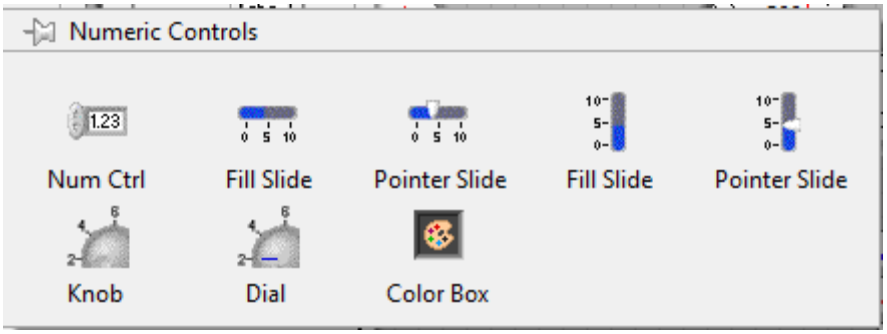
Hình 6 Bảng điều khiển

-Bảng điều khiển được sử dụng để người sử thiết kế cấu trúc mặt hiển thị gồm các thiết bị ví dụ: các công tắc, các loại đèn, các loại màn hình hiển thị…Với bảng điều khiển này, người sử dụng có thể chọn các bộ thiết bị chuẩn do hãng sản xuất cung cấp.

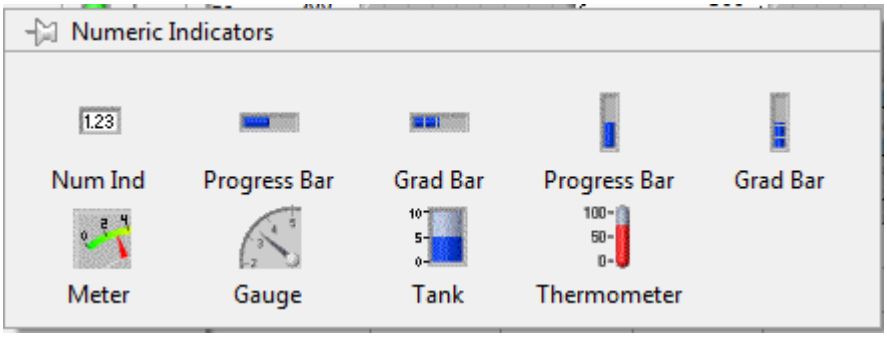
- Bảng điều khiển dùng để cung cấp dữ liệu đầu vào và hiển thị các kết quả đầu ra.

- Một số bộ điều khiển và hiển thị trên controls palette:

Numeric Controls/Indicators: Bộ công cụ này được sử dụng để hiển thị và điều khiển dữ liệu dạng số trong chương trình :



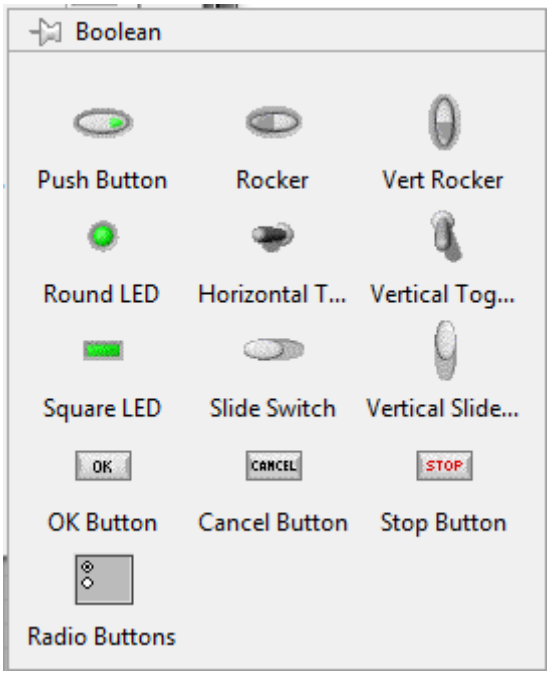
Hình 7 Numeric control



Hình 8 Numeric Indicatiors

Boolean Controls/Indicators:

Bộ công cụ này cung cấp 2 giá trị là True và False. Khi thực hiện chương trình người sử dụng sử dụng chuột để điều khiển giá trị của thiết bị. Việc thay đổi giá trị của các thiết bị chỉ có tác dụng khi các thiết bị đó được xác lập ở chế độ là các Control. Còn nếu ở chế độ là các Indicator thì giá trị không thay đổi vì chúng chỉ là các thiết bị hiển thị



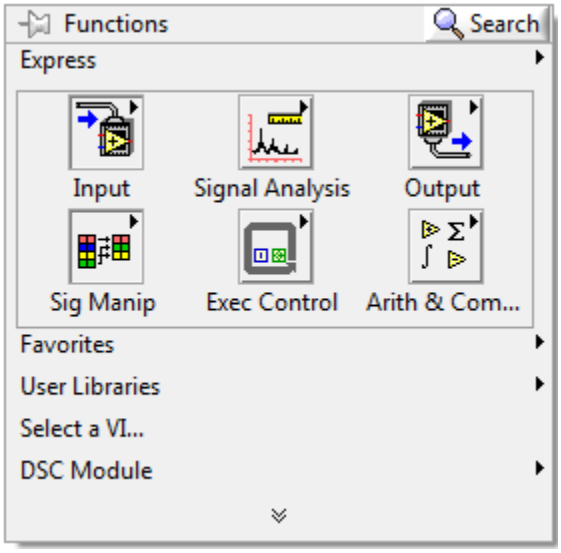
Hình 9 Boolean

String Controls/Indicators:

Các điều khiển này dùng để nhập và hiển thị các ký tự, nó cũng có thể được xác lập ở chế độ đầu vào hay đầu ra.

f) *Bảng chức năng (Functions Palette)*

Bảng Funtions palette chỉ xuất hiện trên Block Diagram. Bảng này chứa các VIs và các hàm mà người sử dụng dùng để xây dựng nên các khối lưu đồ. Bảng chức năng có dạng như sau:



Hình 10 Funtion

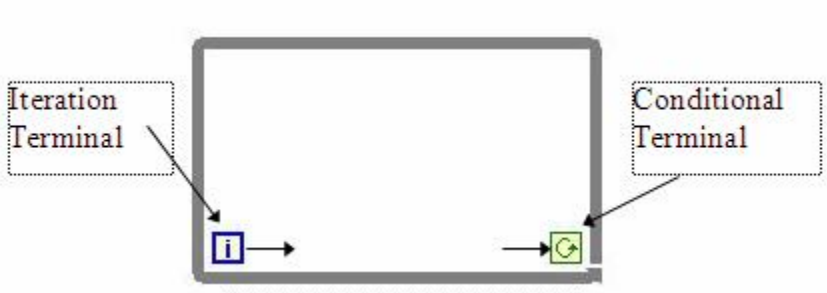
Với bảng Function Palette, người lập trình thực hiện các cú pháp ví dụ:phép lặp, phép lựa chọn thông qua các nhóm hàm, chức năng đã được cung cấp bên cạnh đó từ bảng này người sử dụng có thể tạo ra và sử dụng lại các hàm, chức năng mà người sử dụng tự xây dựng. Các hàm toán học được minh hoạ thông qua các biểu tượng. Khi muốn lựa chọn thực hiện một hàm nào đó thì người sử dụng chọn biểu tượng thể hiện cho hàm đó và có thể kéo thả ở bất kỳ vị trí nào trên Block Diagram sau đó xác định những đầu vào và đầu ra cần thiết.

1.5 Cấu trúc, hoạt động của vòng lặp.

a) While Loop

 Giống như vòng lặp Do Loop hoặc Repeat – Until Loop trong ngôn ngữ văn bản. Vòng lặp While Loop lµ một cấu trúc lặp thực hiện sơ đồ bên trong nó cho đến khi giá trị boolean đưa tới conditional terminal (một terminal đầu vào) là trùng với điều kiện được thiết lập để thực hiện vòng lặp. để truy cập while loop ta chọn menu: **functions→ structures → while loop.** Sau đó sử dụng con trỏ kích và kéo tạo ra vùng mong muốn mà ta muốn lặp.

Biểu tượng của while loop được minh hoạ ở hình dưới đây.



Hình 11 Vòng lặp While loop

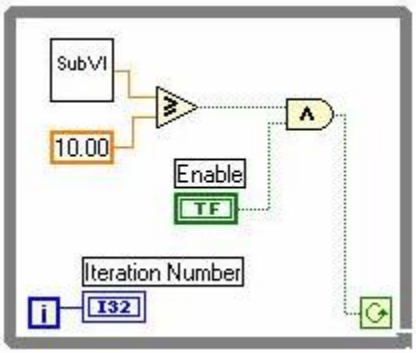
VI kiểm tra Conditional Terminal tại cuối mỗi vòng lặp, do đó While Loop luôn thực hiện ít nhất một lần. Iteration Terminal là một Terminal đầu ra mà đưa ra số lần vòng lặp thực hiện được. Việc tính số lần lặp luôn được bắt đầu từ 0. Vì vậy, nếu vòng lặp chạy một lần thì Iteration Terminal đưa ra kết quả 0. Việc thực hiện vòng lặp có thể được xác định thông qua Conditional Terminal. Tại Conditional Terminal,ta có thể chọn các điều kiện:

+ Stop if true.

+ Continue if true.

Việc xác định điều kiện để thực hiện vòng lặp tại Conditional Terminal rất quan trọng vì nếu không xác định đúng thì vòng lặp có thể rơi vào vòng lặp vô hạn.

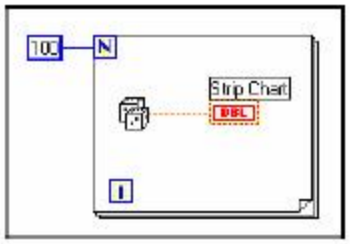
Trong sơ đồ khối sau hoạt động của vòng lặp While Loop hoạt động cho đến khi giá trị đầu ra từ subVI là bé hơn 10 hoặc điều khiển **Enable** là FASLE. Đầu ra của hàm AND là TRUE chỉ khi hai đầu vào là TRUE, ngoài ra khi một trong các đầu vào là FASLE đầu ra là FASLE.



Hình 12 While loop với điều kiện lặp

b) For loop

- Cấu trúc vòng lặp For Loop là quá trình thực hiện lặp trong sơ đồ khối với số vòng lặp xác định. Ta chọn vòng lặp For Loop từ vị trí **Functions -→Structures →For loop** và chứa đoạn mã mà ta muốn lặp nằm bên trong vòng for loop. Một vòng lặp For Loop là một hộp có kích thước nào đó bao gồm 2 terminal: count terminal (là đầu vào của terminal) và iteration terminal (đầu ra của terminal). Count terminal là số lần lặp. Iteration terminal chứa số lần lặp đã được thực hiện.
- Cấu trúc vòng lặp For Loop khác với cấu trúc vòng lặp While Loop ở chỗ vòng lặp For Loop hoạt động với số lần lặp xác định. Trong khi đó vòng lặp While Loop chỉ dừng quá trình lặp khi giá trị điều kiện được kiểm tra là đúng. Hoạt động của vòng lặp For Loop tương đương với đoạn mã sau:
- For i = 0 to N-1
- Bên trong sơ đồ khối thực hiện và lặp lại đến khi đến giá trị N-1.
- Ví dụ sau đây minh họa hoạt động của vòng lặp phát ra 100 số ngẫu nhiên và hiển thị lên đồ thị.

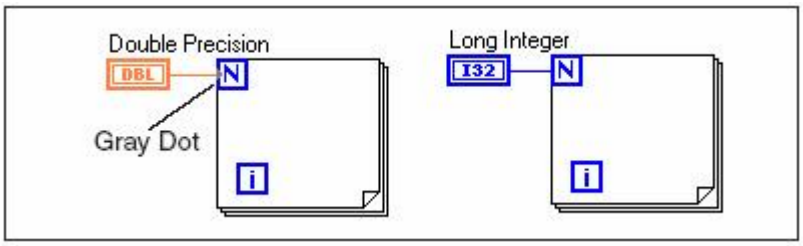


Hình 13 For loop

1.6 Chuyển đổi số.

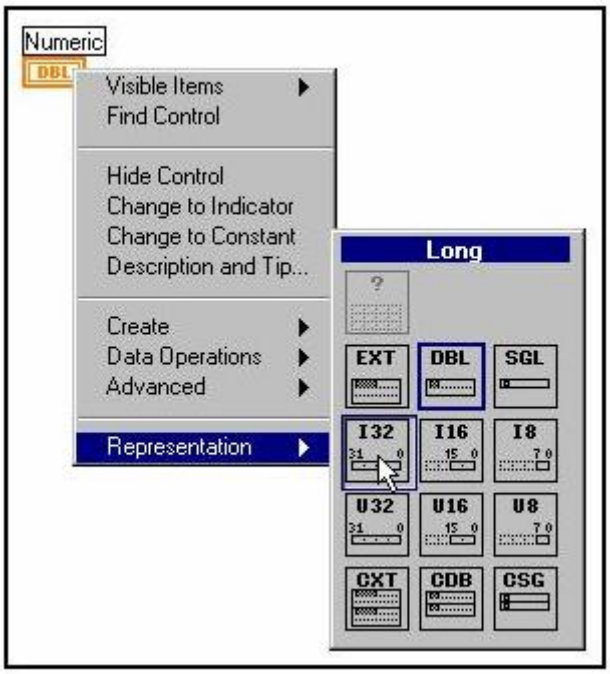
Trong LABVIEW biểu diễn các kiểu dữ liệu số như số nguyên (byte, word, long), kiểu số thực (single, double, kiểu mở rộng), kiểu số phức (single, double, kiểu mở rộng). Nếu ta nối 2 terminal với nhau thì 2 terminal này cần phải cùng kiểu dữ liệu nếu không nó sẽ không cho phép nối. Để nối 2 terminal khác kiểu loại dữ liệu thì LABVIEW sẽ thực hiện chuyển đổi dữ liệu của terminal này sang kiểu dữ liệu của terminal kia.

Ví dụ, chúng ta xem xét count terminal trong vòng lặp For Loop. Nó có kiểu số nguyên dài (long Integer). Nếu ta nối với count terminal một số thực LABVIEW sẽ biến đổi số thực này sang số nguyên.



Hình 14 Chuyển đổi dữ liệu trong For loop

Để thay đổi nội dung của một đối tượng, kích chuột phải vào đối tượng trên front panel hoặc từ sơ đồ khối và chọn **Representation** từ menu. Khi panel xuất hiện ta chọn kiểu số.

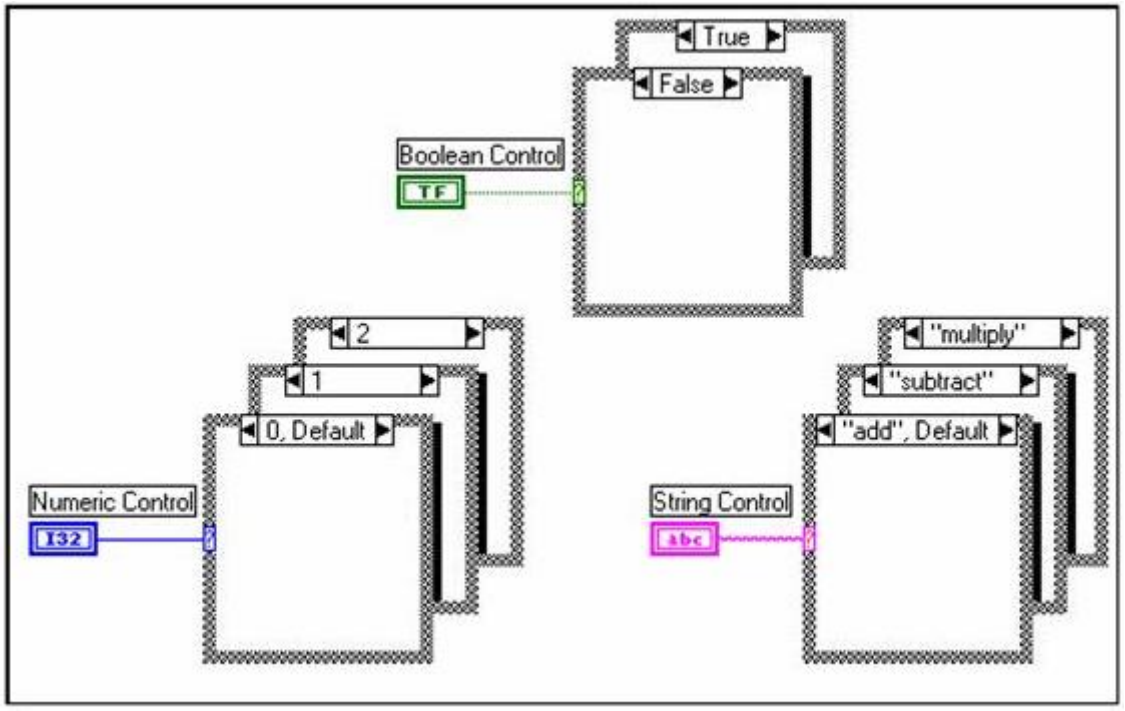


Hình 15 Cách thay đổi đối tượng

1.7 Hoạt động của cấu trúc lựa chọn.

Ta có thể đặt cấu trúc lựa chọn lên trên sơ đồ khối bằng việc chọn nó từ **Structure** từ **Functions.**

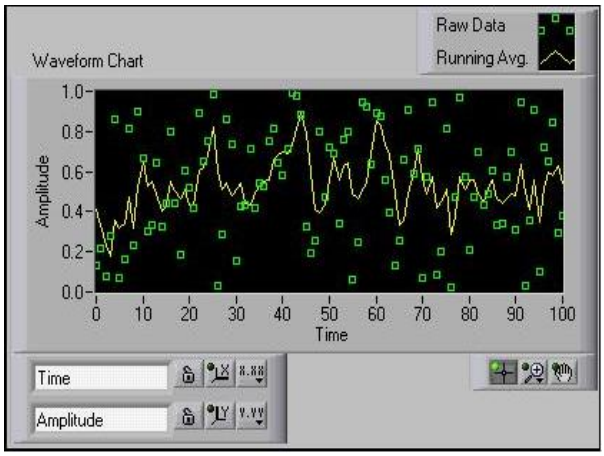
Cấu trúc lựa chọn giống như những câu lệnh *if ...then... else* trong ngôn ngữ lặp trình văn bản. Cấu trúc lựa chọn được cấu hình như sau: tại một thời điểm chỉ có một trường hợp là đúng. Mỗi một trường hợp tương ứng với một sơ đồ con (subdiagram). Chỉ có một trường hợp hoạt động, phụ thuộc vào giá trị nối tới chọn terminal (selector terminal). Selector terminal có thể là số, Boolean hoặc string. Nếu kiểu dữ liệu là Boolean, cấu trúc có thể là True hoặc Fasle. Nếu kiểu dữ liệu là số hoặc string cấu trúc có thể nhận $2^{31} - 1$ trường hợp.



Hình 16 Cấu trúc lựa chọn

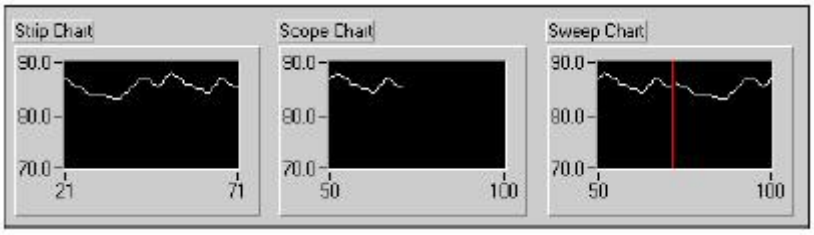
1.8 Các dạng biểu đồ sóng

Các biểu đồ dạng sóng là các bộ hiển thị số đặc biệt, nó biểu diễn một hoặc nhiều độ thị cùng một lúc. Các biểu đồ dạng sóng được lấy từ **Controls » Graph.** Hình dưới đây minh họa biểu đồ dạng sóng.



Hình 17 Front panel của sơ đồ dạng song

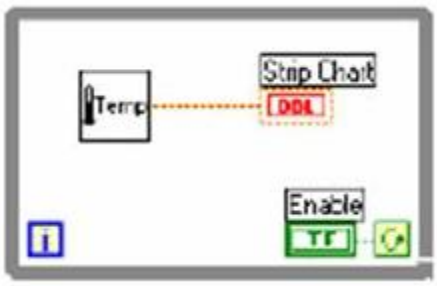
Các biểu đồ thông thường sử dụng 3 chế độ khác nhau để hiển thị dữ liệu, hình vẽ dưới đây mô tả 3 chế độ:

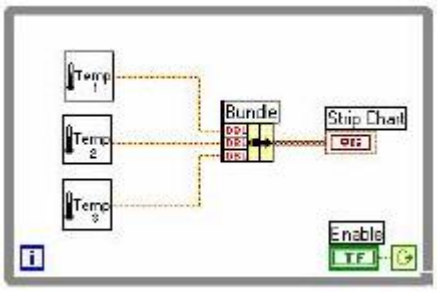


Hình 18 Các dạng hiển thị sóng

Đấu nối với biểu đồ dạng một tín hiệu

Ta có thể đấu nối trực tiếp đầu ra của VI với đầu vào của biểu đồ dạng sóng. Kiểu dữ liệu hiển thị trên biểu đồ dạng sóng này được xem như kiểu dữ liệu đầu vào có dạng như hình vẽ sau:





Hình 19 Đấu nối trực tiếp **Hình 20 Đấu nối nhiều tín hiệu tới biểu đồ**

Đấu nối với biểu đồ sóng nhiều tín hiệu

Khi ta muốn hiển thị nhiều dữ liệu lên một biểu đồ thì ta cần phải bó các dữ liệu bằng việc sử dụng hàm Bundle từ **Functions » Cluster » Bundle**. Trong sơ đồ khối dưới đây hàm Bundle bó hoặc nhóm tín hiệu đầu ra của 3 VI khác nhau. Các VI này nhận tín hiệu nhiệt độ và hiển thị nó lên đồ thị. Khi cần thêm hiển thị tín hiệu lên đồ thị thì ta cần phải thay đổi kích thước của hàm Bundle tăng lên theo số lượng đầu vào.

CHƯƠNG 2 :

ĐIỀU KHIỂN ĐỘNG CƠ DC SERVO.

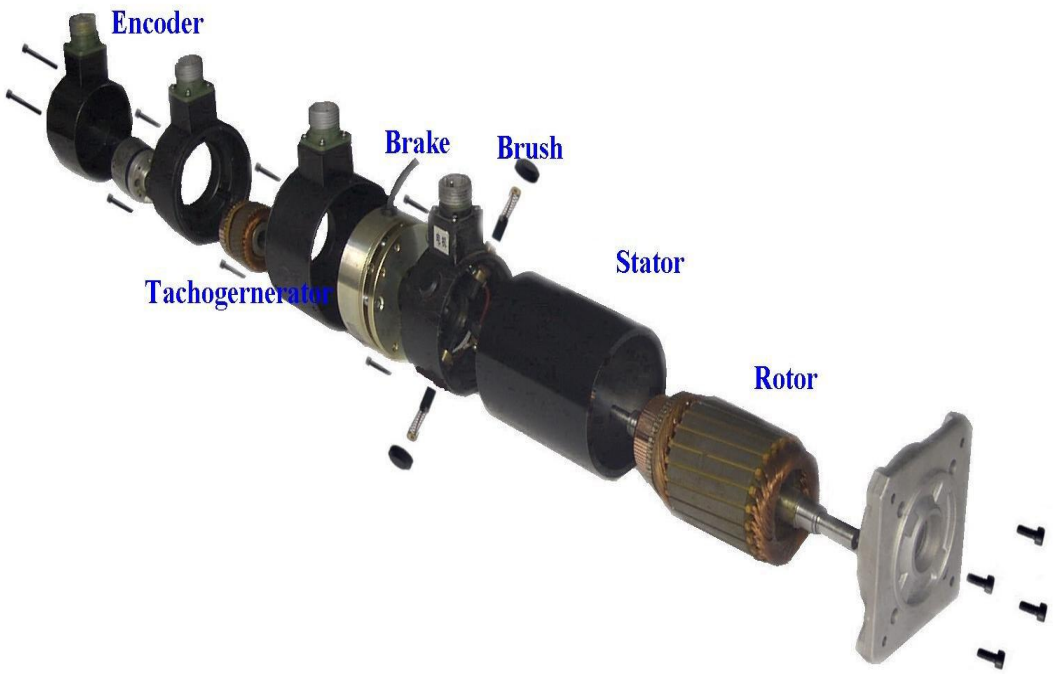
2.1 Cấu tạo động cơ DC SERVO.

Động cơ DC và động cơ bước vốn là những hệ hồi tiếp vòng hở - ta cấp điện để động cơ quay nhưng chúng quay bao nhiêu thì ta không biết, kể cả đối với động cơ bước là động cơ quay một góc xác định tùy vào số xung nhận được. Việc thiết lập một hệ thống điều khiển để xác định những gì ngăn cản chuyển động quay của động cơ hoặc làm động cơ không quay cũng không dễ dàng.

Hình 21 Động cơ DC Servo trong thực tế

Mặt khác, động cơ servo được thiết kế cho những hệ thống hồi tiếp vòng kín. Tín hiệu ra của động cơ được nối với một mạch điều khiển. Khi động cơ quay, vận tốc và vị trí sẽ được hồi tiếp về mạch điều khiển này. Nếu có bất kỳ lý do nào ngăn cản chuyển động quay của động cơ, cơ cấu hồi tiếp sẽ nhận thấy tín hiệu ra chưa đạt được vị trí mong muốn. Mạch điều khiển tiếp tục chỉnh sai lệch cho động cơ đạt được điểm chính xác.

Động cơ servo có nhiều kiểu dáng và kích thước, được sử dụng trong nhiều máy khác nhau, từ máy tiện điều khiển bằng máy tính cho đến các mô hình máy bay và xe hơi.



Hình 22 Các thành phần của động cơ DC Servo

Một động cơ DCservo tiêu biểu gồm có các thành phần chính sau:

- **Stator**: được gắn liền với vỏ động cơ
- **Rotor**: là thành phần tạo chuyển động quay
- **Chổi than và vành góp**: giúp đưa điện vào Rotor
- **Encoder:** hay còn gọi là bộ mã hóa vòng quay, phản hồi xung, đơn vị tính (xung/vòng)

Ngoài ra, DC servo còn có thể có thêm các thành phần sau:

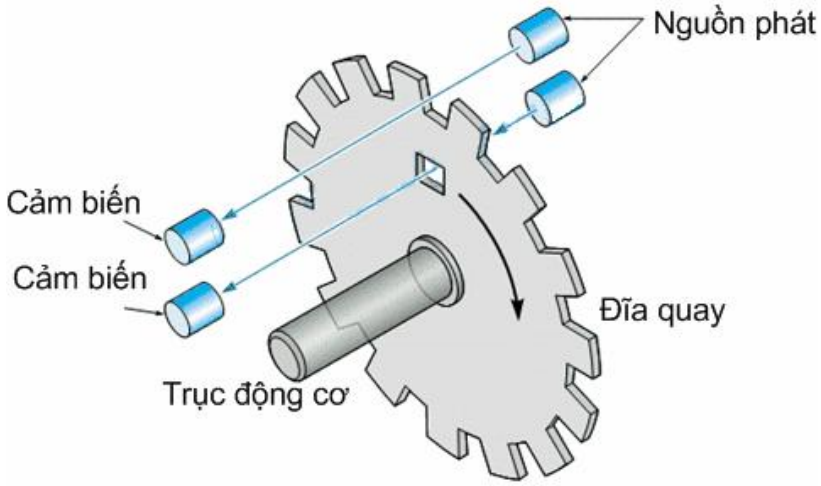
- **Phanh điện từ:** giúp hãm động cơ trong trường hợp cần thiết
- **Tachometer :** là thành phần phản hồi tương tự, thực chất là một máy phát điện nhỏ, với điện áp phàn hồi được tính bằng (vol/vòng quay)

2.2 Nguyên lý điều khiển động cơ DC SERVO.

Để điều khiển số vòng quay hay vận tốc động cơ thì chúng ta nhất thiết phải đọc được góc quay của motor.

Một số phương pháp có thể được dùng để xác định góc quay của motor bao gồm tachometer, hoặc dùng biến trở xoay, hoặc dùng encoder. Trong đó 2 phương pháp đầu tiên là phương pháp analog và dùng optiacal encoder (encoder quang) thuộc nhóm phương pháp digital.

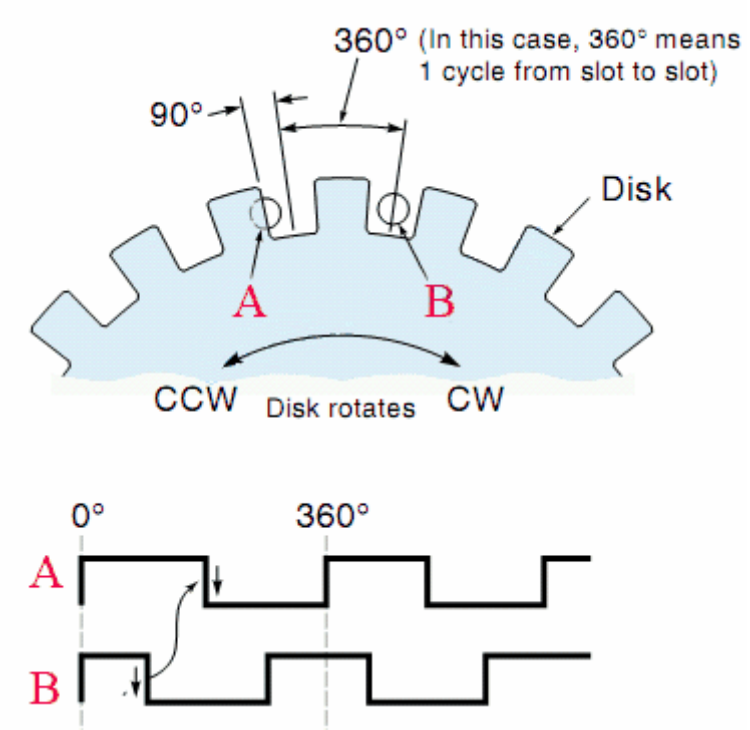
Hệ thống optical encoder bao gồm một nguồn phát quang (thường là hồng ngoại – infrared), một cảm biến quang và một đĩa có chia rãnh. Optical encoder lại được chia thành 2 loại: encoder tuyệt đối (absolute optical encoder) và encoder tương đối (incremental optical encoder). Trong đa số các DC Motor, incremental optical encoder được dùng đa số.



Hình 23 Cấu tạo một encoder quang.

Encoder thường có 3 kênh (3 ngõ ra) bao gồm kênh A, kênh B và kênh I (Index).

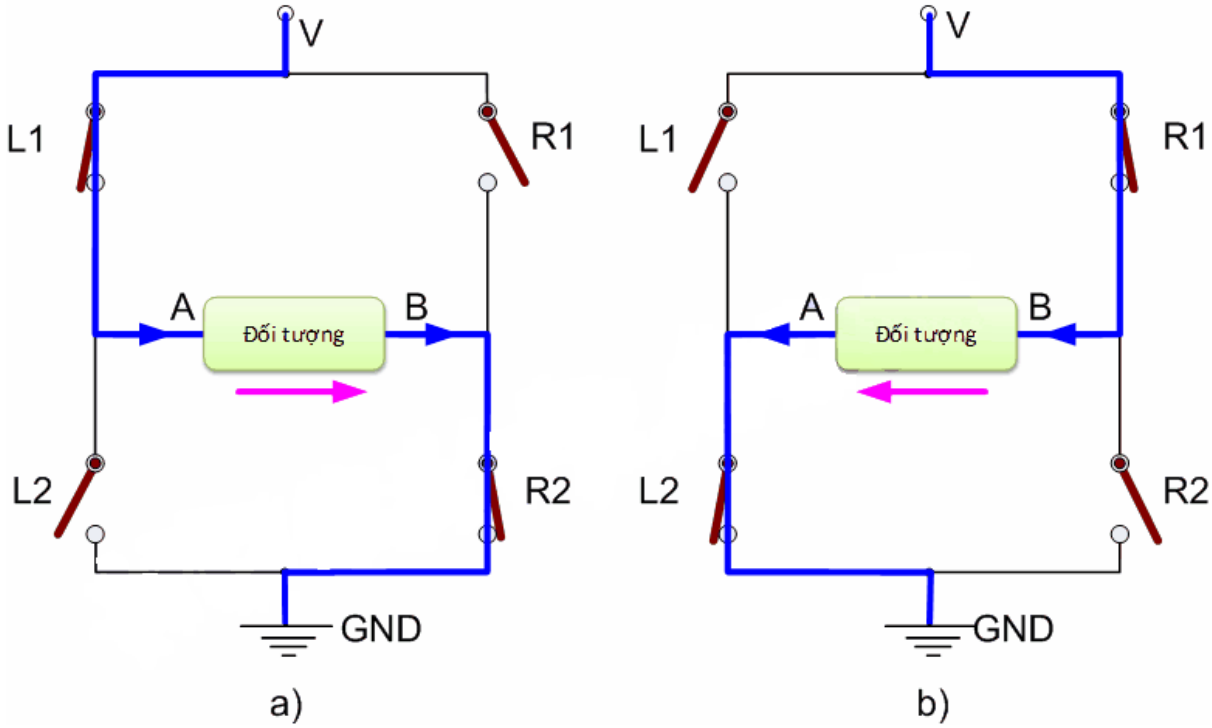
- Trong hình trên chú ý rằng có một lỗ nhỏ bên phía trong của đĩa quay và một cặp phát-thu dành riêng cho lỗ nhỏ này. Đó là kênh I của encoder. Cứ mỗi lần motor quay được một vòng, lỗ nhỏ xuất hiện tại vị trí của cặp phát-thu, hồng ngoại từ nguồn phát sẽ xuyên qua lỗ nhỏ đến cảm biến quang, một tín hiệu xuất hiện trên cảm biến. Như thế kênh I xuất hiện một "xung" mỗi vòng quay của motor.
- Bên ngoài đĩa quay được chia thành các rãnh nhỏ và một cặp thu-phát khác dành cho các rãnh này. Đây là kênh A của encoder, hoạt động của kênh A cũng tương tự kênh I, điểm khác nhau là trong 1 vòng quay của motor, có N "xung" xuất hiện trên kênh A. N là số rãnh trên đĩa và được gọi là độ phân giải (resolution) của encoder. Mỗi loại encoder có độ phân giải khác nhau, có khi trên mỗi đĩa chỉ có vài rãnh nhưng cũng có trường hợp đến hàng nghìn rãnh được chia. Để điều khiển động cơ, ta phải biết độ phân giải của encoder đang dùng. Độ phân giải ảnh hưởng đến độ chính xác điều khiển và cả phương pháp điều khiển.
- Tuy nhiên trên các encoder còn có một cặp thu phát khác được đặt trên cùng đường tròn với kênh A nhưng lệch một chút (lệch M+0,5 rãnh), đây là kênh B của encoder. Tín hiệu xung từ kênh B có cùng tần số với kênh A nhưng lệch pha 90^o. Bằng cách phối hợp kênh A và B người đọc sẽ biết chiều quay của động cơ.



Hình 24 : Hoạt động của một encoder quang.

Hình trên thể hiện sự bộ trí của 2 cảm biến kênh A và B lệch pha nhau. Khi cảm biến A bắt đầu bị che thì cảm biến B hoàn toàn nhận được hồng ngoại xuyên qua, và ngược lại. Hình thấp là dạng xung ngõ ra trên 2 kênh. Xét trường hợp motor quay cùng chiều kim đồng hồ, tín hiệu "đi" từ trái sang phải. Ta hãy quan sát lúc tín hiệu A chuyển từ mức cao xuống thấp (cạnh xuống) thì kênh B đang ở mức thấp. Ngược lại, nếu động cơ quay ngược chiều kim đồng hồ, tín hiệu "đi" từ phải qua trái. Lúc này, tại cạnh xuống của kênh A thì kênh B đang ở mức cao. Như vậy, bằng cách phối hợp 2 kênh A và B chúng ta không những xác định được góc quay (thông qua số xung) mà còn biết được chiều quay của động cơ (thông qua mức của kênh B ở cạnh xuống của kênh A).

Động cơ Dc servo được điều khiển theo nguyên lý điều khiển độ rộng xung (Pulse width modulation – PWM), sử dụng mạch cầu H



Hình 25 Mạch cầu H

Trong hình 25, hãy xem 2 đầu V và GND là 2 đầu (+) và (-) của ắc qui, "đối tượng" là động cơ DC mà chúng ta cần điều khiển, "đối tượng" này có 2 đầu A và B, mục đích điều khiển là cho phép dòng điện qua "đối tượng" theo chiều A đến B hoặc B đến A. Thành phần chính tạo nên mạch cầu H của chúng ta chính là 4 "khóa" L1, L2, R1 và R2 (L: Left, R:Right). Ở điều kiện bình thường 4 khóa này "mở", mạch cầu H không hoạt động.

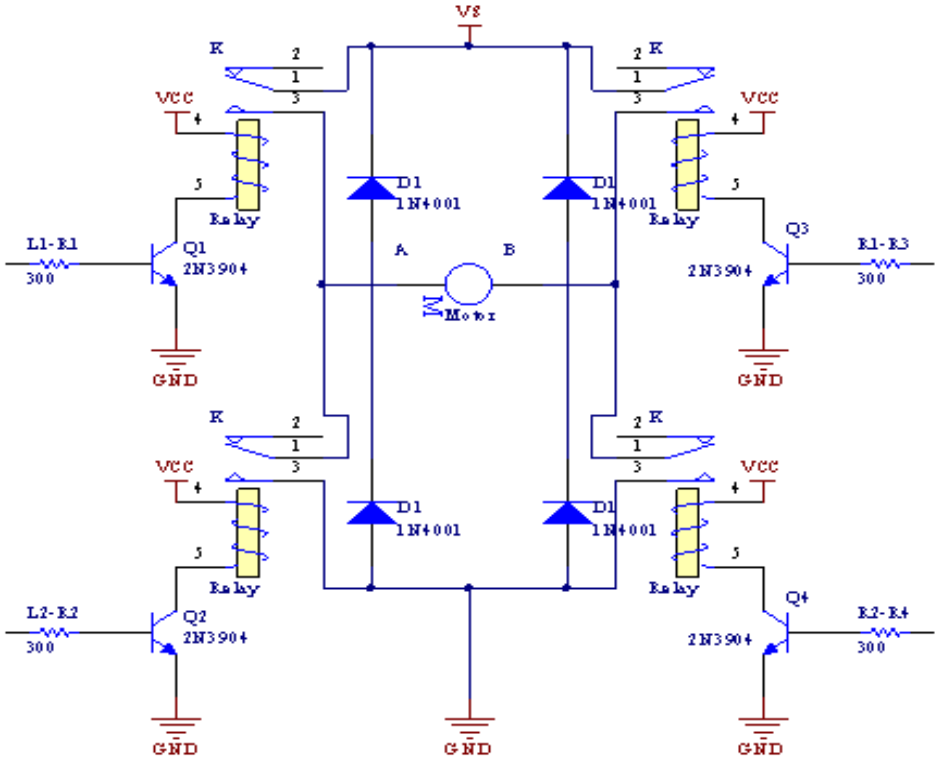
Giả sử bằng cách nào đó mà 2 khóa L1 và R2 được "đóng lại" (L2 và R1 vẫn mở), có một dòng điện chạy từ V qua khóa L1 đến đầu A và xuyên qua đối tượng đến đầu B của nó trước khi qua khóa R2 và về GND (như hình 25 a). Như thế, với giả sử này sẽ có dòng điện chạy qua đối tượng theo chiều từ A đến B.

Bây giờ hãy giả sử khác đi rằng R1 và L2 đóng trong khi L1 và R2 mở, dòng điện lại xuất hiện và lần này nó sẽ chạy qua đối tượng theo chiều từ B đến A như trong hình 25b (V->R1->B->A->L2->GND). Vậy là chúng ta có thể dùng mạch cầu H để đảo chiều dòng điện qua một "đối tượng" (hay cụ thể, đảo chiều quay động cơ)

Nếu đóng đồng thời 2 khóa ở cùng một bên (L1 và L2 hoặc R1 và R2) hoặc thậm chí đóng cả 4 khóa? Hiện tượng "ngắn mạch" (short circuit), V và GND gần như nối trực tiếp với nhau và hiển nhiên ắc qui sẽ bị hỏng hoặc nguy hiểm hơn là cháy nổ mạch xảy ra. Cách đóng các khóa như thế này sẽ làm hỏng mạch cầu H. Để tránh việc này xảy ra, người ta thường dùng thêm các mạch logic để kích cầu H, chúng ta sẽ biết rõ hơn về mạch logic này trong các phần sau.

Giả thiết cuối cùng là 2 trường hợp các khóa ở phần dưới hoặc phần trên cùng đóng (ví dụ L1 và R1 cùng đóng, L2 và R2 cùng mở). Với trường hợp này, cả 2 đầu A, B của "đối tượng" cùng nối với một mức điện áp và sẽ không có dòng điện nào chạy qua, mạch cầu H không hoạt động. Đây có thể coi là một cách "hãm" động cơ (nhưng không phải lúc nào cũng có tác dụng).

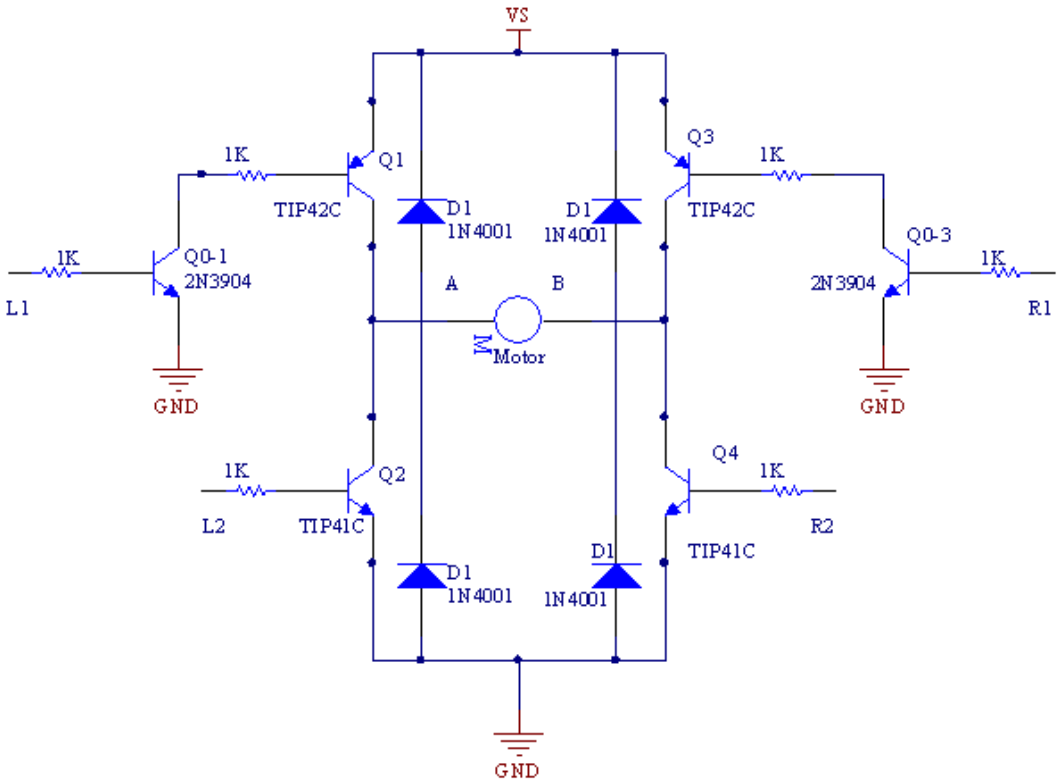
Đó là nguyên lý cơ bản của mạch cầu H. Như vậy thành phần chính của mạch cầu H chính là các "khóa", việc chọn linh kiện để làm các khóa này phụ thuộc vào mục đích sử dụng mạch cầu, loại đối tượng cần điều khiển, công suất tiêu thụ của đối tượng và cả hiểu biết, điều kiện của người thiết kế. Nhìn chung, các khóa của mạch cầu H thường được chế tạo bằng rờ le (relay), BJT (Bipolar Junction Transistor) hay MOSFET (Metal Oxide Semiconductor Field-Effect Transistor).



Hình 26 Mạch cầu H dùng Rơ le.

Trong mạch cầu H dùng rờ le ở hình 26 diode được dùng để chống hiện tượng dòng ngược (nhất là khi điều khiển động cơ). Các đường kích solenoid không được nối trực tiếp với chip điều khiển mà thông qua các transistor, việc kích các transistor lại được thực hiện qua các điện trở.

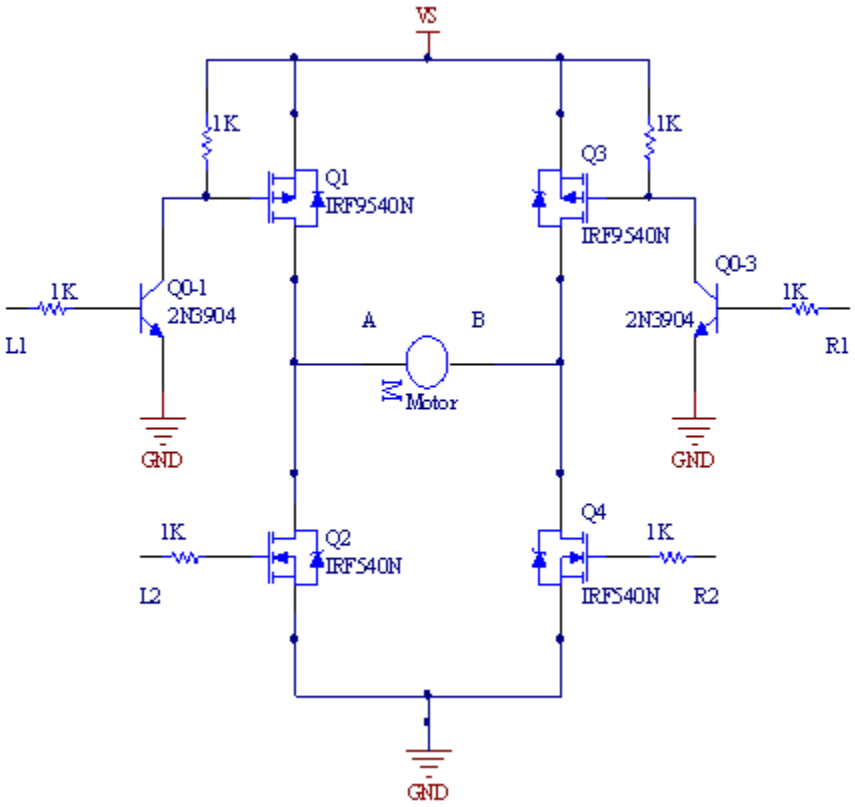
Mạch cầu H dùng rờ le có ưu điểm là dễ chế tạo, chịu dòng cao, đặc biệt nếu thay rờ le bằng các linh kiện tương đương như contactor, dòng điện tải có thể lên đến hàng trăm ampere. Tuy nhiên, do là thiết bị "cơ khí" nên tốc độ đóng/mở của rờ le rất chậm, nếu đóng mở quá nhanh có thể dẫn đến hiện tượng "dính" tiếp điểm và hư hỏng. Vì vậy, mạch cầu H bằng rờ le không được dùng trong phương pháp điều khiển tốc độ động cơ bằng PWM. Người ta thay thế rờ le trong mạch cầu H, bằng các tranzitor gọi là các "khóa điện tử" với khả năng đóng/mở lên đến hàng nghìn hoặc triệu lần trên mỗi giây



Hình 27 Mạch cầu H dùng BJT

Do BJT có thể được kích ở tốc độ rất cao nên ngoài chức năng đảo chiều, mạch cầu H dùng BJT có thể dùng điều khiển tốc độ motor bằng cách áp tín hiệu PWM vào các đường .

Nhược điểm lớn nhất của mạch cầu H dùng BJT là công suất của BJT thường nhỏ, vì vậy với motor công suất lớn thì BJT ít được sử dụng. Mạch điện kích cho BJT cần tính toán rất kỹ để đưa BJT vào trạng thái bão hòa, nếu không sẽ hỏng BJT. Mặt khác, điện trở CE của BJT khi bão hòa cũng tương đối lớn, BJT vì vậy có thể bị nóng…



Hình 28 Mạch cầu H dùng Mosfet

Hình trên minh họa một mạch cầu H dùng MOSFET điển hình với cặp IRF9540 và IRF540

Mạch cầu H dùng MOSFET, hoạt động tương tự như mạch cầu H dùng BJT, tuy nhiên do ưu điểm của các Mosfet là tốc độ đóng mở nhanh, dòng tải lớn do đó được dùng nhiều hơn trong thực tế.

2.3 Nguyên lý điều xung PWM.

PWM (pulse width modulation) là biến điệu độ rộng xung, là phương pháp điều chỉnh điện áp ra tải, hay nói cách khác là phương pháp điều chế dựa trên sự thay đổi độ rộng của chuỗi xung vuông dẫn đến sự thay đổi điện áp ra. Trong các thiết bị cơ điện tử, thường dùng PWM để điều tốc, mô-men của động cơ DC rất có hiệu quả.

PWM (Pulse Width Modulation) đang đóng vai trò gần như tuyệt đối trong các hệ công suất chuyển mạch SMPS (Switch Mode Power Supply) còn

gọi là nguồn xung. Nói một cách khác, nhắc tới nguồn xung thì người ta nghĩ ngay đến PWM.

PWM tạo dựng trên nguyên tắc truyền tải năng lượng từ A đến B dưới dạng các xung vuông toàn áp (biên độ xung gần với điện áp nguồn cung cấp) liên tiếp. Trong đó mức năng lượng tỷ lệ thuận với thời gian mở xung (độ rộng xung tính trên đơn vị thời gian.

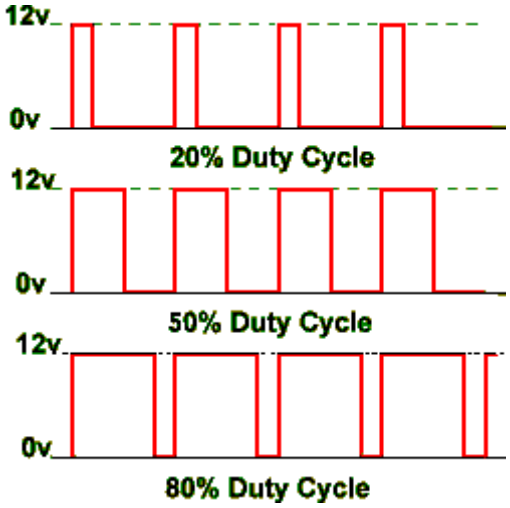
Tần số xung trong PWM có thể cố định hay biến đổi (thường là cố định tần số xung chuyển mạch). Với tần số cố định, chu kỳ t bằng tổng thời gian mở xung t(on) với thời gian tắt xung t(off).

$$t = t(on) + t(off) = t1 + t2$$

Tỷ lệ của thời gian mở trên chu kỳ xung chính là độ sâu điều biến độ rộng xung, đặc trưng thành thuật ngữ "% duty".

Ví dụ, tần số xung 1 KHz --> t = 1 ms.

Với t(on) = 0,5 ms --> ta có độ rộng xung T=(T1/T2).100% = 50%



Hình 29 Tỉ lệ thời gian trên chu kỳ xung

Như vậy

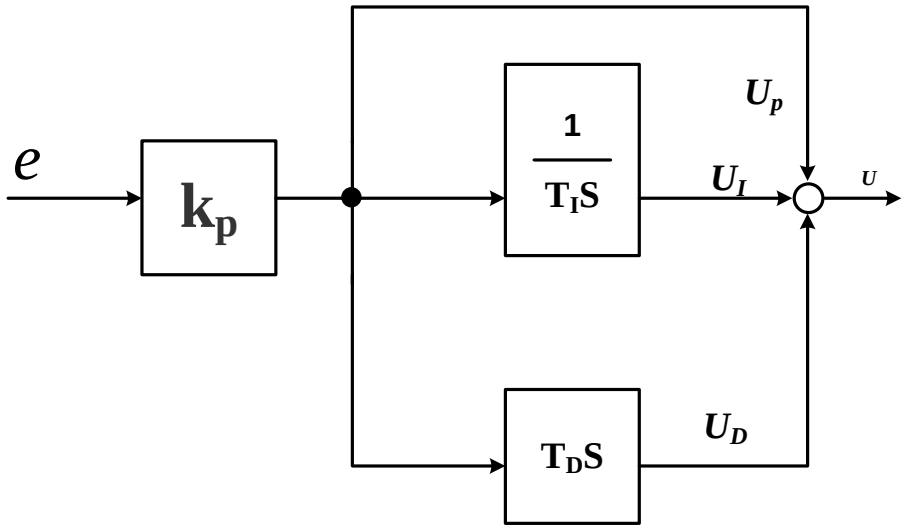
- Đối với PWM = 20% thì Ut = Umax.20%(V)
- Đối với PWM = 50% thì Ut = Umax.50% (V)
- Đối với PWM = 80% thì Ut = Umax.80% (V)

CHƯƠNG 3:

THUẬT TOÁN PID

3.1 Khái quát về bộ điều khiển PID

Cấu trúc của bộ điều khiển PID (hình30) gồm 3 thành phần là khâu khuếch đại (P), khâu tích phân (I) và khâu vi phân (D). Khi sử dụng thuật toán PID nhất thiết phải lựa chọn chế độ làm việc P, I hay D và sau đó đặt tham số cho các chế độ đã chọn. Một cách tổng quát, có ba thuật toán cơ bản được sử dụng là P, PI và PID.



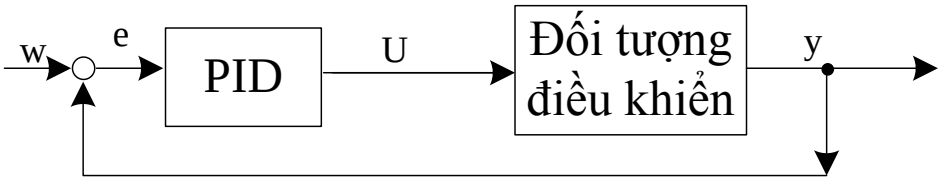
Hình 2930 Cấu trúc bộ điều khiển PID

Bộ điều khiển PID có cấu trúc đơn giản, dễ sử dụng nên được sử dụng rộng rãi trong điều khiển các đối tượng SISO theo nguyên lý hồi tiếp, bộ PID có nhiệm vụ đưa sai lệch e(t) của hệ thống về 0 sao cho quá trình quá độ thỏa mãn các yêu cầu cơ bản về chất lượng:

-Nếu sai lệch tĩnh e(t) càng lớn thì thông qua thành phần $u_P(t)$, tín hiệu điều chỉnh u(t) càng lớn.

-Nếu sai lệch e(t) chưa bằng 0 thì thông qua thành phần $u_I(t)$, PID vẫn còn tạo tín hiệu điều chỉnh.

-Nếu sự thay đổi của sai lệch e(t) càng lớn thì thông qua thành phần $u_D(t)$, phản ứng thích hợp của u(t) sẽ càng nhanh.



Hình 3031 Điều khiển hồi tiếp với bộ điều khiển PID

Bộ điều khiển PID được mô tả bằng mô hình vào-ra:

$$u(t) = k_p\left[e(t) + \frac{1}{T}\int_0^t e(\tau)d\tau + \mathrm{T_D}\frac{de(t)}{dt}\right]$$

Trong đó :

e(t) – tín hiệu đầu vào.

u(t) – tín hiệu đầu ra.

k_p – tín hiệu khuếc đại

T_I – hằng số tích phân.

T_D – hằng số vi phân.

Từ mô hình vào – ra trên, ta có được hàm truyền đạt của bộ điều khiển PID

$$R(s) = k_p\left(1 + \frac{1}{T_I s} + T_D s\right)$$

3.2 Phương pháp xác định tham số PID

Có nhiều phương pháp xác định tham số của bộ điều khiển PID:

- **Phương pháp Ziegler-Nichols.**
- Phương pháp Chien-Hrones-Reswick.
- Phương pháp tổng T của kuhn.
- Phương pháp tối ưu modul và phương pháp tối ưu đối xứng.
- Phương pháp tối ưu theo sai lệch bám

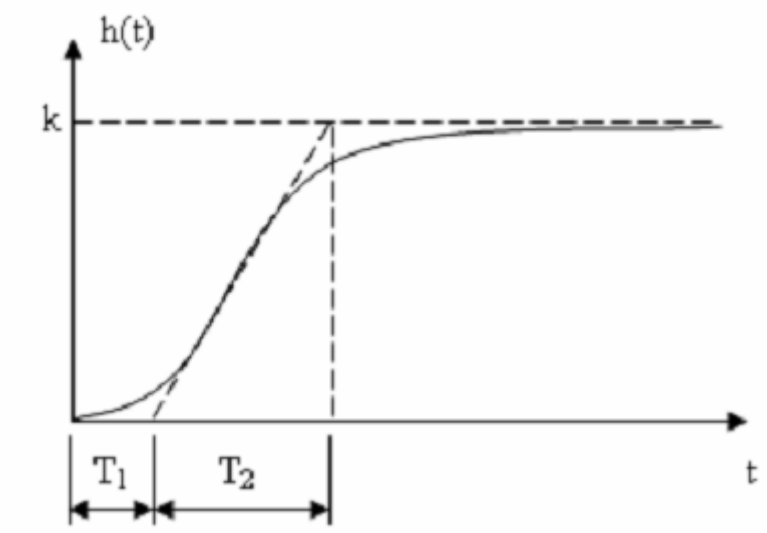
Đề tài sử dụng phương pháp Ziegler-Nichols nên ta chỉ đi sâu vào phương pháp này.

a. Phương pháp Ziegler-Nichols.

Phương pháp Ziegler-Nichols là phương pháp thực nghiệm để xác định tham số bộ điều khiển P, PI hoặc PID bằng cách dựa vào đáp ứng quá độ của đối tượng điều khiển. Tùy theo đặc điểm của từng đối tượng, Ziegler - Nichols đưa ra hai phương pháp lựa chọn tham số của bộ điều khiển

➢ **Phương pháp Ziegler-Nichols thứ nhất:**

Phương pháp này áp dụng cho các đối tượng có đáp ứng đối với tín hiệu vào là hàm nấc có dạng chữ S như nhiệt độ lò nhiệt, tốc độ động cơ…



Hình 3132 Đáp ứng nấc cho hệ hở hình chữ S

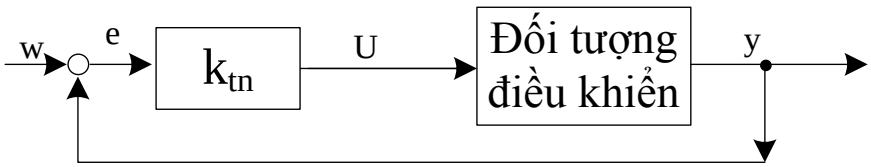
Thông số của bộ điều khiển được chọn theo bảng sau:

Thông số / BĐK	k_p	T_I	T_D
P	$T_2/(k.T_1)$	-	-
I	$0,9T_2/(k.T_1)$	$T_1/0,3$	-
D	1,2T2/(k.T1)	2T1	0,5T1

➢ **Phương pháp Ziegler-Nichols thứ hai:**

Phương pháp này áp dụng cho đối tượng có khâu tích phân lý tưởng như mực chất lỏng trong bồn chứa, hệ truyền động dùng động cơ… đáp ứng quá

độ của hệ hở của đối tượng tăng đến vô cùng. Phương pháp này được thực hiện như sau:

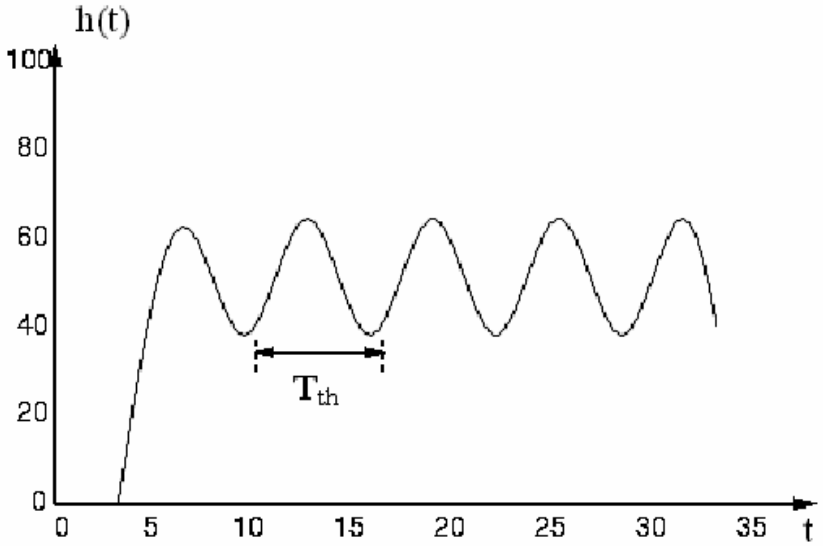


Hình 3233 Xác định hằng số khuếch đại lớn

-Thay bộ điều khiển PID trong hệ kín bằng bộ khuếc đại (hình 33)

-Tăng hệ số khuếc đại tới giá trị tới hạn k_{tn} để hệ kín ở chế độ biên giới ổnđịnh, tức là h(t) có dạng dao động điều hòa.

-Xác định chu kỳ T_{th} của dao động.



Hình 3334 Đáp ứng của hệ kín khi K= K_{tn}

Thông số / BĐK	k_p	T_I	T_D
P	$0,5k_{tn}$	-	-
I	$0,45\ k_{tn}$	$0,85\ T_{th}$	-
D	$0,6\ k_{tn}$	$0,5\ T_{th}$	$0,125\ T_{th}$

a. Tính toán thông số cho bộ điều khiển PID.

Hệ thống chỉ thực sự đạt hiệu quả khi ta thiết kế cho nó 1 bộ điều khiển phù hợp.Với kết cấu phần cứng và đặc tính điều khiển là điều khiển tốc độ thì bộ

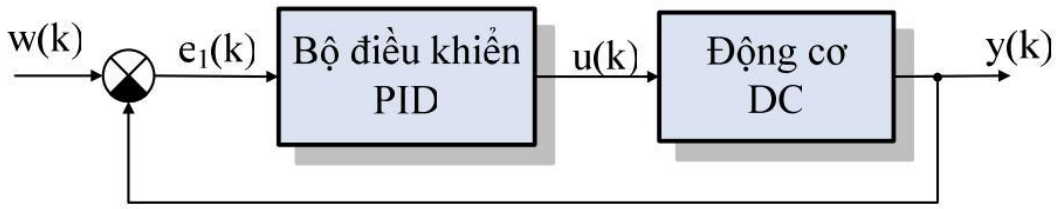
điều khiển PI số là thích hợp hơn cả, ta chỉ dùng thêm khâu vi phân nếu liên quan tới điều khiển vị trí. Do ứng dụng vi điều khiển ngày càng rộng rãi, chính vì vậy việc số hóa bộ điều khiển PI là rất quan trọng.Đi cùng với nó là các phương pháp cho ta số hóa một cách tương đối chính xác như Ziegler-Nichols…….v v.

3.3 Thuật toán điều khiển:

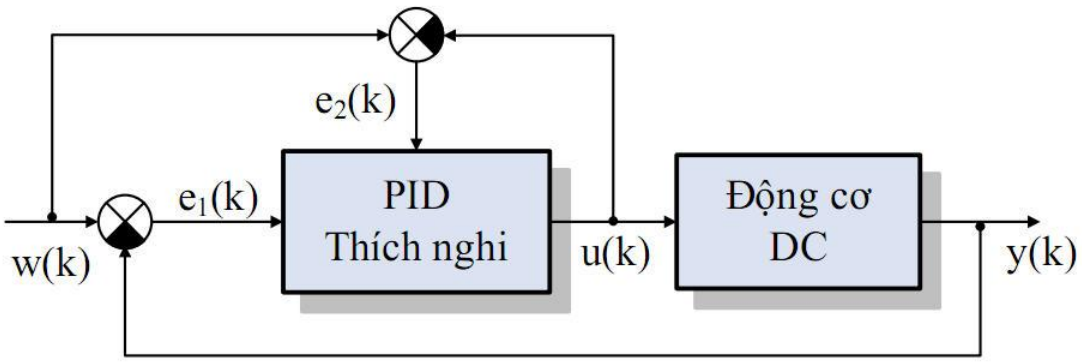
Về nguyên tắc khi xây dựng bộ điều khiển cho một đối tượng nào đó thì ta cần biết được cấu trúc, đặc tính, hàm truyền…của đối tượng.

Với động cơ 1 chiều bất kì thì ta không phải lúc nào cũng mổ xẻ động cơ ra rồi đo đạc các thông số của nó, chính vì vậy bộ điều khiển phải có nhiệm vụ tạo ra tín hiệu điều khiển phù hợp nhất, có khả năng điều khiển linh hoạt với nhiều động cơ khác nhau.

Với điều khiển PID thì có rất nhiều cách thức thực hiện, riêng bộ PID thì có PID thường và PID thích nghi



Hình 3435 PID thường

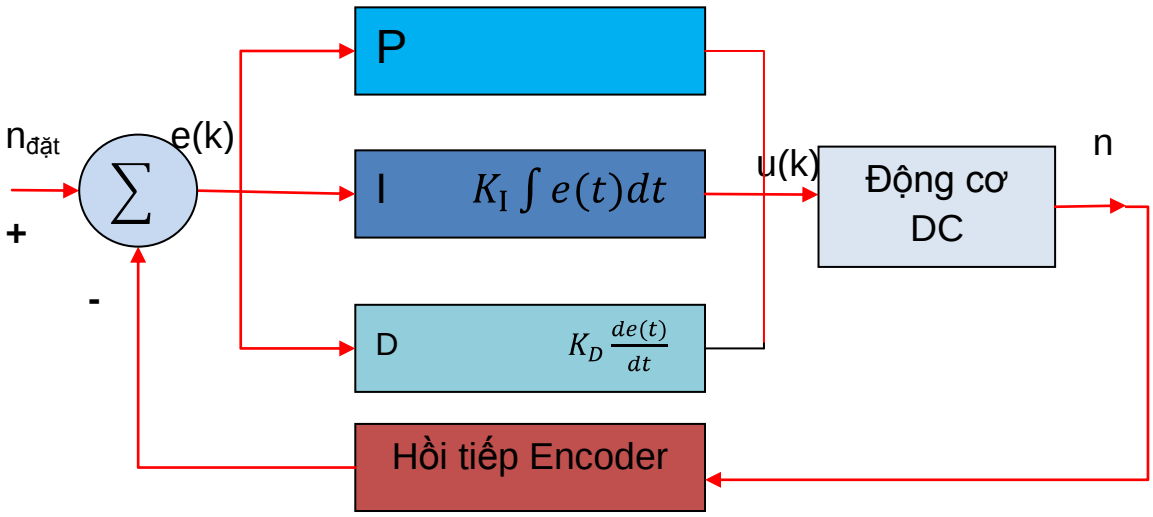


Hình 3536 PID thích nghi

Với PID thường thì đầu ra chỉ phụ thuộc vào tín hiệu $e_1(k)$, còn với PID thích nghi thì còn phụ thuộc thêm vào $e_2(k)$.

Thực tế cho thấy PID thích nghi hoạt động tốt và ổn định hơn PID thường.

- Phương trình toán học bộ PID:



Hình 37 Sơ đồ khối bộ PID

$$U(t) = K_P * e(t) + K_I \int e(t) * dt + K_D \frac{de(t)}{dt}$$

CHƯƠNG 4: CARD GIAO TIẾP MÁY TÍNH VÀ THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN ĐỘNG CƠ DC SERVO TRÊN LABVIEW

4.1 CARD GIAO TIẾP MÁY TÍNH

a) Giới thiệu CARD USB – 9001 :

Thông số chung	
Cổng kết nối	USB (chuẩn giao tiếp RS232)
Hỗ trợ hệ điều hành	Windows
Kiểu đo	6 kênh đo điện áp (ADC) 1 bộ đếm xung từ các loại encoder (đếm lên hoặc xuống tùy theo chiều quay encoder)
Điều khiển	· 4 kênh xuất tín hiệu số · 2 kênh xuất tín hiệu điều chế xung (PWM)
Họ DAQ	
Đọc tín hiệu Analog	
Số kênh	6 SE
Tốc độ lấy mẫu	142S/s
Độ phân giải	8 bits

Trích mẫu đồng thời	Không
Ngưỡng điện áp giới hạn lớn nhất	0 tới 5 V
Độ chính xác	10 mV (V_{ref}=2.56V)
Tín hiệu analog từ các loại cảm biến	Nhiệt độ, áp xuất, lưu lượng vv.
Lĩnh vực ứng dụng đo điện áp	Điều khiển tự động, ô tô, công nghiệp
Xuất tín hiệu PWM	
Số kênh	2
Tốc độ cập nhật	100 S/s
Độ phân giải	8 bits
Ngưỡng điện áp	0..5 V
Tín hiệu điều khiển dòng điện	10 mA (dòng ngắn mạch)
Các chân xuất tín hiệu số	

Số kênh	4
Timing	Software
Logic Levels	TTL
Ngưỡng điện áp ra	0..5 V
Bộ lọc vào lập trình được	No
Output Current Flow	Sinking, Sourcing
Dòng điện (Kênh/Tổng)	10 mA/100 mA
Bộ đếm xung	
Số bộ đếm	1 (đếm lên hoặc đếm xuống)
Độ phân giải	16 bits
Tần số nguồn xung lớn nhất	250 KHz
Độ rộng xung vào nhỏ nhất	2 us
Mức logic	TTL

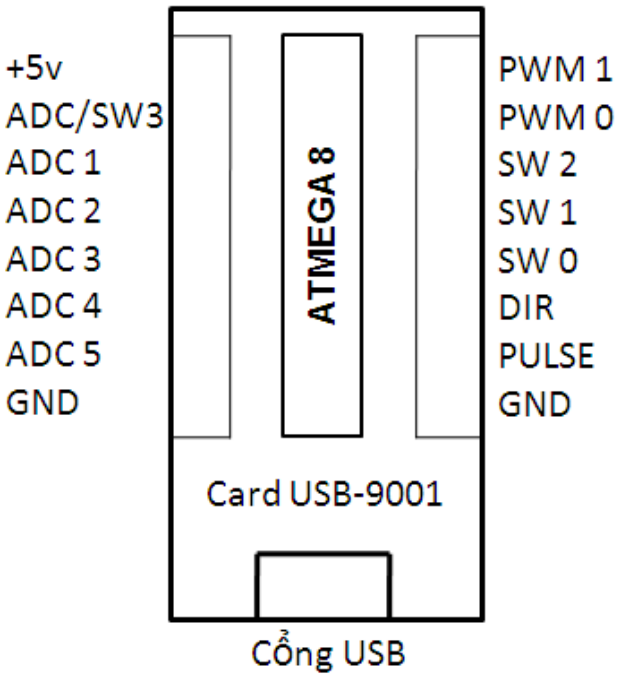
Ngưỡng cực đại	0..5 V
Ứng dụng	Đo tốc độ động cơ từ Encoder, đo xung, vv.
Cho phép thực hiện nhớ tạm	Yes
Tác động (Triggering)	Digital
Kích thước card Hocdelam USB - 9001	
Dài	10 cm
Rộng	6 cm
Cao	2.5 cm
Đầu nối vào ra	Dễ dàng mở bằng tua vít
Chất liệu vỏ hộp	Nhựa
Nhiệt độ bảo quản	Dưới 65 độ C
Chiều dài cable nối	Lớn hơn 500mm và dài hơn 1000mm

b) Cách sử dụng:

Chân	Kí Hiệu	Giá trị	Mô tả	Giá trị Reset
Input	ADC0 -ADC5		Nhận tín hiệu dạng tương tự(analog). Vref sẽ là 5v trên USB hoặc 2.55v do set trên máy tính	NA
Input	PULSE		Đếm xung cạnh lên (0-5v)	0
Input	DIR	0 5V	Set bộ đếm xung PULSE đếm xuống Set bộ đếm xung PULSE đếm lên	5v
Output	PWM0 PWM1		Tạo xung với tần số cố định và hệ số xung thay đổi từ 0-255 tùy số đặt trên máy tính (xung 0-5v và 2 tổng trở 470 Ohm)	0
Output	SW0-SW2 SW3		Tính hiệu ra dạng số(0 hoặc 5v. Tổng trở 470 Ohm) tùy set trên máy tính Tính hiệu ra dạng số (0 hoặc 5v), sẽ không sử dụng ADC0	0
Nguồn	GND		Mass	0
Nguồn	+5v		Lấy từ USB	5v

Sơ đồ chân:

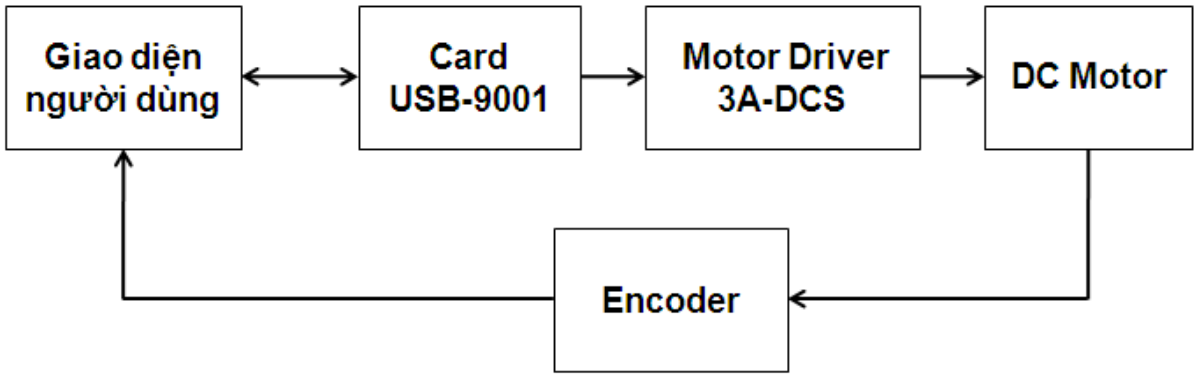
- ADC0-ADC5: trả về giá trị chuyển đổi các chân ADC tương ứng (0-255).
- DAC0-DAC1: đặt giá trị ngõ ra chân PWM cho chân DAC tương ứng (0-255).
- SW0-SW3: đặt giá trị cho 3 ngõ ra số (TRUE-FALSE).
- PULSE: trả về giá trị số xung đã đếm từ chân PULSE (giá trị từ 0-65635).



Hình 3638 Các chức năng của Card USB - 9001

4.2 Thiết kế hệ thống điều khiển trên LABVIEW thông qua Card USB – 9001:

Mô hình điều khiển sử dụng card USB-9001:



Hình 3739 Mô hình điều khiển

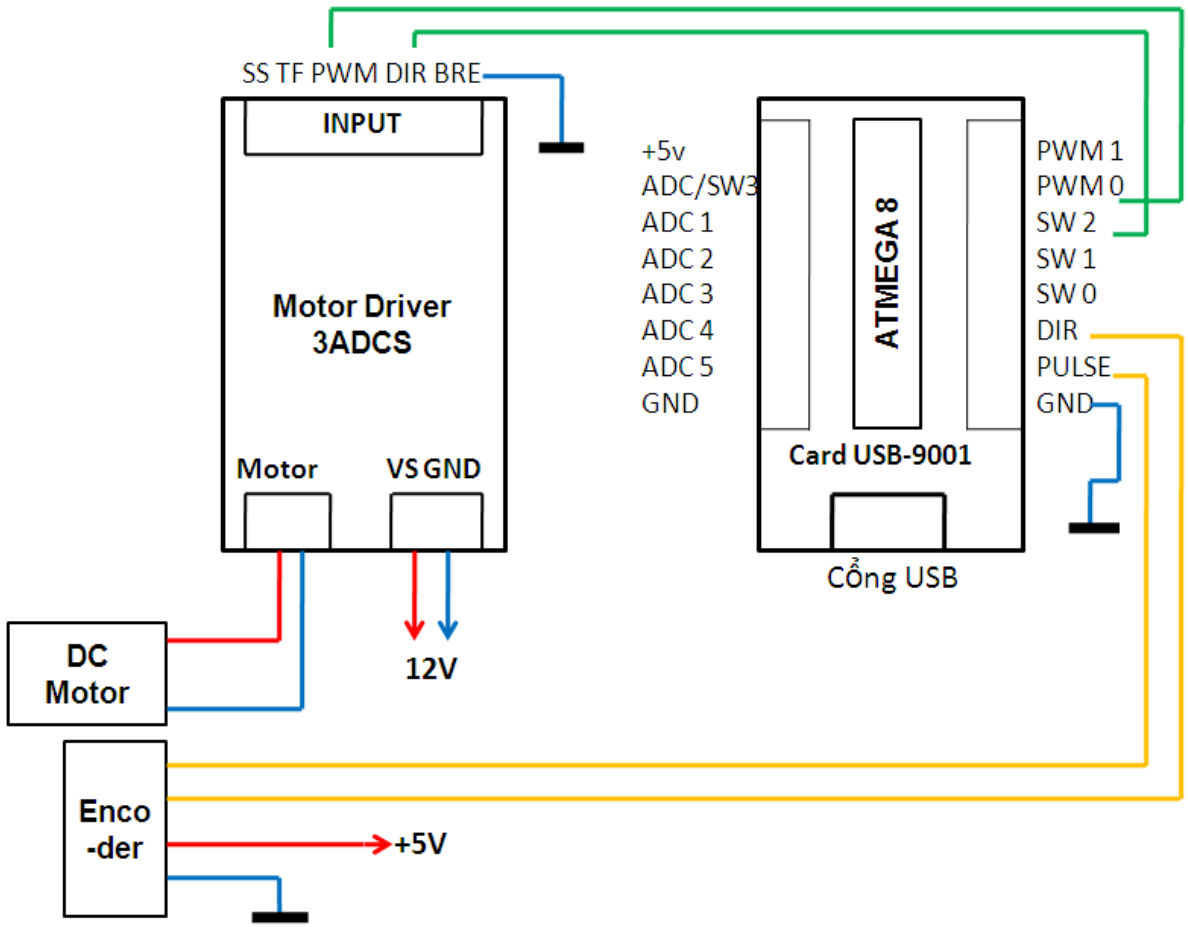
Trong đó:

+ Phần giao diện người dùng: được tạo ra trên máy tính bằng ngôn ngữ LabVIEW

+ Card USB-9001: đóng vai trò giao tiếp với máy tính qua cổng USB. Nó nhận lệnh điều khiển từ máy tính, xử lý - tạo xung PWM để ổn định tốc độ và xác định chiều quay cho Motor DC.

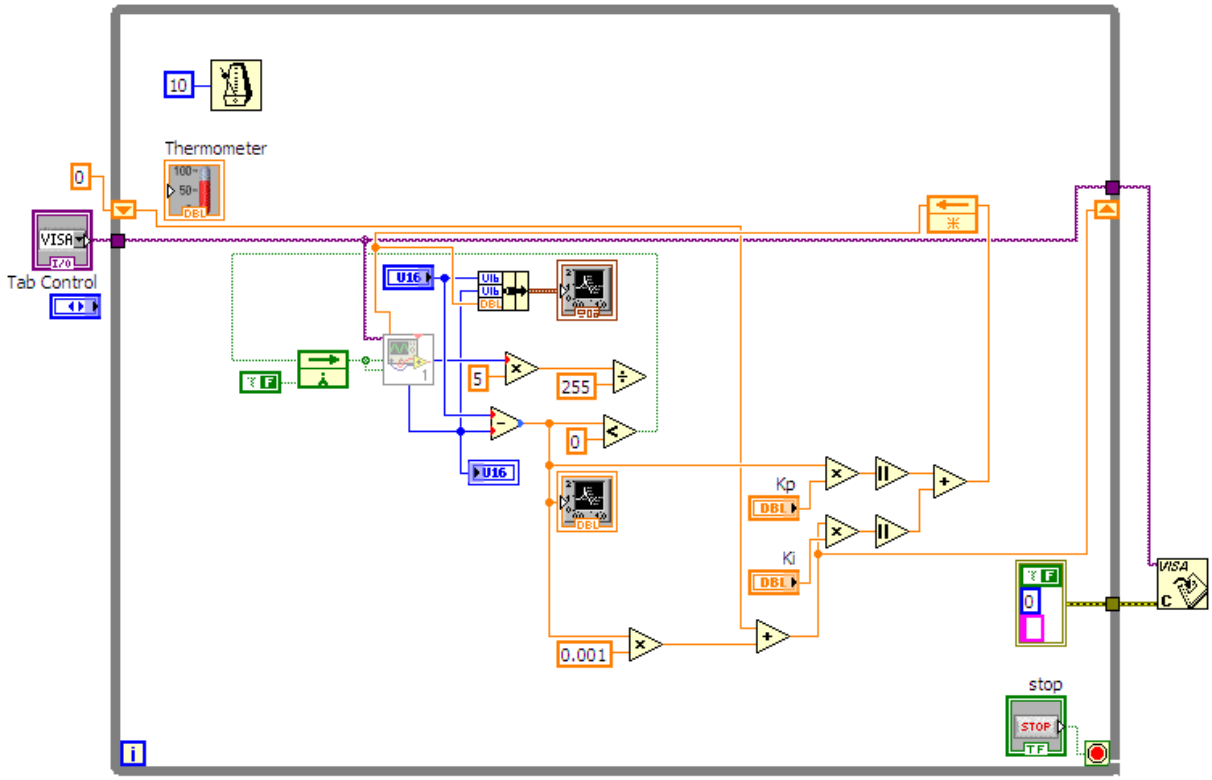
+ Motor Driver 3A-DCS: là modul công suất mắc theo kiểu mạch cầu chữ H, để cấp dòng và đảo chiều quay cho motor DC.

+ Encoder: đưa xung hồi tiếp về Card USB-9001, để ổn định tốc độ theo luật PID.
Hình 40 là sơ đồ kết nối phần cứng cụ thể giữa Card USB-9001, Motor Driver 3A-DCS với Motor DC và Encoder



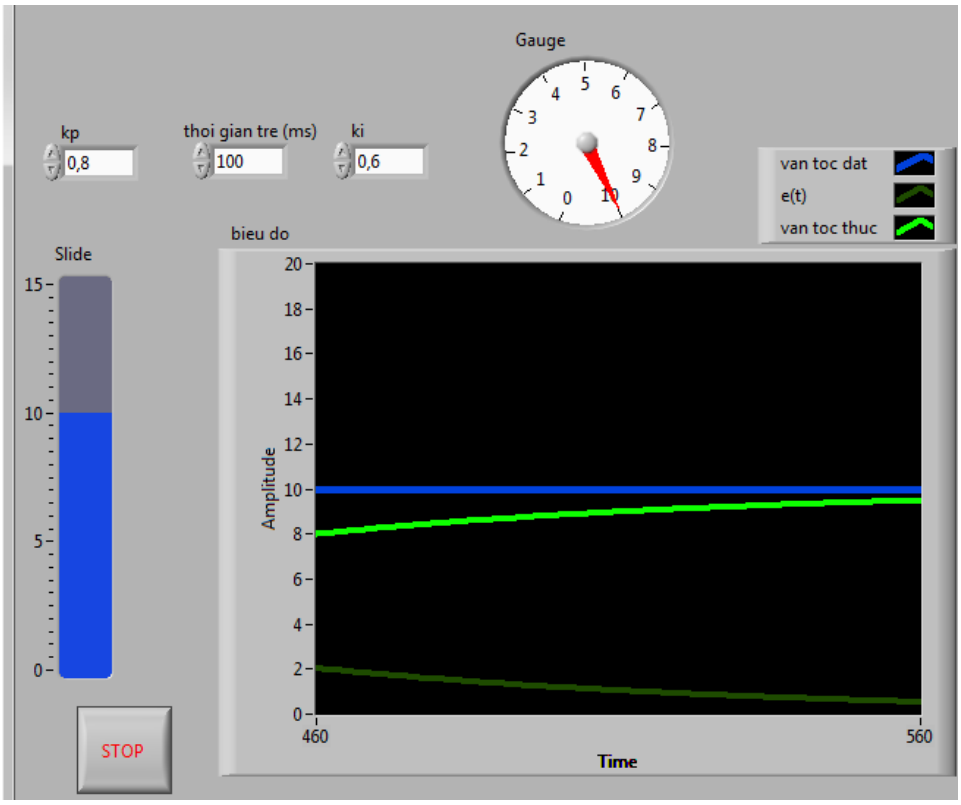
Hình 3840 Sơ đồ nối phần cứng giữa Card và Motor Driver

4.3 Sơ đồ nguyên lý của hệ thống:



Hình 41 Sơ đồ nguyên lý hệ thống điều khiển

4.4 Kết quả đạt được



Hình 42 Kết quả đạt được

Đường màu xanh dương là vận tốc đặt.

Đường màu vàng là vận tốc thực của động cơ.

Đường màu xanh lục là sai số giữa vận tốc đặt và vận tốc thực e(t).

Theo kếết quả ta thấy:

-Đường màu vàng đang tiến tới đường màu xanh dương (vận tốc động cơ đang tiến tới giá trị đặt).

- Đường màu xanh lục đang tiến về không (sai số giữa điểm đặt và vận tốc thực đang giảm về không để tiến tới ổn định).

Kết Luận:

- Bằng phương pháp điều khiển này ta không cần thi công nhiều mạch phức tạp, dễ dàng kiểm tra,sữa chữa và thay đổi hệ thống điều khiển nếu hệ thống cũ có sai sót hoặc chưa như mong muốn của người điều khiển.
- Chi phí ban đầu lớn (so với hệ thống điều khiển Motor DC khác) cho mua bản quyền phần mềm phần cứng của NI.
- Tính cơ động kém, không thể làm việc trong những môi trường khắc nhiệt.
- Người dùng phải có kiến thức về máy tính , phần mềm phần cứng do đó tính thông dụng kém.

Tài liệu tham khảo:

1) Phạm Minh Hà, Kỹ thuật mạch điện tử
2) Lương Ngọc Hải, Giáo trình kỹ thuật xung số
3) Getting Stared With LabVIEW – Author (Apache Software Foundation).
4) LabVIEW Fundamentals - Author (Apache Software Foundation).
5) LabVIEW Quick Reference Card – National Instruments Corporation.
6) LabVIEW Advanced I, của hãng National Instrument.
7) LabVIEW Basics I + Basics II Course Manual của National Instrument.

8) PCI – 1710/1710 HG Multifunction DAS card for PCI bus user's Manual.
9) LabVIEW Tutorial Manual - National Instruments Corporation.
10) http://vietnam.ni.com/
11) http://hoiquandientu.com/
12) http://hocdelam.org/site1/

MỤC LỤC