## NI MyDaq:

Einfach über USB ansprechbar, mehrere digitale Ein und Ausgänge, Analoge Eingänge, hat einen AD Wandler eingebaut. Ausgezeichnet in LAbView integriert, kann fast alles messen und steuern.

## Labview allgemein:

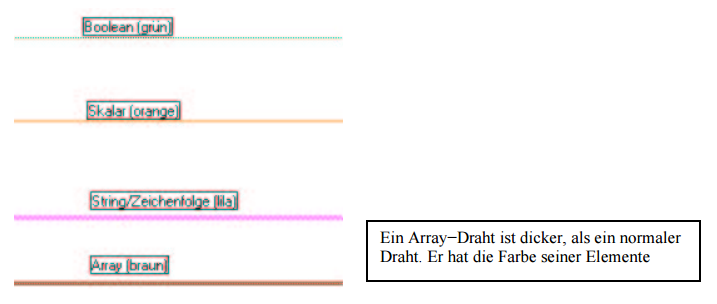
Labview ist eine grafische Programmierumgebung von National Instruments (NI). Es dient zur Erstellung von Prüf-, Mess-, Steuer- und Regelanwendungen. Es lässt sich aufgrund der intuitiven Handhabung einfach in Projekte und Anwendungen integrieren. Zudem verbindet Labview grafische Programmierung mit Hardware, um die Entwicklung von Anwendungen zu vereinfachen und zu beschleunigen. Es unsterstützt eine Vielzahl von externen (Mess)Geräten. Deshalb eignet es sich gut für den Einsatz mit einem NI MyDaq (LabVIEW beinhaltet auch speziell für dieses Gerät bereits einige vorgefertigte Funktionen).

Ein Labview Programm bezeichnet man als VI (Virtual Instrument). Ein solches besteht grundsätzlich aus einem Front Panel (dient während dem Programmablauf zur Eingabe und Ausgabe von Daten) und einem Block Diagram (dient während der Entwicklung zum Platzieren von Programmelementen, um den Datenfluss sowie den Programmablauf festzulegen).

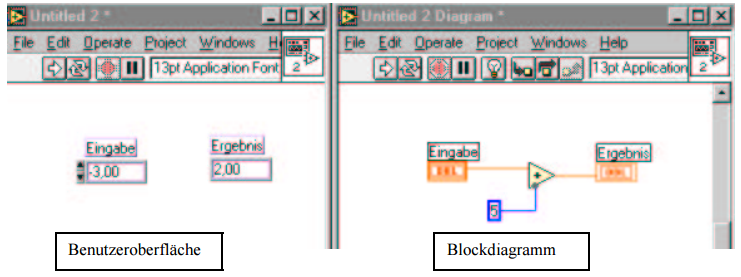
**Vorteile von LabVIEW:**1.) Einfachheit: Wegen grafischer Programmierung.  
2.) Erstellen der Bedieneroberfläche über das Front Panel ist sehr bequem, da die Elemente mit denen im Block Diagramm automatisch verknüpft werden.  
3.) Grafische Darstellung des Programms erhöht die Lesbarkeit (=> auch für Naturwissenschaftler und Techniker leichter verständlich).  
4.) Viele bereits inkludierte Funktionen, vor allem im Bereich der Datenanalyse und Mathematik. Es gibt auch bereits viele Funktionen zum Ansteuern von externen Geräten.  
5.) Zum Erreichen von weit entfernten Geräten werden Kommunikationsprotokolle und Verbindungstechniken unterstützt (z.B.: TCP; siehe Sockets).

**Front Panel – Komponente:**Ist also die Benutzeroberfläche der Anwendung/VI. Dort gibt es einerseits Bedienelemente, welche es dem Benutzer erlauben Daten einzugeben. Diese beinhalten zum Beispiel Elemente zum Eingeben von Texten oder Zahlen, aber auch Schalter, Knöpfe, etc. (um einen Zustand zu wechseln) und weitere. Andererseits gibt es die Anzeigeelemente, welche zum Anzeigen von Daten. Diese Daten sind meist Zahlen, Texte oder Zustände (dargestellt durch LEDs etc.).

**Block Diagram – Komponente:**In diesem wird der Programmablauf grafisch entwickelt. Zudem kann hier auf die Dateneingaben am Front Panel reagiert werden und es können natürlich auch Messergebnisse und ähnliches am Front Panel angezeigt werden. Um den Programmablauf zu definieren, verbindet man mehrere Elemente (diese können vom Front Panel stammen, oder Funktionen wie +/- oder Messfunktionen etc. sein) mit Drähten an en Anschlüssen eines Elementes. Jedes Element im Block Diagramm enthält also Eingänge sowie Ausgänge für diese Anschlüsse. Über diese Anschlüsse werden Daten/Informationen ausgetauscht. Eine Funktion wird nur ausgeführt, wenn an allen angeschlossenen Eingängen ein Wert angekommen ist. Für das Block Diagram gibt es zahlreiche vorgefertigte Funktionen für Messungen, mathematische Berechnungen, das Ansteuern von Geräten und vieles weiteres.

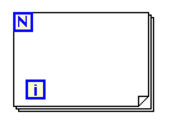
Drähte haben eine Farbe und Dicke, welche den übertragenen Datentyp angeben:  


**Beispiel: Front Panel und Block Diagram:**



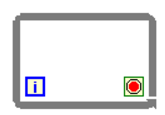
## Block Diagramm:

**Kontrollstrukturen:**For-Schleife:



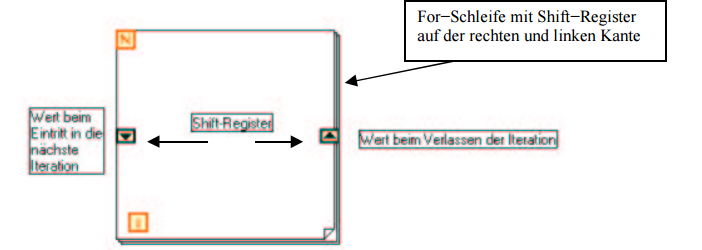
Die for−Schleife führt die Programmteile, die sie enthält, genau N−mal aus. Die Variable N muß gesetzt werden, indem eine numerische Konstante mit dem Anschluß "N" verdrahtet wird. Die Laufvariable "i" enthält die Nummer der gerade anstehenden Iteration. Die Zählung von "i" beginnt bei 0! Bei jedem Schritt wird "i" um den Wert 1 erhöht. Die LabVIEW−for−Schleife ist also äquivalent zu folgendem C−Code: *for(i=0; i<N;i++) {statement1;statement2; ....}*

While-Schleife:

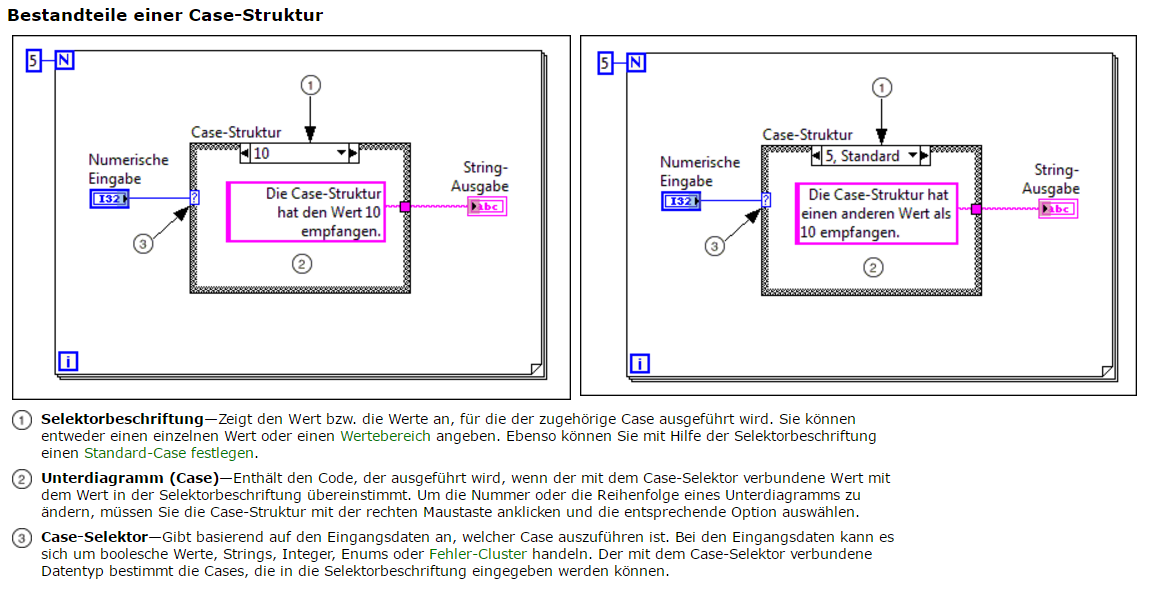


Die while−Schleife führt die in ihr enthaltenen Programmteile so lange durch, wie die Abbruchbedingung wahr ist. An die Abbruchbedingung muss also ein logischer Ausdruck angeschlossen werden. Wenn nichts angeschlossen wird, wird der logische Wert "false" angenommen, was zur Folge hat, dass die Schleife dann genau einmal ausgeführt wird. Die Laufvariable "i" enthält die Nummer der gerade anstehenden Iteration. Die while−Schleife ist äquivalent zu folgendem C−Code: *do {statement1;statement2; ....} while (condition == TRUE )*Es gibt in Labview 2 Einstellungen für die Abbruchbedingung: Bei TRUE stoppen oder Bei TRUE fortfahren.

Shift-Register:

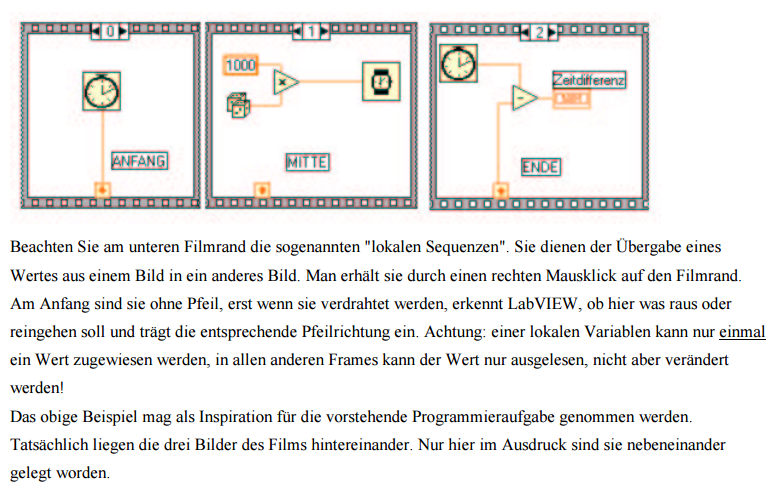


Shift−Register − verfügbar für while−Schleifen und for−Schleifen − sind lokale Variablen, die Werte von einer Iteration der Schleife in die nächste Iteration transferieren.

Case-Struktur:

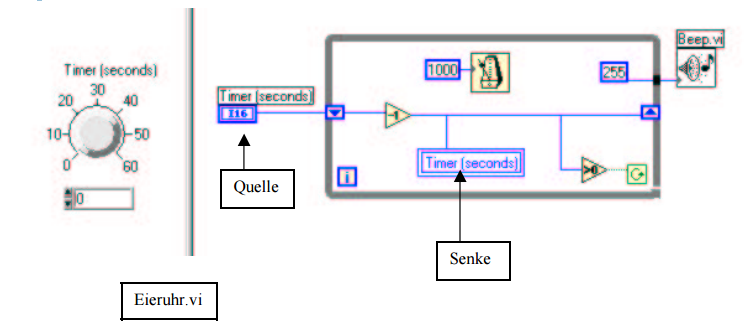
In LabVIEW können Sie Alternativen mit einer einzigen Form realisieren: die "case"−Struktur, die entweder eine boolesche "if ... then ... else ... " − Struktur darstellt, oder die "switch−case"−Struktur: Welcher der beiden Fälle eintritt, hängt vom Typ der Eingabevariablen ab, die mit der Abfrage verdrahtet wird (also Boolean oder String/int/etc.).

Sequenzstruktur:

Alle klassischen Programmiersprachen enthalten einen inhärent durch die zeilenweise Abfolge der Programm− anweisungen festgelegten Ablauf. Bei LabVIEW ist das anders, denn LabVIEW selbst legt diese Abfolge fest. Wenn Sie dies verhindern müssen, weil Sie auf einen fest (von Ihnen, vom Problem, vom Lauf der Dinge) bestimmten Ablauf angewiesen sind, müssen Sie die Sequenzstruktur einsetzen.

Lokale Variable:

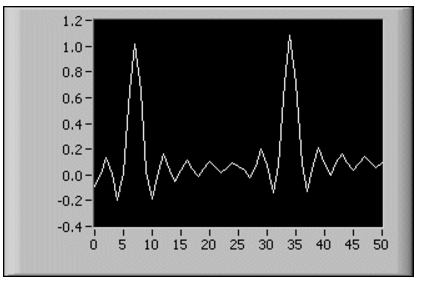
Stellen Sie sich vor, daß Sie in einem VI den Wert einer Variablen an vielen verschiedenen Stellen benötigen. Mit vielen Drahtmetern könnten Sie dieses Problem zwar lösen. Aber Sie erhalten bei dieser Vorgehensweise ein verwirrendes Programm! Außerdem kann man sich Fälle vorstellen, wo man selbst mit viel Draht nicht weiterkommt. Stellen Sie sich vor, Ihr Programm enthält zwei while−Schleifen, die beide mit der gleichen Bedingung abgebrochen werden sollen, dann bekommen Sie Schwierigkeiten. Das Problem können Sie lösen, wenn Sie in der rechten Schleife eine lokale Variable, die eine Kopie der Programmabbruchvariablen ist, erzeugen. Lokale Variable haben noch eine wichtige Eigenschaft: Sie können als Datenquelle oder als Datensenke definiert werden − beachten Sie diesen Punkt, wenn Sie Verdrahtungsprobleme bekommen. Darüber hinaus kann eine lokale Variable an der einen Stelle im Lesemodus (Quelle) sein und an einer anderen Stelle im Schreibmodus, also Senke! Damit können Sie z. Bsp. eine virtuelle Eieruhr programmieren: eine Uhr also, die auf einen gewissen Wert gestellt (control−Modus) wird und dann bis auf Null zurückläuft (indicator− Modus):



Globale Variable:

Globale Variable sollten aus den oben genannten. Gründen nur sehr sorgfältig und überlegt eingesetzt werden. Der Geltungsbereich von lokalen Variablen überspannt nur das VI, in dem sie definiert wurde. Angenommen, Sie müssen mehrere VI’s betreiben, die "quasi−parallel" arbeiten sollen. Wenn dann gewisse Informationen in den verschiedenen VI’s gemeinsam benötigt werden, dann müssen Sie globale Variable einsetzen.

**Charts:**Mit LabVIEW können Sie die bei einer Messung erfaßten oder die bei einer Simulation berechneten Werte in sehr bequemer Weise in graphischer Form in Form von Diagrammen oder Graphen darstellen. Der Unterschied dabei ist, daß Diagramme (Charts) jeden neu erfaßten oder berechneten Meßwert sofort darstellen, während bei Graphen zunächst die Gesamtheit aller Werte erfaßt oder berechnet werden und dann erst dargestellt werden.



**Sockets:**Mit Sockets kann man zum Beispiel Messgeräte oder andere Anwendungen über beispielsweise TCP/IP odr UPD ansteuern. Somit ist es also auch möglich Daten von weiter entfernten Messgeräten zu erhalten und diese Geräte zu steuern.  
Für diese Sockets gibt es bereits vordefinierte Funktionen.

