# LabVIEW

aus Wikipedia, der freien Enzyklopädie

**LabVIEW** ist ein grafisches Programmiersystem von National Instruments. Das Akronym steht für „**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench“.

Die erste Version erschien 1986 für Macintosh-Computer. Heute gibt es die Entwicklungsumgebung außerdem für Windows und Linux. Vergleichbar entwickelte Hewlett-Packard (inzwischen Agilent) die visuelle Programmiersprache VEE. Haupt-Anwendungsgebiete von LabVIEW sind die Mess-, Regel- und Automatisierungstechnik. Die Programmierung erfolgt mit einer graphischen Programmiersprache, genannt **„G“**, nach dem Datenfluss-Modell. Durch diese Besonderheit eignet sich LabVIEW besonders gut zur Datenerfassung und -Verarbeitung. LabVIEW-Programme werden als Virtuelle Instrumente oder einfach VIs bezeichnet. Sie bestehen aus zwei Komponenten: das Frontpanel enthält die Benutzerschnittstelle, das Blockdiagramm den graphischen Programmcode. Dieser wird nicht von einem Interpreter abgearbeitet, sondern compiliert. Dadurch ist die Performance vergleichbar mit der anderer Hochsprachen.

LabVIEW wird laufend weiterentwickelt. Die derzeit aktuelle Version ist LabVIEW 2010 (Stand August 2010) als Nachfolger von LabVIEW 2009.

| LabVIEW | |
|---|---|
| **Entwickler** | National Instruments |
| **Aktuelle Version** | 2010 |
| **Betriebssystem** | Mac OS X, Windows, Linux |
| **Kategorie** | Programmiersprache |
| **Lizenz** | proprietär |
| **Deutschsprachig** | ja |
| LabVIEW (http://www.ni.com/labview/d/) | |

## Inhaltsverzeichnis



## Programmiermethode

Funktionsblöcke werden in LabVIEW (genau wie vollständige Programme) als Virtuelle Instrumente (VIs) bezeichnet. Dies kommt daher, dass prinzipiell jedes Programm als Unterprogramm (Sub-VI) in einem anderen verwendet werden kann, bzw. jedes Sub-VI auch eigenständig lauffähig ist. Aufgrund des Datenfluss-Konzeptes waren bis zu Version 8.6 rekursive Aufrufe grundsätzlich nicht möglich, mit etwas zusätzlichem Aufwand lassen sich aber auch Rekursionen verwirklichen. [1]. Seit Version 9.0 kann ein ablaufinvariantes VI sich selbst als SubVI enthalten und damit rekursiv aufrufen.[2]

Der Programmierer verbindet VIs mit Verbindungslinien (Drähten) und definiert damit den Datenfluss. Jedes VI kann dabei Ein- und Ausgänge besitzen. Die Ausführung eines VIs beginnt, wenn alle Eingangsdaten vorhanden sind; die Ergebnisse liegen erst dann an den Ausgängen an, wenn das gesamte Unterprogramm abgearbeitet ist. Auf diese Weise wird die Abarbeitungsreihenfolge der Schritte durch Datenabhängigkeiten definiert. Eine vordefinierte Reihenfolge (z. B. „von rechts nach links“) gibt es nicht.

Besitzt ein Sub-VI keine Eingänge so wird es bei Programmstart ausgeführt, besitzt es keine Ausgänge so werden die Ergebnisdaten entweder verworfen oder auf einem anderen Weg „verwertet“ (z. B. Schreiben auf Festplatte oder Netzwerk, Ausgabe auf Peripheriegeräte). Genauso kann ein Sub-VI ohne Eingänge Daten von Peripheriegeräten erhalten oder auch selbst generieren (z. B. Zufallsgenerator).

Sub-VIs können beliebig tief verschachtelt werden. Viele der LabVIEW-eigenen Funktionen sind ihrerseits normale VIs, die auch vom Programmierer bearbeitet werden können (wenngleich dies in der Regel nicht zu empfehlen ist). Letztlich basieren alle VIs auf einer Reihe grundlegender Funktionen, sogenannter Primitives, die sich nicht als VIs öffnen lassen.

Viele VIs und Primitives in LabVIEW sind polymorph, d. h. ihre Funktionalität passt sich an den Typ der übergebenen Daten an. Beispielsweise kann die Build-Array Funktion für die Erstellung jeglicher Felder genutzt werden, d. h. Strings, Integer oder auch Arrays und Cluster. Es ist auch möglich, eigene polymorphe VIs zu erstellen. Letztlich handelt es sich hierbei um eine Sammlung mehrerer VIs mit unterschiedlichen Datentypen an den Ein- und Ausgängen.

Datenquellen und Datensenken können mit Anzeige- und Bedienelementen auf dem Frontpanel verknüpft sein. So kann z. B. eine Zahleneingabe mit einem Drehknopf und eine Ausgabe einer booleschen Variablen mit einer Leuchtdiode realisiert werden.

Bei sehr großen und umfangreichen Projekten ist es (wie in jeder Programmiersprache) extrem wichtig, von Anfang an eine durchdachte Struktur zu verwenden und den Code zu modularisieren. Durch den vorhandenen Projektmanager (ab V8.0) wird dies um einiges erleichtert. Die Verwaltung einer großen Anzahl an VIs sowie externer Dateien ist dadurch um vieles übersichtlicher. Auch die major und minor Versionsverwaltung gestaltet sich hiermit wesentlich einfacher. Eine wesentliche Neuerung (ab V8.20) besteht darin, objektorientiert programmieren zu können. Klassen und Attribute sowie deren Methoden können natürlich auch vererbt werden.

LabVIEW Robotics 2009 enthält alle notwendigen Werkzeuge für den Entwurf eines anspruchsvollen Robotersystems. Teil des Softwarepakets LabVIEW Robotics ist das LabVIEW Robotics Module, welches mit LabVIEW eingesetzt werden kann und eine umfassende Robotikbibliothek mit Anbindungsmöglichkeiten an Standardrobotiksensoren und -aktoren, grundlegenden Algorithmen für den intelligenten Betrieb sowie Wahrnehmungs- und Motorsteuerungsfunktionen für Roboter und autonome Fahrzeuge umfasst.

## Vorteile

- Eine wichtige Konsequenz LabVIEWs graphischer Programmierung ist die Einfachheit, mit der in LabVIEW parallele Abläufe programmiert werden können. Es reicht, zwei Sub-VIs ohne Datenabhängigkeit nebeneinander zu legen, um sie gleichzeitig mit Multithreading abzuarbeiten. Man muss allerdings, ähnlich wie in text-basierten Programmiersystemen, auf mögliche Race Conditions achten und, wo nötig, Ressourcen sperren. Zur Synchronisierung bzw. Kommunikation zwischen mehreren Threads stehen verschiedene Möglichkeiten zur Verfügung (z. B. Semaphoren, Melder, Warteschlangen).
- Das Frontpanel von LabVIEW ist ein sehr bequemes Mittel, um Programme mit guter grafischer Bedieneroberfläche zu erstellen. Bei allen Programmierarbeiten in LabVIEW muss der Programmierer prinzipiell keinen Text eingeben, außer Beschriftungen von Gestaltungselementen.
- Die graphische Darstellung des Programmablaufs erhöht zumindest bei nicht zu umfangreichen Vorhaben die Lesbarkeit deutlich. Insbesondere Naturwissenschaftler und Techniker verstehen die Programmlogik meist recht schnell und können Software damit an ihre konkreten Bedürfnisse anpassen.
- Die je nach Lizenz mitgelieferten umfangreichen Funktionsbibliotheken decken insbesondere die Datenanalyse und Mathematik sehr weitgehend ab. Aber auch die Ansteuerung von zusätzlichen (auch externen) (Mess-)Geräten und Systemfunktionen ist gut gelöst.
- Über die unterstützten Kommunikationsprotokolle und Verbindungstechniken ist es möglich, auch weit entfernte Geräte (z. B. an unzugänglichen Stellen oder in anderen Ländern) zu steuern und zu nutzen. Hier kommt unter anderem TCP zum Einsatz.
- LabVIEW ermöglicht seit der Version 8.6.1 auch die Programmierung auf/von Mikrocontrollern und DSPs. Es unterstützt auch einige Echtzeitbetriebssysteme.
- Ab Version 2009 bietet LabVIEW die Möglichkeit der parallelen Programmierung von Multicore-Prozessoren und FPGAs und den Zugriff auf Wireless-Technologien.

## Nachteile

Neben den genannten Vorteilen hat die graphische Programmierung gegenüber der textbasierten auch Nachteile:

- LabVIEW-Programme lassen sich nur mit der originalen LabVIEW-Entwicklungsumgebung bearbeiten, an deren Funktionsumfang man gebunden ist. Allerdings lassen sich Funktionen aus dynamischen Bibliotheken oder ActiveX-Objekte nutzen. Der Plattformunabhängigkeit geschuldet sind Hindernisse in der Gestaltung von Benutzeroberflächen, so werden Windows-Hotkeys nicht unterstützt, und das Verhalten von Accelerator-Keys entspricht nicht exakt dem Verhalten des Betriebssystems (hier: Fokus-Verlust). Die Unicode-Unterstützung ist unzureichend.
- Ausführbare LabVIEW-Programme können vom Entwicklungssystem zwar erstellt werden, erfordern jedoch die Installation einer Laufzeitumgebung auf dem Zielsystem (vergleichbar mit der Installation des .NET-Frameworks für .NET Applikationen). Bei Verwendung von bestimmten Zusatzmodulen, wie z. B. IMAQ Vision, ist zudem eine kostenpflichtige Lizenz pro Zielplattform notwendig.
- Die Prinzipien moderner Objektorientierung versucht National Instruments mit neueren LabVIEW Versionen zwar nachzubilden, jedoch gelingt dies bisher nur unzureichend. Zudem zeigt die Programmierung gegen große bestehende Klassenhierarchien wie das Microsoft .NET-Framework die Grenzen im Umgang mit grafischen Zugriffsknoten auf - mit textbasierten Programmiersprachen, zum Beispiel C#, sind dieselben Aufgaben i. d. R. schneller programmatisch zu lösen.
- Kleine Änderungen können aufwendige Neustrukturierungen nach sich ziehen, wenn das Schaffen von Raum auf dem Blockdiagramm durch Verschieben geschieht, da dann die Drähte und Symbole oftmals neu geordnet werden müssen, um die Übersichtlichkeit wiederherzustellen. Dieses Problem kann durch Strukturierte Programmierung gemildert werden (insbesondere durch konsequente Verwendung von Sub-VIs).
- Der einfache Einstieg in die LabVIEW-Programmierung verleitet dazu, die ordentliche Planung des Projektes zu vernachlässigen.

## Literatur


- Wolfgang Georgi; Ergun Metin: *Einführung in LabVIEW*. Fachbuchverlag Leipzig im Carl Hanser Verlag, 4. neu bearbeitete Auflage 2009, ISBN 978-3-446-41560-7
- Bernward Mütterlein: *Handbuch für die Programmierung mit LabVIEW. Mit Studentenversion LabVIEW 8 (Gebundene Ausgabe)*, 1. Auflage April 2007, ISBN 3-8274-1761-9


## Referenzen


1. Rekursion in LabVIEW (english) (http://digital.ni.com/public.nsf/allkb/7140920082C3AC15862572840015A81E)

2. Verwendung der Rekursion bei VIs (english) (http://zone.ni.com/devzone/cda/tut/p/id/9387)

Von „http://de.wikipedia.org/wiki/LabVIEW“
Kategorien: Programmiersprache | Messdatenerfassung