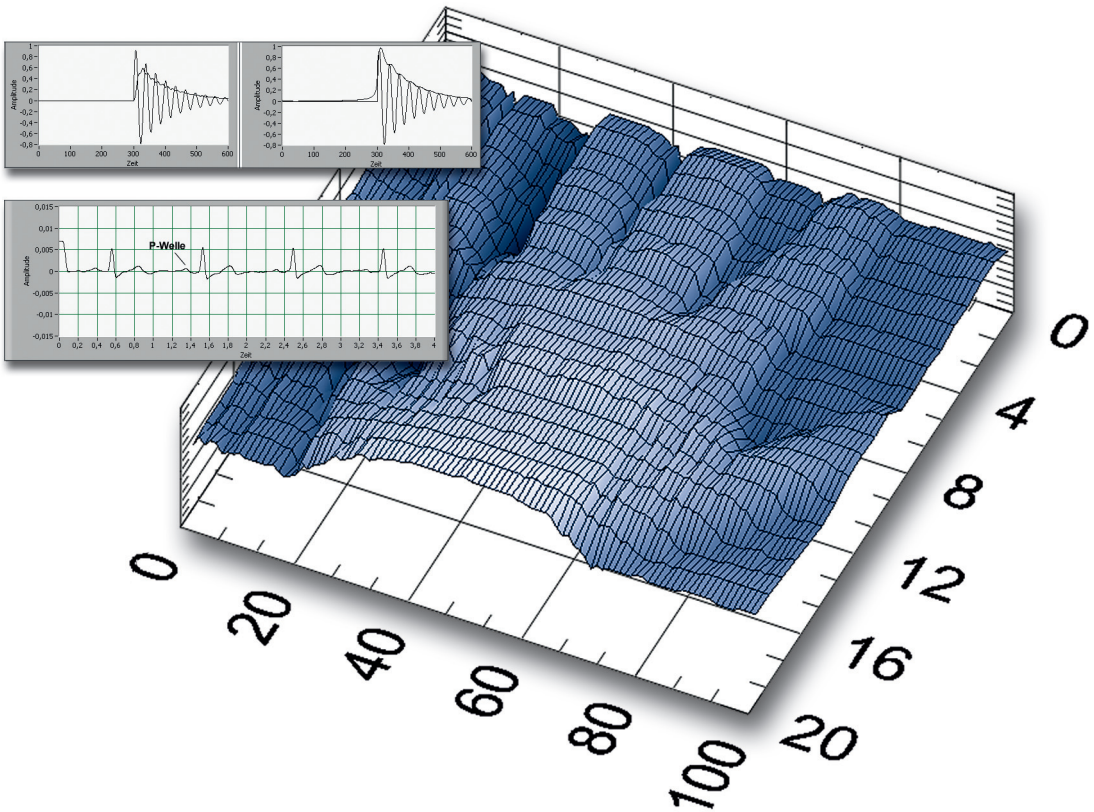
FRANZIS
PC & ELEKTRONIK

Friedrich Plötzeneder / Birgit Plötzeneder



Praxiseinstieg LabVIEW

Eine Einführung in die Praxis in 12 Experimenten

Vorwort

In jeder Firma, jeder Schule oder jedem Institut gibt es jemanden, der jedem hilft und alles kann. Für alle Sonderfälle oder Probleme hat er (oder sie) den passenden Kleber, eine Bezugsquelle oder sogar schon eine Lösung in der Schublade.

LabVIEW ist dieser Praktiker in der PC-Messtechnik. Es hat viele Algorithmen, ist leicht programmierbar und kann ausgezeichnet mit externer Hardware kommunizieren.

Das vorliegende Buch ist in zwei Abschnitte gegliedert:

Der erste Teil des Buchs, Kapitel 1–19, erklärt die Programmiersprache LabVIEW Schritt für Schritt. Die Kapitel sind mit instruktiven Beispielen abgeschlossen. Aufgrund des Umfangs von LabVIEW, das mehr als tausend ausprogrammierte Algorithmen hat, sind nur die wichtigsten und Sprachelemente besprochen.

Der zweite Teil des Buchs, Kapitel 20–30, zeigt, wie leicht man mit LabVIEW technische, physikalische oder mathematische Probleme unterschiedlichster Bereiche lösen kann. Die Experimente sind mit einfacher, oft bereits vorhandener Ausrüstung möglich. Sie können zu Hause oder im Labor einer Schule oder Hochschule ohne große Investition durchgeführt werden. Sie sind bezüglich der Durchführung und Mathematik vollständig beschrieben und erprobt.

Die Experimente sind als Laborübung an Hochschulen oder HTLs und für Lehramtsstudenten, Ingenieure und Autodidakten, die mit LabVIEW arbeiten, geeignet.

Dateien zum Buch können von der Website www.ploetzeneder-labview.com heruntergeladen werden. E-Mails an f.ploetzeneder@fh-wels.at oder b.ploetzeneder@gmail.com.

Geleitwort

Am Campus Wels der Fachhochschule Oberösterreich haben wir seit nunmehr über 15 Jahren die Software-Entwicklungsumgebung *LabVIEW* im Studiengang *Automatisierungstechnik* erfolgreich im Einsatz und schon sehr viele Praktiker für unseren Wirtschaftsraum damit ausgebildet. Bis heute ist meine persönliche Begeisterung als Lehrender für die LabVIEW charakterisierende intuitive, grafisch-visuelle Programmierung ungebrochen. Gemeinsam mit dem Autor des vorliegenden Lehrbuchs gelingt es uns in Wels, diese Begeisterung an Studierende weiter zu vermitteln.

Die Mächtigkeit von LabVIEW, der Umfang an wiederverwendbaren Programmbeispielen und Dokumentationen, aber auch der jährliche „Zuwachs" an neuer Funktionalität und Leistungsfähigkeit ist enorm. Das macht es Anfängern schwer, sich zu orientieren. Aber auch für erfahrene Praktiker, die nicht tagtäglich mit LabVIEW arbeiten, ist es nicht leicht, auf dem Laufenden zu bleiben. Vor diesem Hintergrund sehe ich in diesem Buch eine große Bereicherung. Es liefert einem fundierten und kompakten Einstieg und enthält eine große Anzahl konkret anwendbarer Beispiele guten Programmierstils.

In einer didaktisch ausgereiften Form wird dem Leser eine umfassende Einführung in die grafisch-visuelle Programmierung mit LabVIEW geboten. Ausgehend von elementaren Begriffen werden alle Kenntnisse vermittelt, die zur Lösung umfangreicher Problemstellungen notwendig sind. Das Buch ist für Anfänger ohne besondere Vorkenntnisse geeignet. Es bleibt dabei aber nicht an der Oberfläche, sondern vermittelt auch fortgeschrittene Programmiermethoden und enthält nützliche Laborexperimente als konkrete Beispiele. Selbst erfahrene LabVIEW-Anwender können mithilfe des Buchs Neues hinzulernen.

Positiv aufgefallen sind mir die sorgfältig erstellten realen Laborexperimente. Sie sind dank ihrer Übersichtlichkeit nicht nur leicht verständlich, sondern vermitteln auch die Funktionsvielfalt von LabVIEW anschaulich.

FH. Prof. Univ. Doz. Dipl. Ing. Dr. Karl Kellermayr

Fachbereichsleiter Informationstechnologie

Campus Wels

Fachhochschule Oberösterreich

Inhaltsverzeichnis

11 Unterprogramme

Unterprogramme sind ein wichtiges Konzept, um Komplexität in den Griff zu bekommen. Dabei trennt man Teile des Programms vom Gesamtprogramm und verwendet sie nur noch symbolisch. Sie kennen bereits viele Unterprogramme: Die meisten LabVIEW-Funktionen sind genau das! Die Nützlichkeit liegt nicht nur in der Übersichtlichkeit, sondern auch darin, dass man Programmteile sehr gut wiederverwerten kann.

11.1 Erstellen eines Unterprogramms

Eine wichtige Entwicklungsmethode für Software heißt *Bottom-up*. Bei dieser Methode wird das unterste Unterprogramm zuerst entwickelt und gründlich getestet. Danach werden die nächsten Ebenen der Unterprogramme programmiert, die auf die ersten zugreifen. Stellen Sie sich vor, sie müssten eine beliebig komplexe Funktion, die zwei Eingangsvariablen benötigt und zwei Werte ausgibt, in Ihr Programm einbauen. Eine sinnvolle Möglichkeit ist es, diese Funktion erst einmal fehlerfrei umzusetzen, und sie dann einfach einzufügen. In unserem Beispiel ist es nur eine einfache Addition/Subtraktion.

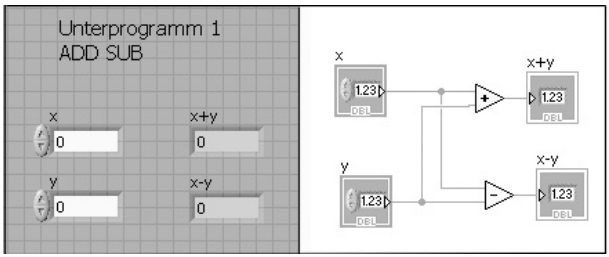


Abb. 11.1: Dieses Programm soll als Unterprogramm eingesetzt werden.

Wenn Sie dieses Programm jetzt als Unterprogramm wünschen, müssen Sie bestimmen, wie es im eigentlichen Programm (Hauptprogramm) aussieht. Dazu wählen Sie oben rechts im Frontpanel mit Rechtsklick *Symbol bearbeiten*.

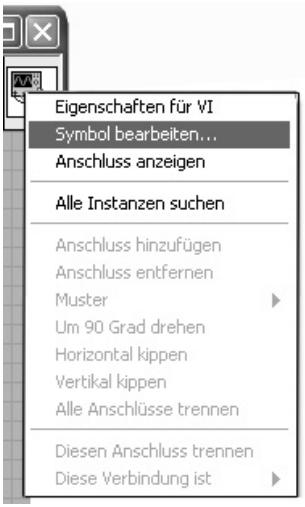


Abb. 11.2: Öffnen des Kontextmenüs am Symbol des Programms

In einer grafischen Programmiersprache hat ein Unterprogramm keinen Namen, sondern wird als Symbol eingesetzt. Gestalten Sie also ein neues Symbol. Wenn Sie Ihr Symbol erstellt haben, müssen Sie noch die Anschlüsse erstellen.

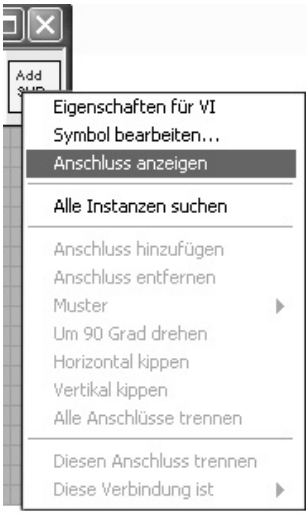


Abb. 11.3: Öffnen des Anschlussfensters

Sie sehen jetzt sehr viele Anschlussmöglichkeiten und benötigen genau vier Anschlüsse. Öffnen Sie das Kontextmenü und wählen Sie das Muster mit 4 Anschlüssen aus.

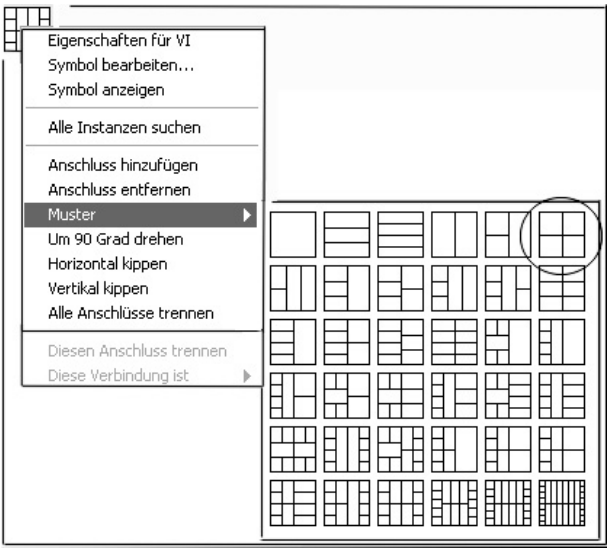


Abb. 11.4: Auswahl des Anschlussmusters

Schließen Sie die Frontpanelelemente am Anschlussmuster an. Klicken Sie das Frontpanel und das Anschlussmuster in der angegebenen Reihenfolge 1, 2, 3, 4, 5, 6, 7, 8 an. Sie wählen also jeweils die Variable und danach den entsprechenden Anschluss aus.

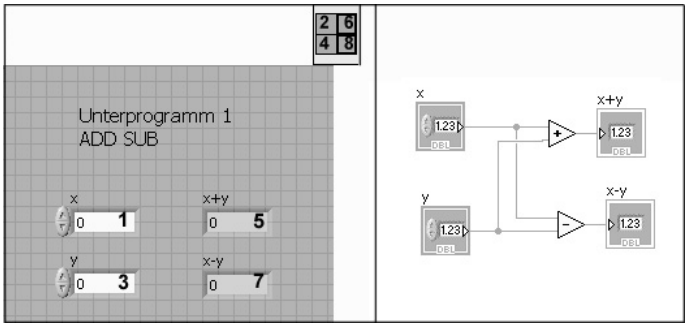


Abb. 11.5: Zuordnung von Frontpanelelementen und die Anschlüsse im Unterprogramm

Speichern Sie über *Datei >> Speichern unter* das Programm unter dem Namen *UP1.vi*. Erstellen Sie ein neues Programm mit zwei numerischen Eingaben und einem Zeigerinstrument.

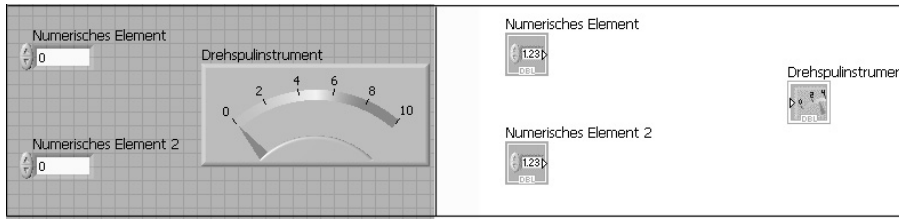


Abb. 11.6: Ein neues Programm wird das Hautprogramm.

Gehen Sie über das Blockdiagramm und öffnen Sie die Funktionspalette (rechte Maustaste):

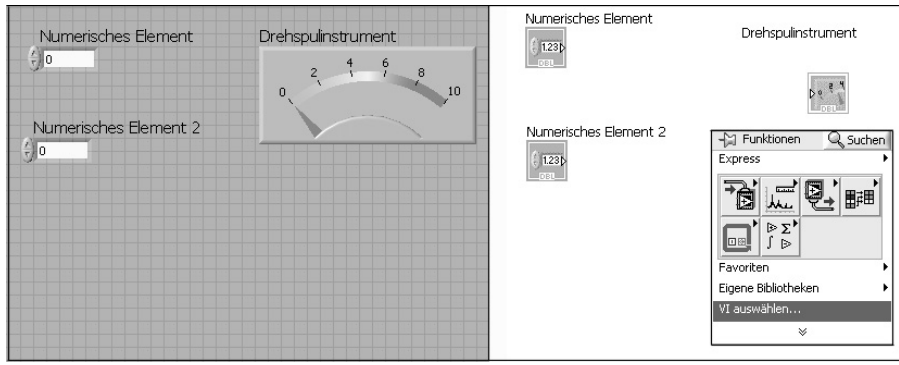


Abb. 11.7: Wählen des eigenen Unterprogramms in der Funktionspalette

Wählen Sie das Unterprogramm *UP1* aus und setzen Sie es in das Hauptprogramm ein.

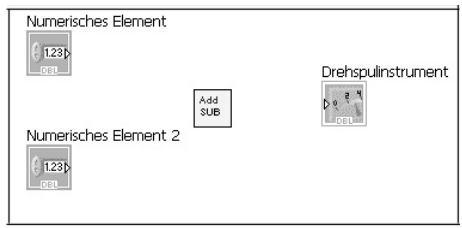


Abb. 11.8: Eingesetztes Unterprogramm

Verbinden Sie das Unterprogramm mit den Frontpanelelementen und testen Sie das Programm.

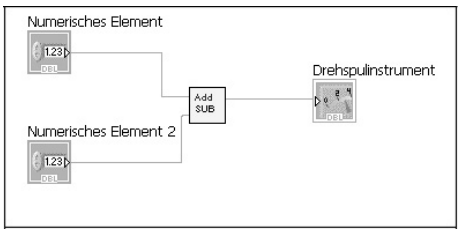


Abb. 11.9: Hauptprogramm mit Unterprogramm nach Verdrahtung

11.2 Automatisches Erstellen eines Unterprogramms im Hauptprogramm

Bei der Entwicklung eines LabVIEW-Programms kommt es vor, dass das Programm größer wird, als ursprünglich geplant. Schnell kommt der Wunsch auf, einen Teil des Hauptprogramms in ein Unterprogramm zu verwandeln. Dies wird in der Entwicklungsumgebung von LabVIEW auf folgende Weise unterstützt:

- Den Teil markieren, der in ein Unterprogramm verwandelt werden soll
- In der Symbolleiste *Bearbeiten >> SubVI erstellen* aufrufen

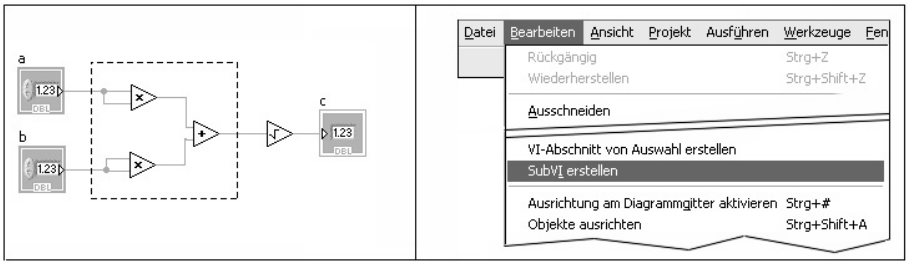


Abb. 11.10: Automatisches Erstellen eines Unterprogramms

Danach ist das Unterprogramm aus dem Hauptprogramm erstellt und schon in dieses eingesetzt.

11.3 Modi beim Aufrufen eines Unterprogramms

Das Aufrufen eines Unterprogramms kann auf unterschiedliche Arten durchgeführt werden. Beispielsweise können Frontpanels neu geöffnet werden (etwa für einen Anmeldedialog).

11.3.1 Frontpanel des Unterprogramms bei Aufruf öffnen

Beispiel:

Es soll ein Programm erstellt werden, das einen Anmeldedialog hat. Beim Start des Hauptprogramms soll das Frontpanel des Unterprogramms (der Dialog) geöffnet werden. Im Fenster des Unterprogramms kann der Name des Anwenders eingegeben werden. Nach einer Bestätigung soll das Fenster automatisch geschlossen werden und der String mit dem Namen ist vom Unterprogramm an das Hauptprogramm zu übergeben.

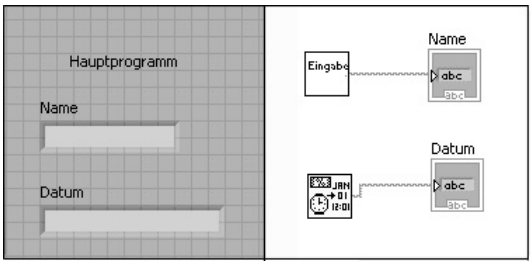


Abb. 11.11: Hauptprogramm; nach dem Start wird automatisch das Frontpanel des Unterprogramms geöffnet.

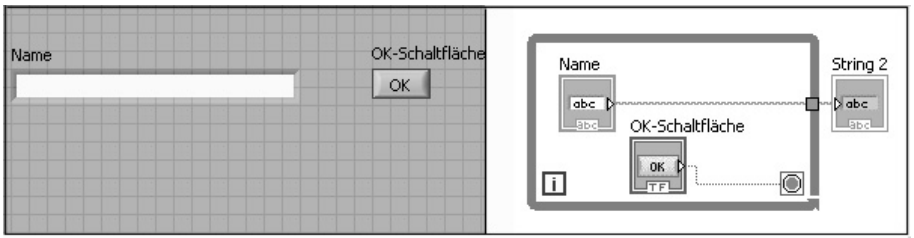
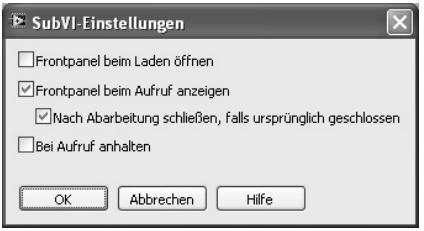


Abb. 11.12: Unterprogramm-Eingabe, das vom Hauptprogramm geöffnet wird.

In die String-Eingabe des Unterprogramms kann der Name eingegeben werden. Die *While*-Schleife wird so lange ausgeführt, bis auf den Button *OK* gedrückt wird. Danach wird das Unterprogrammfenster geschlossen und der String an das Hauptprogramm übergeben.

Wie wird erreicht, dass bei Aufruf des Unterprogramms das Frontpanel geöffnet und nach Ausführung wieder geschlossen wird? Dazu ist im Hauptprogramm mit der rechten Maustaste das Unterprogramm anzuklicken. Im entstehenden Kontextmenü kann man die *SubVI*-Einstellungen konfigurieren. Wie ist ein Unterprogramm zu konfigurieren, damit bei seinem Aufruf das Frontpanel geöffnet wird?

Tabelle 11.1: Konfiguration des Unterprogramms; Bedingung für das Öffnen des Frontpanels

Kontextmenü des Unterprogramms	Einstellung des Unterprogramms a) Öffnen bei Aufruf b) Schließen bei Beendigung
 Konfiguration	 Öffnen

11.3.2 Ablaufinvariante Ausführung eines Unterprogramms

Wird ein Unterprogramm mehrmals in ein Hauptprogramm eingesetzt, kann es zu einem besonderen Problem kommen. Soll beim Verlassen des Unterprogramms ein Wert einer Variablen erhalten bleiben, muss auch für jedes eingesetzte Unterprogramm ein Speicher bereitgestellt werden.

Spezialbeispiel aus der Elektronik:

Ein Binärzähler besteht aus zwei T-Flipflops. Jedes Flipflop soll in Form eines Unterprogramms realisiert werden.

Hinweis zum T-Flipflop: Der Ausgang Q ändert sich bei einer positiven Flanke am T-Eingang. *T-Eingang* steht dabei für „Trigger-Eingang“.

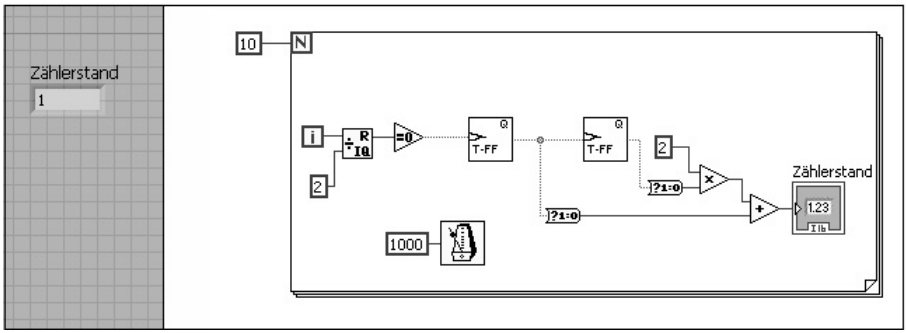


Abb. 11.13: Hauptprogramm eines Binärzählers

Beobachtet man im Hauptprogramm den Zählerstand, erkennt man, dass man einen Rückwärtszähler vor sich hat.

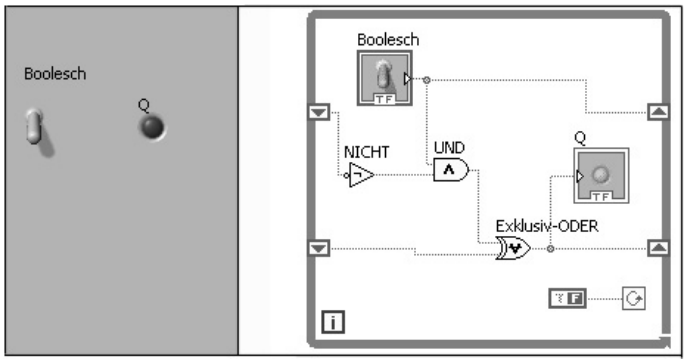


Abb. 11.14: T-Flipflop als Unterprogramm

Das Unterprogramm, das im Hauptprogramm zweimal eingesetzt ist, muss so konfiguriert werden, dass jedes Unterprogramm einen eigenen Datensatz erhält. Das ist bei der Konfiguration nach dem Mode *Ablaufinvariant* der Fall. Nur dadurch kann sich jedes Flipflop den Zustand auch wirklich merken und ein mehrstufiger Zähler ist realisierbar.

Konfigurieren des Unterprogramms:

Mit rechter Maustaste (Kontextmenü) am Symbol des Unterprogramms klicken.

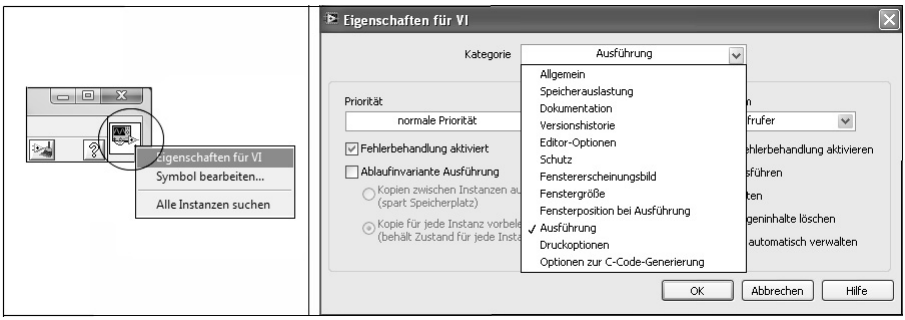


Abb. 11.15: Eigenschaften für VI und danach Ausführungsmode wählen

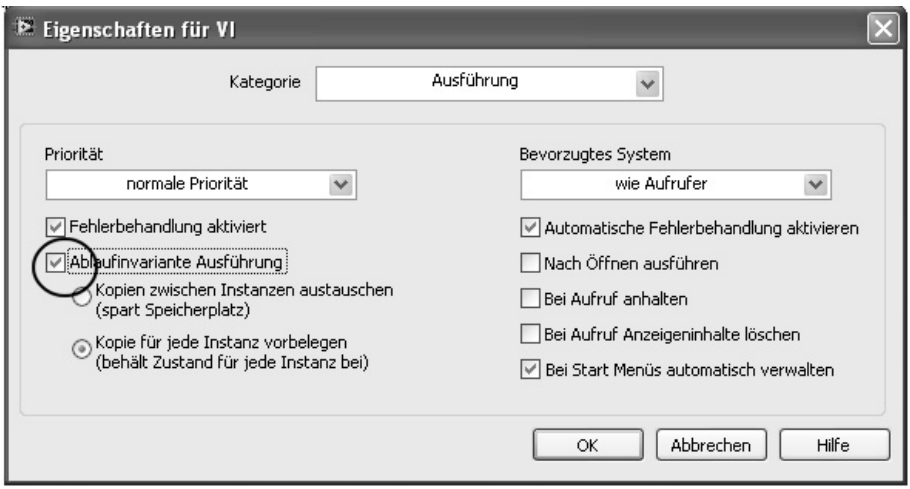


Abb. 11.16: Ablaufinvariante Ausführung wählen.

Ablaufinvariant (oder wiedereintrittsfähig) wird in der englischen Version als reentrant bezeichnet.

21 Schrittmotoransteuerung

Zur Ansteuerung eines Schrittmotors ist nur ein einziger Treiber-IC nötig, der direkt an die USB-6008 angeschlossen wird. Damit können seine Eigenschaften im Labor sehr gut untersucht werden.

21.1 Hintergrund

Der Vorteil eines Schrittmotors ist, dass er ohne Sensor eine genaue Position anfahren kann. Daher wird er vor allem für Präzisionsgeräte häufig eingesetzt. Im Gegensatz zum Gleichstrommotor hat er ein Haltemoment (wirkt also im Stillstand Bewegung von außen entgegen). Von der Wirkungsweise ist der Schrittmotor (englisch: *Stepper*) ein Synchronmotor, d. h., seine Bewegung ist zur Wechselspannung synchron. Meistens ist der Rotor (der sich bewegende Läufer) ein Permanentmagnet, aber es sind auch weichmagnetische Ausführungen möglich. Der Stator (umgebender, nicht drehbarer Teil) besteht im Wesentlichen aus zwei Spulen, die als Elektromagnete wirken. Diese Spulen werden in Schritten angesteuert und ziehen den Rotor im Drehsinn um einen Schritt weiter. Es gibt mehrere Arten, den Motor zu betreiben. Die wichtigsten sind der Wave-Mode, der Two-Phase-on-Mode und der Half-Step.

21.1.1 Wave-Mode

Bei dieser Ansteuerungsart ist immer eine Spule stromlos.

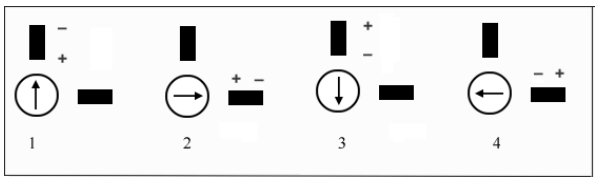


Abb. 21.1: Wave-Mode; die Magnetisierung wird durch den Pfeil im Rotor dargestellt.

1. An der senkrechten Spule wird eine Spannung mit der eingezeichneten Polarität angelegt und der Pfeil nach oben gezogen.
2. Die horizontale Spule zieht die Magnetisierung, die durch den Pfeil dargestellt wird, nach rechts. Der Motor hat sich um 90° im Uhrzeigersinn gedreht.
3. Die senkrechte Spule wird mit umgepolter Spannung betrieben und der Pfeil nach unten gezogen. Eine weitere Drehung im Uhrzeigersinn entsteht.

4. Durch Umpolen der Spannung in horizontaler Wicklung wird die nächste viertel Umdrehung bewirkt.

21.1.2 Two-Phase-on-Mode

Hier fließt immer durch zwei Wicklungen gleichzeitig Strom. Das bewirkt, dass der Rotor im Fall A nach rechts oben zeigt. Der Motor hat bei dieser Ansteuerung ein größeres Drehmoment, aber auch einen größeren Stromverbrauch, da immer zwei Wicklungen angeschaltet sind.

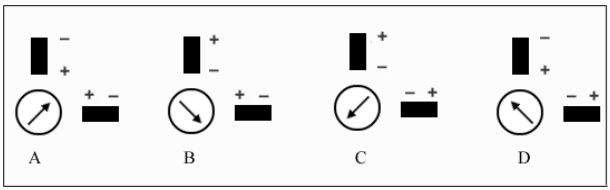


Abb. 21.2: Two-Phase-on Mode

21.1.3 Half-Step-Mode

Dieser Modus ist eine Kombination von Wave- und Two-Phase-on-Mode. Die Drehung des Rotors ist bei einem Schritt ist nur halb so groß wie bei den anderen Ansteuermethoden.

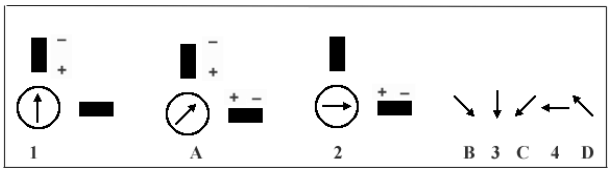


Abb. 21.3: Half-Step-Mode

21.2 Versuchsaufbau

Bei einem realen Schrittmotor ist die erläuterte Drehung von 90° oder 45° pro Schritt schon zu groß. Kleinere Schritte erhält man, wenn man mit den Spulen jeweils mehrere Nord-Süd-Pol-Paare erzeugt. Der im Versuch verwendete Schrittmotor hat eine Auflösung der Schritte von 7,5°. Dies kann man schon durch Drehen mit der Hand nachprüfen (drehen: man spürt ein Ruckeln).

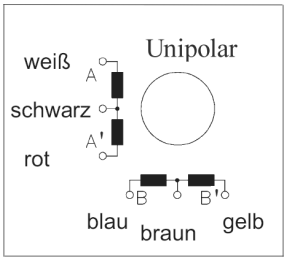


Abb. 21.4: Anschlussbild des verwendeten Schrittmotors nach Datenblatt von Nanotec

Bei diesem Schrittmotor hat jede Wicklung eine Mittelanzapfung (schwarz und braun). Legt man an diese +12 V an, kann man mit einem einfachen Transistor den weißen oder roten Abschluss auf Masse ziehen und damit das Magnetfeld umpolen. Mit der zweiten Spule ist die gleiche Methode an den Anschlüssen braun und blau/gelb möglich. Diese Typen von Schrittmotoren werden als *Unipolar* bezeichnet, da keine Umpolung der Spannung notwendig ist. Im angegebenen Versuch werden nur die Wicklungen schwarz/rot und braun/gelb verwendet, dafür wird aber mit einer Spannung verschiedener Polarität gearbeitet. Der Unipolarmotor wird also im Bipolarbetrieb eingesetzt. Der Strom, den die USB-6008 direkt am Port (Digitalausgang) ausgibt, ist allerdings viel zu klein, um den Schrittmotor anzutreiben. Deswegen ist ein Treiber-IC notwendig, um den nötigen Strom durch den Motor zu schicken. Man verwendet den Baustein L293D, der vom 5-V-Spannungsausgang der Messkarte versorgt wird.

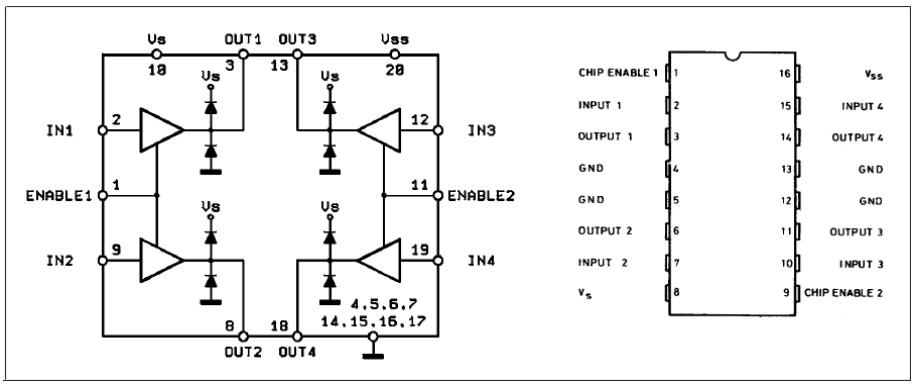


Abb. 21.5: L293-IC zur Ansteuerung des Schrittmotors [6]

Obwohl der Motor für 12 V pro Wicklung ausgelegt ist, funktioniert er auch noch mit 5 V. Wie auch aus dem Bild ersichtlich ist, ist kein zusätzliches Netzgerät erforderlich. Der Wicklungswiderstand des Schrittmotors von 50 Ω garantiert, dass die Strombelastung für die USB-6008 nicht zu groß wird.

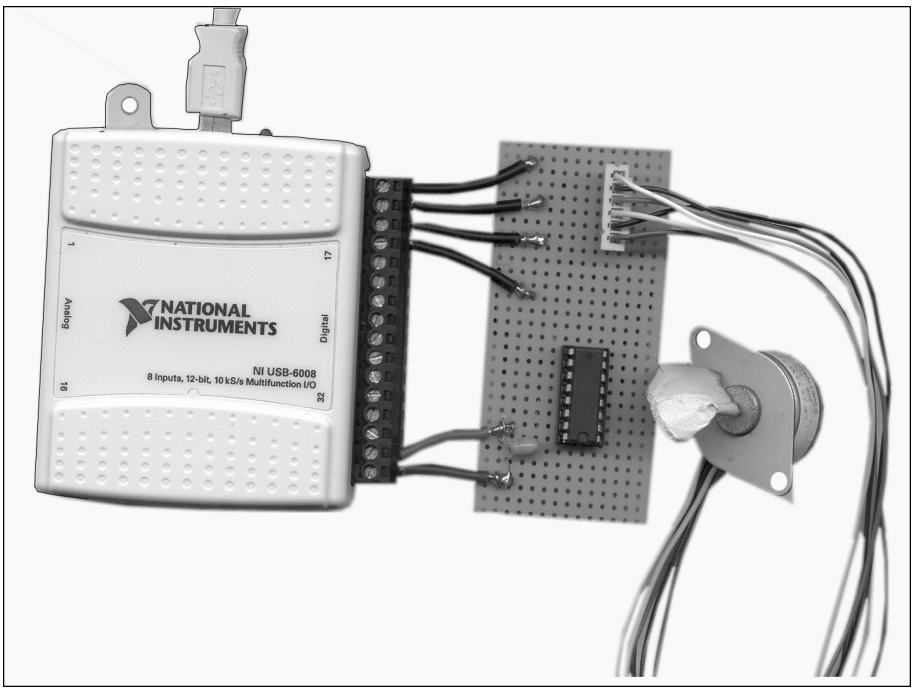


Abb. 21.6: Aufbau der Schaltung

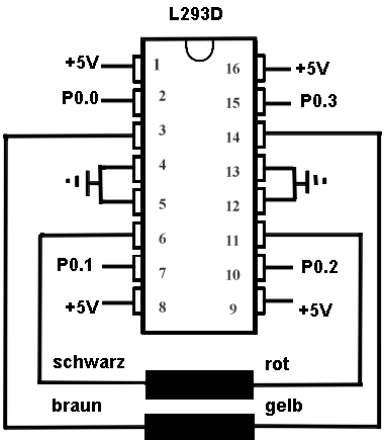


Abb. 21.7: Anschluss des Schrittmotors am Treiber-IC

21.3 Programme

21.3.1 Wave-Mode

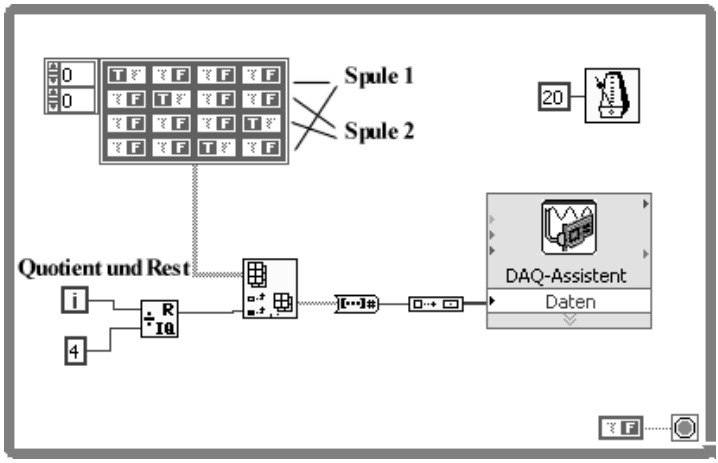


Abb. 21.8: Programm zum Ansteuern des Schrittmotors im Wave-Mode

Aus dem zweidimensionalen Array wird mit *Array indizieren* eine Spalte herausgeschnitten. Mit den Werten aus dieser Spalte werden die Spulen des Schrittmotors angesteuert. Zuerst erfolgt dazu mit *Boolesches Array nach Zahl* die Umwandlung in einen Integer. Danach wird mit *Array erstellen* ein Array gebildet, das die Integerwerte enthält (also Einsen und Nullen, statt T und F). Diese spezielle Art der Formatierung ist für die Digitalausgabe an einen Port notwendig, wenn man mit dem DAQ-Assistenten arbeiten möchte.

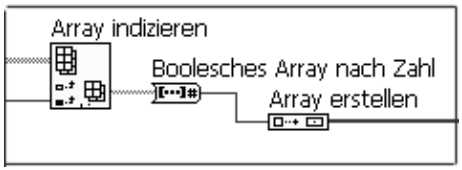


Abb. 21.9: Spaltenweise Ausgabe des 2-D-Arrays in den DAQ-Assistenten

Welche Spalte genau zur Steuerung verwendet wird, soll sich zyklisch ändern (0, 1, 2, 3, 0, 1, 2, 3, 0 usw.). Das erreicht man mit der *Modulo*-Funktion (Bildung des Rests bei Division). Bei der verwendeten Funktion mit Modulo 4 wird aus der Folge i = 0, 1, 2, 3, 4, 5, (von der Schleife stammend) am Iterationsterminal die Folge 0, 1, 2, 3, 0, 1, ...

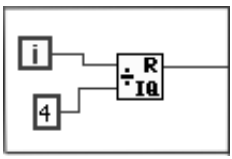


Abb. 21.10: Funktion Quotient und Rest; aus 0, 1, 2, 3, 4, 5, ... wird 0, 1, 2, 3, 0, 1, ...

Man betrachtet nun eine einzelne Spalte, also eine Steuerungsanweisung. Das oberste und das unterste Element (Zeilen 0 und 3 im zweidimensionalen Array) sind für die senkrechte Spule verantwortlich. Die mittleren Elemente (Zeilen 1 und 2) beeinflussen die horizontale Spule. Die Spalte 0 mit den Werten *T, F, F, F* bewirkt, dass der magnetisierte Rotor nach oben gezogen wird. Die horizontale Spule ist auf *F, F* gesetzt und somit stromlos. Das gleiche Ergebnis erhält man mit *T, T, T, F*, da auch hier die horizontale Spule keinen Spannungsunterschied hat und somit stromlos wird. Damit ist das erste Bild im Diagramm für den Wave-Mode realisiert. Spalte 1 hat die Werte *F, T, F, F*. Damit wird die senkrechte Spule stromlos und die horizontale Spule zieht den Rotor nach rechts. Dies entspricht dem zweiten Bild im Diagramm des Wave-Mode. Die beiden weiteren Fälle werden durch Umkehr der Stromrichtung aus den Fällen 1 und 2 gebildet.

Rücklauf

Eine Umkehr der Drehrichtung des Motors kann durch eine veränderte Tabelle oder durch einen Tabellenzugriff in umgekehrter Reihenfolge erreicht werden. Es werden dabei die Spalten 3, 2, 1, 0, 3, 2, 1, 0 … aus dem zweidimensionalen Array herausgeschnitten.

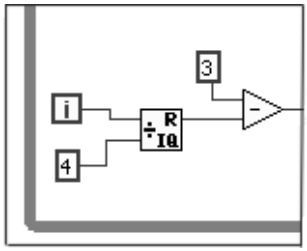


Abb. 21.11: Rücklauf des Motors durch veränderten Array-Zugriff

21.3.2 Two-Phase-on- und Half-Step-Mode

Der Two-Phase-on- und der Half-Step-Mode können einfach durch Austausch der Tabellen im Programm realisiert werden.

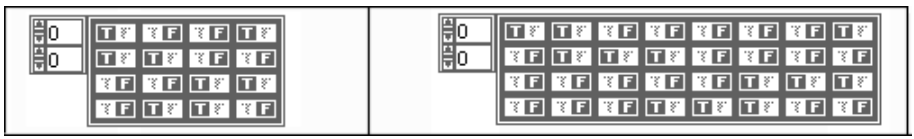


Abb. 21.12: Tabelle für den Two-Phase-on- und den Half-Step

Dabei ist im Half-Step-Mode bei der Auswahl der Spalten durch die Funktion *Array indizieren* statt Modulo 4 noch Modulo 8 zu wählen.

Praxistipp

Der verwendete Motor hat nicht wie in der schematischen Darstellung zwei Polpaare, sondern für jede Wicklung sechs Polpaare. Aus diesem Grund ist ein Schritt nicht eine Drehung um 90°, sondern um 90/6, also um 15°. Beim Half-Step-Mode ist der kleinste Drehwinkel dadurch 7,5°.

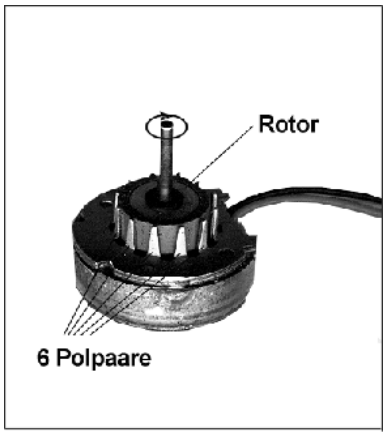


Abb. 21.13: Innenansicht des verwendeten Schrittmotors

Bezugsquelle des verwendeten Motors:

www.farnell.de

Hersteller: NANOTEC

Farnell Bestellnummer: 4743209

Herstellerbezeichnung: SP2575M0206-A

Stichwortverzeichnis

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

X

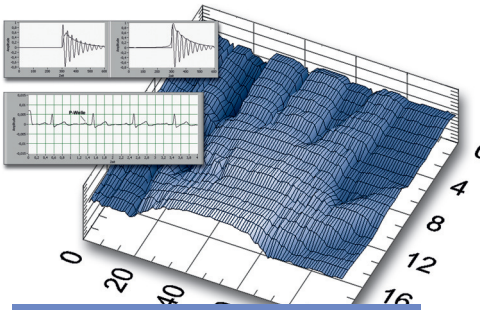
Z

FRANZIS
PC & ELEKTRONIK

Friedrich Plötzeneder / Birgit Plötzeneder

Praxiseinstieg

LabVIEW



Dieses Buch führt Anfänger und Fortgeschrittene Schritt für Schritt mit vielen Praxisbeispielen in die führende Programmiersprache für Messtechnik „LabVIEW" ein.

Der erste Teil des Buchs erklärt ausführlich und umfassend die Programmiersprache LabVIEW. Die Kapitel schließen jeweils mit instruktiven Beispielen. Aufgrund des Umfangs von LabVIEW, das mehr als tausend ausprogrammierte Algorithmen hat, beschränkt sich das Buch auf die wichtigsten Algorithmen und Sprachelemente, sodass Sie sich nicht mit unnötigem Ballast befassen müssen.

Der zweite Teil des Buchs zeigt, wie leicht man mit LabVIEW technische, physikalische und mathematische Probleme unterschiedlichster Bereiche lösen kann. Die Experimente sind mit einfacher, oft bereits vorhandener Ausrüstung möglich. Praxisrelevante Themen wie Messdatenerfassung, serielle Schnittstelle und Soundkartenprogrammierung werden ausführlich besprochen. Anhand von einfach durchzuführenden Experimenten wie dem Messen eines EKGs, der Ansteuerung eines Schrittmotors oder dem Bau eines 3-D-Scanners werden wichtige Kenntnisse für die Realisierung eigener Projekte vermittelt. Sie können ohne große Investition durchgeführt werden und sind hinsichtlich der Durchführung und Mathematik vollständig beschrieben und erprobt.

Die Experimente sind auch als Laborübung an Hochschulen sowie für Auszubildende, Ingenieure und Autodidakten, die mit LabVIEW arbeiten, geeignet.

Praktische Experimente :

- EKG
- Schrittmotoransteuerung
- Wechselrichter – Drehstrom aus dem Laptop
- Dehnungsmessstreifen
- Terminalprogramm
- Signalverarbeitung in der Praxis
- Akustisches GPS
- Bildverarbeitung mit zweidimensionaler Fourier-Transformation
- Temperaturverteilung in einem Ring
- 3-D-Scanner mit Laptop und Beamer
- Praktische Bildverarbeitung
- und viele mehr

ISBN 978-3-7723-4039-0

Euro **29,95** [D]