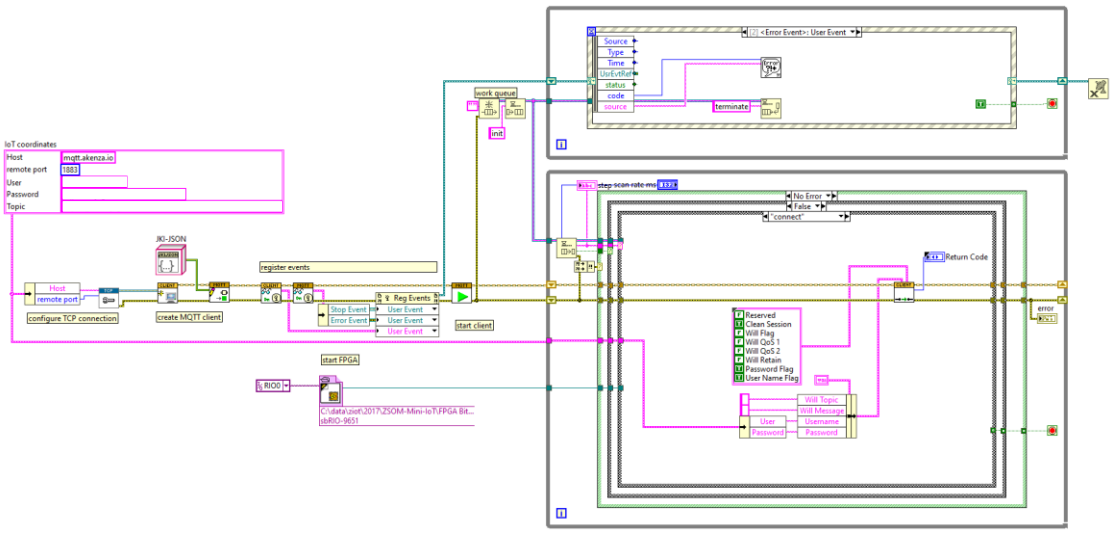
Whitepaper

Der Charme und Nutzen Grafischer Programmierung mit LabVIEW



Dieses Whitepaper ist ideal für alle, die einen einfachen und schnellen Einstieg in die grafische Programmierung suchen und sich mit LabVIEW und seinem datenzentrierten Prinzip vertraut machen wollen. Auch die Vor- und Nachteile dieser grafischen Methode werden angesprochen. Auf die Wichtigkeit der Community gehen wir genauso ein wie auf die Bedeutung der Übernahme von NI durch EMERSON. Und last but not least gibt es einen Ausblick, wie sich NI R&D die Zukunft von LabVIEW in der Welt smarter Software-Assistenten à la Copilot vorstellt.

Inhalt

1 Was ist der Unterschied zwischen LabVIEW und «G»? 2

2 Eine gut verständliche Analogie zur Realität 2

3 Eine «richtige» Programmiersprache? 3

4 Ein Graph und Daten als zugrunde liegende Architektur 4

5 Das Geheimnis hinter LabVIEW's Produktivität 4

6 Vor- und Nachteile von LabVIEW 5

7 Die Power der NI Community – it's ok to have fun! 6

8 Die Bedeutung der Übernahme von NI durch EMERSON 7

9 Die Zukunft von LabVIEW mit Generativer KI 7

10 Glossar 9

LabVIEW steht für "**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench". Es handelt sich um eine Entwicklungsumgebung, die für Mess-, Steuer- und Regeltechnikaufgaben ausgelegt wurde. Seit seiner ersten Veröffentlichung im Jahr 1986 auf dem Mac wurde LabVIEW als Schnitt-stelle zu Mess- und Automatisierungsgeräten entwickelt. Diese Sprache ist abstrakt, gleichzeitig hardwarenah und sie ist mit physikalischen Signalen grossgeworden. Im Laufe der Jahre entwickelte NI skalierbare und modulare Hardware: das PXI, das CompactRIO und das Singleboard RIO, um nur einige zu nennen. Das führte zur NI-Plattform, die Hard- und Software nahtlos verbindet und die Anwendungsentwicklung dank hoher Abstraktion messbar vereinfacht.

1 Was ist der Unterschied zwischen LabVIEW und «G»?

Der Schlüssel und die Stärke von LabVIEW liegt in «G», der grafischen Programmiernotation. «G» ist der auf Datenfluss ausgelegte grafische Code, den wir in der LabVIEW-Entwicklungsumgebung erstellen und dem ein mathematisches Netzwerk, ein Graph, zugrunde liegt. Viele betrachten LabVIEW und „G" als ein und dasselbe und das werden wir der Einfachheit halber nun auch tun.

Weitere Details zu LabVIEW findet der interessierte Leser in diesem WIKIPEDIA-Eintrag oder diesem Fachartikel.

2 Eine gut verständliche Analogie zur Realität

Schauen wir uns die LabVIEW-Terminologie und Grundphilosophie nun etwas genauer an. Jedes Programm in LabVIEW ist ein virtuelles Instrument: ein **VI**. Wenn ich also ein Programm mit dem Namen «**Energiemessung**» erstelle, heisst die Datei des Programms: **Energiemessung.vi**.

LabVIEW wurde von Beginn an von Ingenieuren für Ingenieure entwickelt. Daher schlagen viele Begriffe eine Brücke zum technischen Entwickleralltag, etwa einem Oszilloskop.

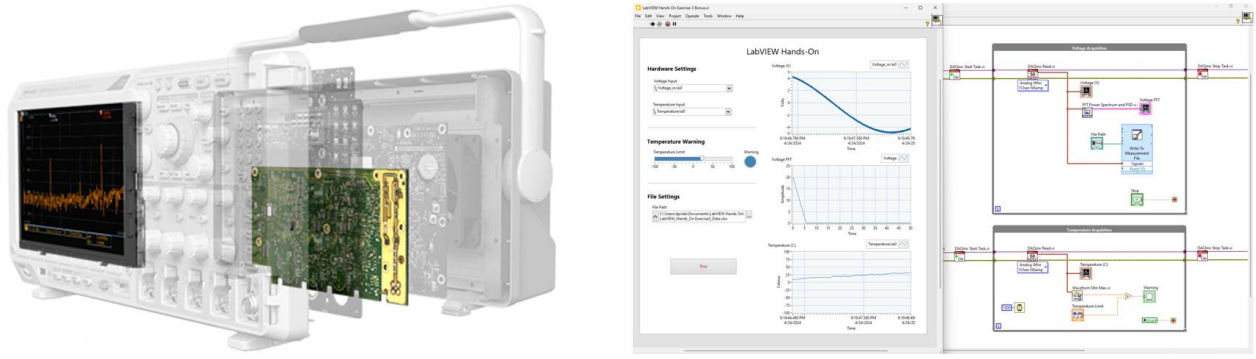


Bild 1 | Im realen Instrument (links, Bild:Tektronix) fliessen Elektronen in Drähten zwischen Bausteinen. Im virtuellen Instrument (rechts, Bild:NI) fliessen Daten zwischen Knoten.

Ein reales Oszilloskop hat Signaleingänge und einen Bildschirmausgang mit einem Bedienfeld. In LabVIEW gibt es verschiedene Arten von Eingängen und Ausgängen, die im Allgemeinen das imitieren, was wir bei einem Messgerät sehen können. Im aktuellen Beispiel etwa zeigt ein Frontpanel das, was ich auch auf dem Bedienfeld des realen Oszilloskops sehe: Drehknöpfe, Taster und einen Bildschirm mit dem Signal drauf.

Werfen wir mal einen Blick unter die Haube. Im realen Instrument fliessen Elektronen in Drähten von Baustein A zu Baustein B. Wie beim realen können wir auch beim virtuellen Instrument in sein Inneres blicken. Im virtuellen Instrument gibt es ebenfalls Drähte, in denen jedoch anstelle von Elektronen Daten von Knoten zu Knoten fliessen. In LabVIEW ist der eigentliche Code analog zur Hardware ein Schaltplan, der nach dem Datenfluss funktioniert.

3 Eine «richtige» Programmiersprache?

Ein LabVIEW Diagramm, so nennen wir diesen Schaltplan, wird laufend in Maschinencode übersetzt und per Knopfdruck auf dem Zielsystem ausgeführt. Auch wenn es grafisch und nicht textorientiert ist, greift LabVIEW trotzdem auf bekannte, klassische Programmierprinzipien zurück. Beispielsweise sind auch hier bekannte Konstrukte wie Datentypen, Variablen, Schleifen, Rekursion, Verzweigungen, Eventhandling und objektorientierte Ansätze zu finden. Mehr noch: an jeder Stelle im grafischen Code können wir direkt auf unterlegte Hardware oder Funktionen des Betriebsystems zugreifen, z.B. «Warte x Millisekunden».

In Summe unterscheidet sich LabVIEW etwa von rein modellbasierten Ansätzen oder Low- und No-Code.

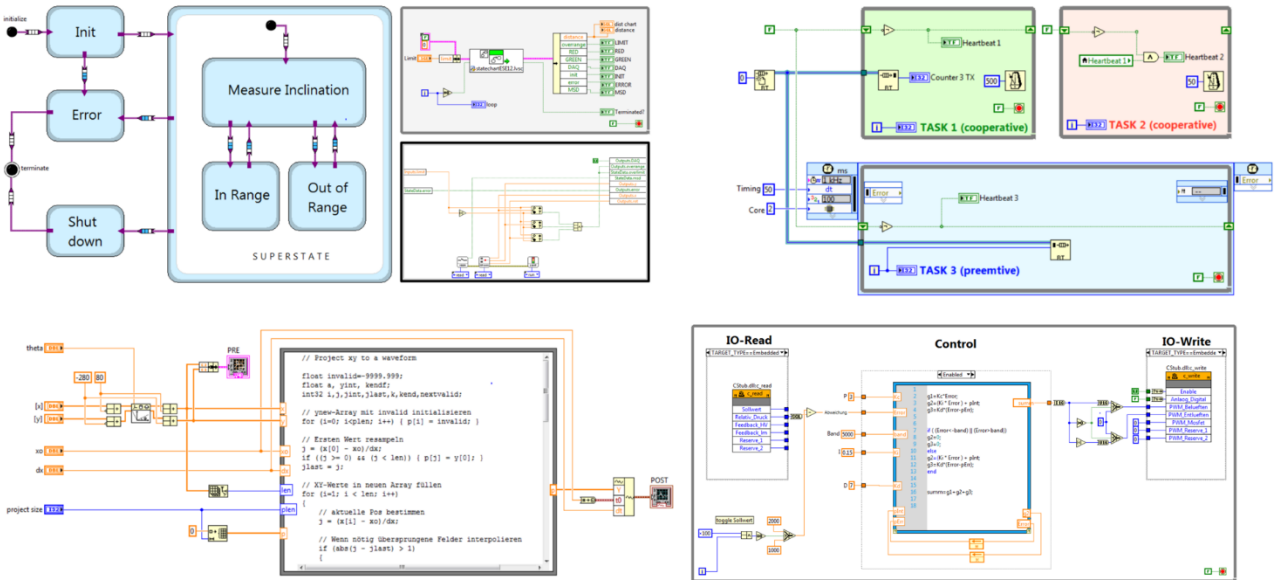


Bild 2 | Vielleicht ist es angesichts der vielen Programmiermodelle, die sich flexibel in LabVIEW einbetten und ausführen lassen, die falsche Frage?

4 Ein Graph und Daten als zugrunde liegende Architektur

LabVIEW wird also datenzentriert ausgeführt und bestimmt Daten als das Hauptkonzept hinter jedem Programm. Anstelle sequentieller Instruktionen bestimmt der Datenfluss die Reihenfolge der Ausführung. Dahinter steht ein mathematisches Konstrukt: ein Petrinetz, welches Graphen mit Knoten und Kanten zur Darstellung von Prozesszuständen nutzt. Das Prinzip dahinter ist demnach eine mächtige Kombination aus Graphenmodell, Datenzentrierung und Echtzeit, womit wir gleich mehrere Zeitdomänen beherrschen.

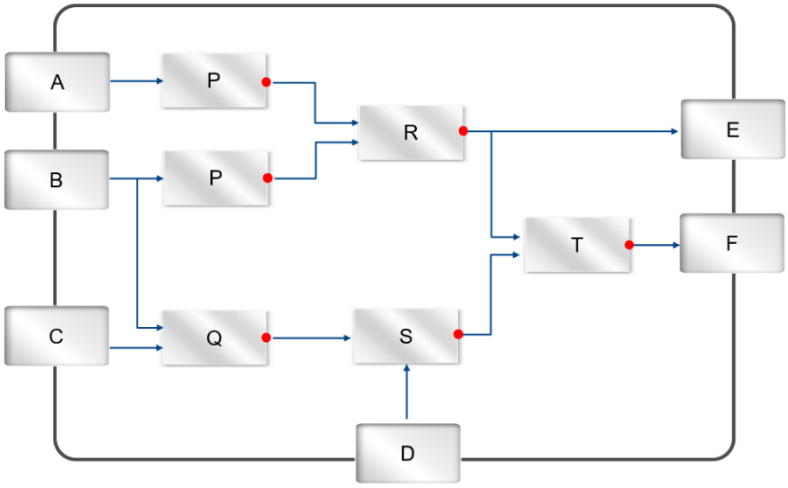


Bild 3 | Hinter LabVIEW arbeitet ein mächtiges Petrinetz aus Knoten und Kanten. Es wird von Natur aus parallel ausgeführt.

Darüber hinaus ist LabVIEW über die gesamte Wertschöpfungskette einsetzbar:

1. Machbarkeitsprüfung
2. Minimum Viable Products
3. Prototyp
4. Serienprodukt
5. Testing

Dies macht es zu einem guten Beispiel, wie eine über Jahrzehnte gereifte Sprache und Entwicklungsumgebung mit heutigen, modernsten Trends und Bedürfnissen mithalten kann.

5 Das Geheimnis hinter LabVIEW's Produktivität

Das Programmieren in LabVIEW geht messbar schneller im Vergleich zu textbasierten Sprachen. Der Grund dafür liegt in der Arbeitsweise unseres Gehirns und wie es grafischen Code wahrnimmt. Unsere Neuronen und Synapsen zeigen nämlich eine gewisse Ähnlichkeit mit den Knoten und Kanten eines LabVIEW-Programms. Ein grafisches Blockschaltbild, welches visuell den Fluss von Daten zwischen Knoten repräsentiert, ist demnach für einige Softwareentwickler intuitiver als einzelne, aufeinanderfolgende Textanweisungen.

6 Vor- und Nachteile von LabVIEW

Vorteile

1. Die grafische Oberfläche ist flexibel und einfach zu bedienen. Die meisten Ingenieure und Wissenschaftler können die Anwendung schnell erlernen.
2. LabVIEW bietet eine universelle Plattform und mächtige Bibliotheken und Frameworks für zahlreiche Anwendungen in unterschiedlichen Bereichen.
3. Mit dem gleichen Ansatz sind verschiedene Zeitdomänen beherrschbar:
   von [ms] (PC) über [µs] (Mikrocontroller) bis zu [ns] (FPGA).
4. Dank der gleichen Sprache für die gesamte Wertschöpfungskette (Machbarkeitsprüfung, Minimum Viable Products MVP's, Prototypen, Seriengerät, Test und Service) lässt sich der in diesen verschiedenen Phasen entwickelte Code wiederverwenden.
5. LabVIEW zeigt sich offen, lässt sich auf Hardware von Drittanbietern verwenden und bindet C/C++, Python, und M-Code ein.
6. Es verfügt über eine grosse, global aktive Community.

Nachteile

1. LabVIEW ist proprietär. Einige Unternehmen möchten möglicherweise kein Produkt verwenden, das aus einer einzigen Quelle stammt und nicht der OpenSource-Philosophie folgt.
2. Seine Einfachheit verführt zum «Spielen» und damit zu ad-hoc Programmen, die später organisch wachsen. Dabei sind gerade bei grossen Anwendungen auch bei grafischen Ansätzen seriöses Software Engineering und eine skalierbare Architektur Voraussetzung.
3. Die Betriebskosten - obwohl im Einklang mit ähnlichen Industrieprodukten - sollten vor der Einführung bedacht werden. LabVIEW ist kein Low-Cost-Produkt!
4. Grafischer Code kostet, vor allem im preissensiblen Embedded-Bereich! Beim Overhead und bei der Hardware. Low-Cost-Mikrocontroller liegen außerhalb der Reichweite. Mindestanforderung für eine zuverlässige Anwendung ist das SOM und diese Hardware hat ihren Preis.
5. Für diejenigen, die seit langem an die Textprogrammierung gewöhnt sind, kann die grafische Programmierung eine gewisse Umgewöhnungszeit erfordern.
6. Auf die Vorteile generativer AI, die heute bei textbasierter Programmierung genutzt wird, müssen die LabVIEW User noch etwas warten. NI R&D arbeitet daran: Codename «Nigel»

7 Die Power der NI Community – it's ok to have fun!

Die wahre Kraft von NI und LabVIEW liegt in seiner weltweit aktiven Community. Sie tauscht sich in unzähligen Foren aus und pflegt lokale User Groups. Vor allem hier ist die emotionale Nähe dieser Programmierer zu LabVIEW spürbar. Neben den vielen Vorteilen hadern wir im Alltag natürlich oft auch mit den Nachteilen. Im Grossen und Ganzen ist sich die Community jedoch einig, dass LabVIEW unsere Denkweise unterstützt, uns stark, produktiv und erfolgreich macht und dass grafisches Programmieren ganz einfach Spass macht. Dazu kommt mir der frühere NI-Slogan in den Sinn: «It's ok to have fun!»

Bild 4 | Die im Frühling 2024 gegründete Swiss LabVIEW User Group.

Die Swiss LabVIEW User Group ist eine solche lokale Gruppe. Sie wurde anfangs 2024 von Daniel Roth (PI Electronics AG) und Marco Schmid (Schmid Elektronik AG) ins Leben gerufen. Im Frühling 2024 war die Geburtsstunde. Aus einer einfachen Schweizer User-Group-Idee wurde ein erfolgreicher Event mit über 100 Teilnehmern, 20 Referenten, 6 Tracks, 20 Vorträgen, 7 LabVIEW Trainings und 2 hands-on Workshops zu Team & Führung. Das entspannte und gleichzeitig elektrisierende Startup-Feeling war magisch... Für die Schweizer LabVIEW-Community war dieser Event eine wichtige und positive Zusammenkunft und füllte ein mehrjähriges Vacuum. Die drei wichtigsten Erkenntnisse:

1. NI bleibt in der neuen Konstellation unter EMERSON strategisch selbständig und R&D treibt LabVIEW wieder aktiv voran.
2. Der Community Spirit erinnert an die frühen Zeiten der NIDays, als die Vorträge ähnlich «nerdy» waren.
3. Wir als LabVIEW User sind in der Community bereit, unser Wissen durch Teilen zu vermehren.

Werde gerne Mitglied auf den folgenden zwei lokalen User Groups auf der NI-Website: Swiss LabVIEW User Group und Swiss LabVIEW Embedded User Group!

8 Die Bedeutung der Übernahme von NI durch EMERSON

Am 23.Januar 2024 schickte NI ein klares Signal an die weltweite Community. Darin ist dieser Satz zentral: «*LabVIEW is core technology for NI and will remain central to the success for our users for decades to come.*» Auch die erfreuliche Nachricht, dass der End-of-Life-Status etwa beim NI System-on-Module (SOM, sbRIO9651) aufgehoben wurde, zeigt, dass einige Entscheidungen rückgängig gemacht wurden und positive Änderungen zu erwarten sind. Was die NI-Community seit Jahren vermisst hat, ist nun wieder da: eine zukunftsfähige Strategie für NI und klare Ziele für LabVIEW! Diese Message wurde an der inspirierenden Keynote der NI-Connect 2024 und auch im Interview mit Ritu Favre NI President: „We are the LabVIEW Company" bestätigt.

9 Die Zukunft von LabVIEW mit Generative AI

In der Welt moderner Software-Entwicklung war LabVIEW lange Zeit Vorreiter und bescherte seinen Usern Jahr für Jahr entscheidende Wettbewerbsvorteile, die der Zeit voraus waren. Zusammen mit NI's konsequenter Hardwarestrategie führte das zur hocheffizienten Entwicklungs-plattform, wie wir sie heute kennen und schätzen und sie ermöglichte den einfachen Einstieg in eine sonst komplizierte Technologie. Das macht den oft einzelnen LabVIEW-User in seiner Firma noch heute zum erfolgreichen «NI Fuchs», der zügig selbst komplexe Aufgaben lösen kann.

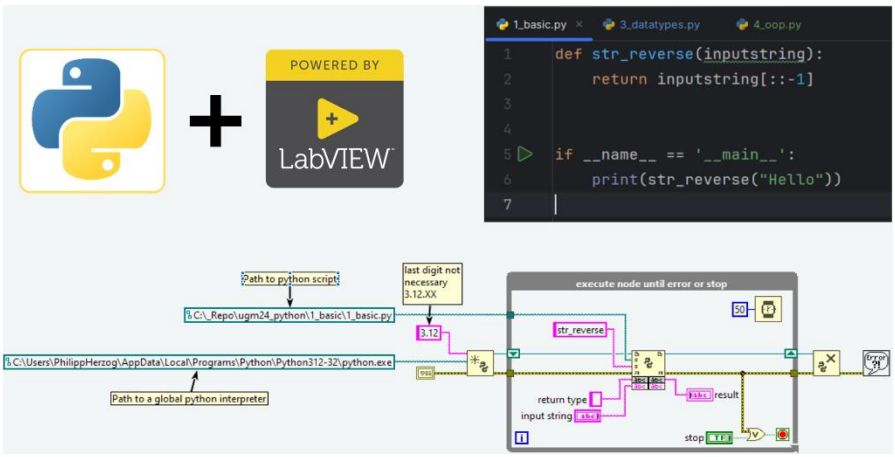


Bild 5 | Warum sich zwischen LabVIEW und Python zu entscheiden,
wenn sich beide Sprachen elegant kombinieren lassen? (Bild: Herzog Engineering)

NI hat den akademischen Markt über die letzten Jahre vernachlässigt und verlor damit seinen einstigen defacto Vorrang an Python. Mit Python kam eine neue, ebenfalls einfach zu erlernende und sehr leistungsfähige Programmiersprache ins Spiel und nimmt die Rolle des Platzhirsches ein. Und sie ist OpenSource! Kombiniert mit dem jüngsten Trend von Programmierassistenten wie Copilot wird das wohl zu einem Quantensprung führen und das könnte den bisher erfolgreichen Paradigmen wie der grafischen Programmierung den Rang ablaufen.

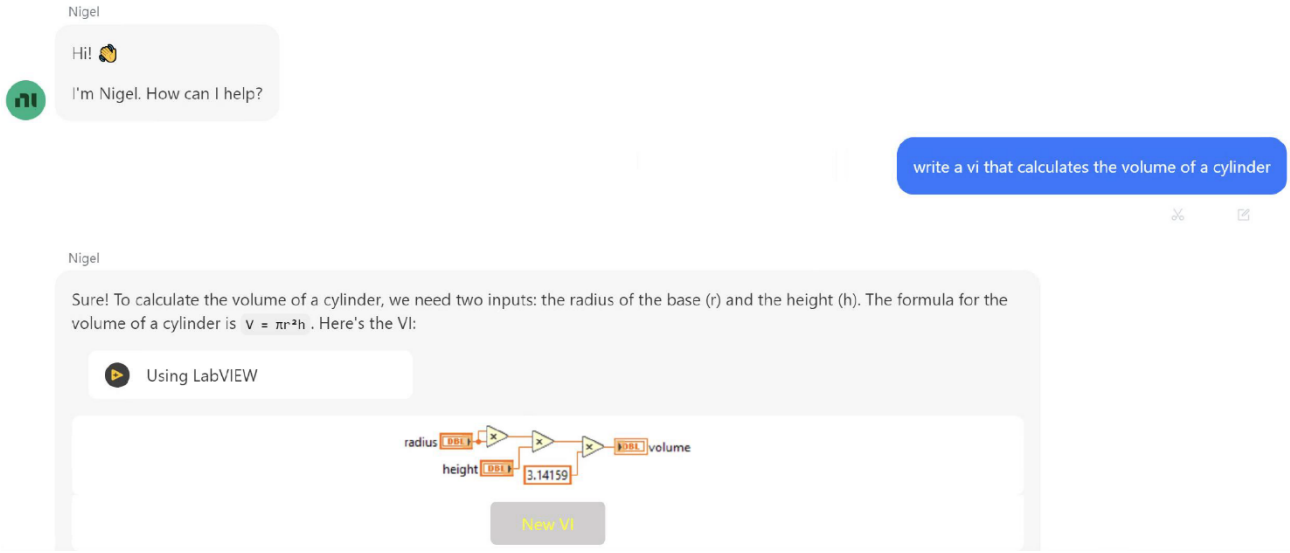


Bild 6 | Die NI-Version eines AI-Assistenten (Bild: NI)

Was aber, wenn sich die grafische Stärke und der Charme von LabVIEW mit der Effizienz von KI-basiertem Codieren verbinden liesse? Die Antwort von NI R&D: «Nigel»! Dieser zukünftige, smarte Assistent wurde an der NI Connect 2023 erstmals vorgestellt. Nigel selber ist NI's Maskottchen: der Adler im ehemaligen blauen National Instruments Logo. Am Swiss LabVIEW User Group Event wurde vorgestellt, wie sich dieser intelligente Assistent im Programmieralltag anfühlt. Nigel kann etwa:

- VI's schreiben! Typischer Case: ich brauche ein Tiefpassfilter 5.Ordnung mit einer Eckfrequenz von 5kHz. Als Ergebnis erhalte ich ein Code-Schnipsel und füge es gewohnt per Drag & Drop in meine Anwendung ein.
- VI's diffen
- VI Beschreibungen schreiben und Icons generieren
- Quick-Drop beschleunigen
- Gerätespezifikationen lesen und Tests für sie schreiben
- VI Unit Tests aufsetzen

Fazit: Nigel wird Programmierern in der domänenspezifischen LabVIEW-Entwicklungsumgebung das bieten, was gängige Sprachmodelle heute bei textbasierten Sprachen liefern.

10 Glossar

Kapitel 1	LabVIEW	Steht für grafisches Programmieren mit dem Datenflusskonzept
	Sprache «G»	Auf Datenfluss basierender, grafischer Code
	Graph	Mathematisches Netz aus Knoten und Kanten
Kapitel 2	VI	VI = Virtuelles Instrument = Funktionsblock in LabVIEW
	Oszilloskop	Elektronisches Messgerät
	Schaltplan	Grafische Darstellung einer elektronischen Schaltung
	Datenfluss	Nicht die Reihenfolge bestimmt den Programmfluss, sondern die Daten
Kapitel 3	Maschinencode	Ein Computerprogramm in einer Form, die der Mikroprozessor versteht
	Zielsystem	Der Computer, auf dem Maschinencode ausgeführt wird
	Programmier-prinzipien	Die Grundlagen einer typischen Programmiersprache wie Konstanten, Variablen, Verzweigungen, Schleifen, Rekursion, etc
	Betriebssystem	Eine Sammlung von Programmen für den Betrieb eines Mikrocontrollers
	Modellbasiert	Aus abstrakten Modellen eine lauffähige Software erzeugen
	Low-Code	Visuelle, einfache Programmierung
	No-Code	Programmierung über Konfiguration grafischer Blöcke
Kapitel 4	Petrinetz	Grafische Darstellung von Prozesszuständen mit Knoten und Kanten
	Graphenmodell	Dem Graphen (Knoten & Kanten) zugrunde liegendes math. Modell
	Datenzentrierung	Die Organisation und Verwaltung drehen sich um Daten
	Echtzeit	Vorgegebene Zeit, die bestimmte Prozesse verbrauchen dürfen
	Machbarkeits-Prüfung	Entscheidungsgrundlagen, ob ein Projekt durchgeführt werden kann
	Minimum Viable Products	Haben nur die allernötigsten Funktionen, eine Idee wird getestet. Die Kurzform heisst: MVP
	Prototyp	Greifbare Darstellung eines Designkonzepts, nahe am Serienprodukt
	Serienprodukt	Kann durch Produktion in einer Serie vervielfältigt werden
	Testing	Die Funktion einer Hard- und/oder Software wird überprüft
Kapitel 5	Neuronen	Eine Nervenzelle als Grundeinheit unseres Nervensystems
	Synapsen	Eine Verbindung zwischen zwei Nervenzellen
	Knoten	Hier: ein Verbindungspunkt in der Informatik
	Kanten	Eine Verbindung zwischen zwei Knoten

Kapitel 6	Frameworks	Eine Grundstruktur für Software
	Mikrocontroller	Hier: kleine, leistungsfähige Computer für Embedded-Systeme
	FPGA	Rekonfigurierbare Logik: Field Programmable Gate Array
	C/C++	C ist die gängige Sprache für Mikrocontroller. C++ ist objektorientiert.
	Python	Universelle, höhere Programmiersprache, läuft häufig interpretiert ab.
	M-Code	In M erzeugter Code, optimiert für mathematische Probleme
	Community	Eine Gemeinschaft von Menschen, die das gleiche Ziel verfolgen
	Proprietär	Software, die herstellergebunden ist wie in diesem Fall NI.
	Open Source	Der Öffentlichkeit zugänglicher Code, der in Communities geteilt wird
	Software Engineering	Prozesse und Technik zur Herstellung und Entwicklung von Software
	Software Architektur	Modularisierung, Frameworks, Objektorientierung, Design Patterns
	Embedded	Kombination von Hardware und Software für Eingebettete Systeme
	SOM	Hier: das System on Module (sbRIO9651) von NI
Kapitel 7	Lokale LabVIEW User Groups	Eine lokale Gruppe von LabVIEW Usern, die sich zu einem bestimmten Thema austauscht.
	Swiss LabVIEW User Group	Die von PI Electronics und Schmid Elektronik gegründete User Group, deren Mitglieder hauptsächlich aus dem DACH Raum stammen.
	Swiss LabVIEW Embedded User Group	Die von Schmid Elektronik gegründete User Group fokussiert sich auf grafisch programmierbare Embedded-Lösungen rund um das NI SOM
Kapitel 8	NI Connect	Der NI Event in Austin, welcher die frühere NIWeek abgelöst hat
Kapitel 9	Generative AI	Modelle der künstlichen Intelligenz (KI, AI), die digitale Inhalte erzeugen
	Copilot	Ein AI-unterstützendes Tool, welches in MS-Office angewendet wird
	Nigel	Der AI-Assistant, der zukünftig innerhalb der LabVIEW Entwicklungs-umgebung genutzt werden kann. Er befindet sich noch in Entwicklung.
	Diffen	Die Unterschiede zwischen Codedateien einander gegenüberstellen
	Quick-Drop	Zügiges einfügen von VI aus der Palette
	Unit Tests	Deckt Fehler innerhalb einer Software auf
	Domänenspezifisch	Hier im Zusammenhang mit domänenspezifischen Programmier-sprachen. Sie sind im Gegensatz zu allgemeinen Sprachen wie C/C++ zur Lösung von Problemen in einem ganz spezifischen Bereich ausgelegt. In LabVIEW ist zum Beispiel das Erfassen, Verarbeiten, Abspeichern und kommunizieren von Signalen sehr einfach und auf hohem Abstraktionsniveau möglich.

Münchwilen, Schweiz, Juni 2024, Marco Schmid, marco.schmid@schmid-elektronik.ch