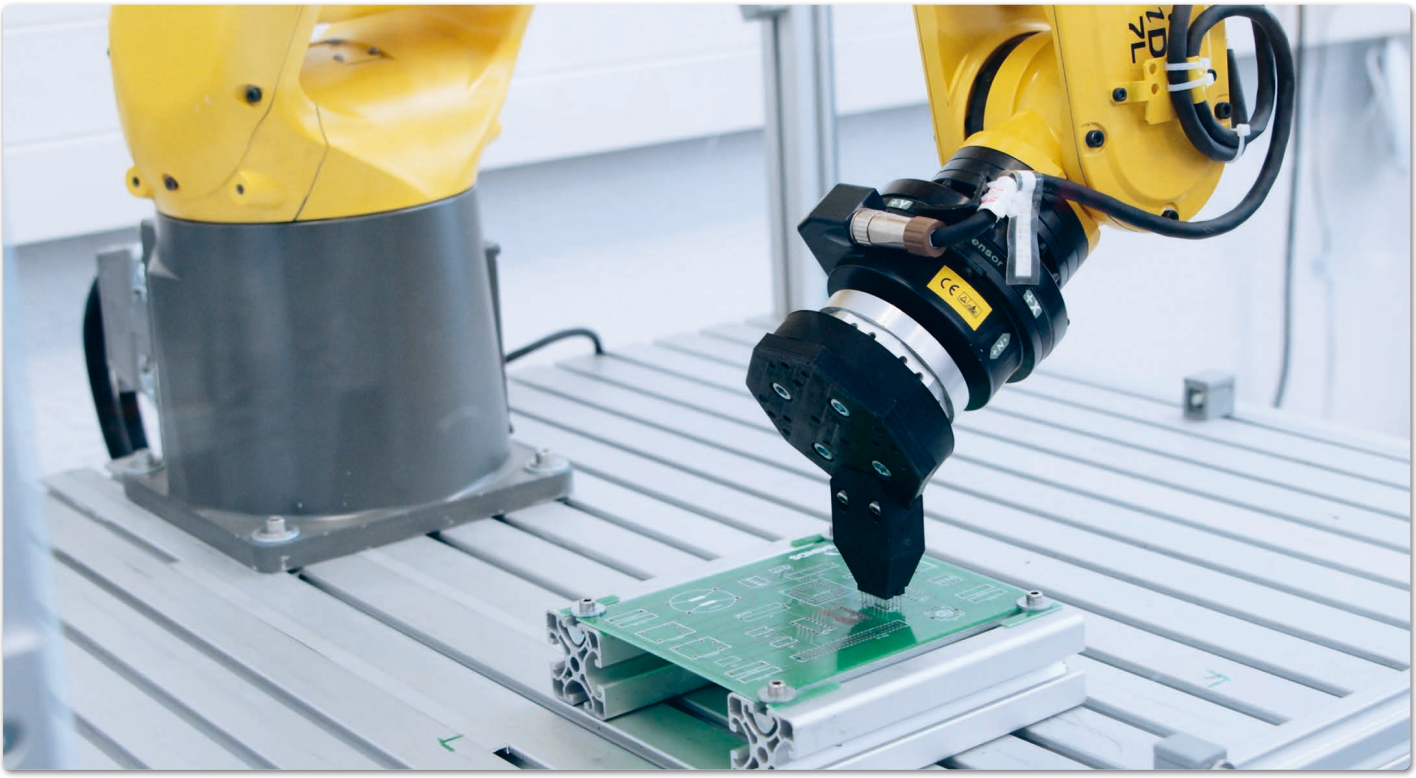
Roboter ansteuern:

Nativ oder durch externen Controller?



Roboter im Einsatz

In der Robotik gibt es zwei grundsätzlich unterschiedliche Herangehensweisen, um einen Industrieroboter für seine individuelle Aufgabe zu programmieren: Entweder über seine native, herstellergegebene Programmiersprache oder durch den Einsatz eines externen proprietären Controllers, der den Roboter dann feingranular ansteuert. Diese Controller werden sehr häufig mit Hilfe des Robot Operation Systems (ROS) implementiert. Beide Methoden haben individuelle Vor- aber natürlich auch Nachteile.

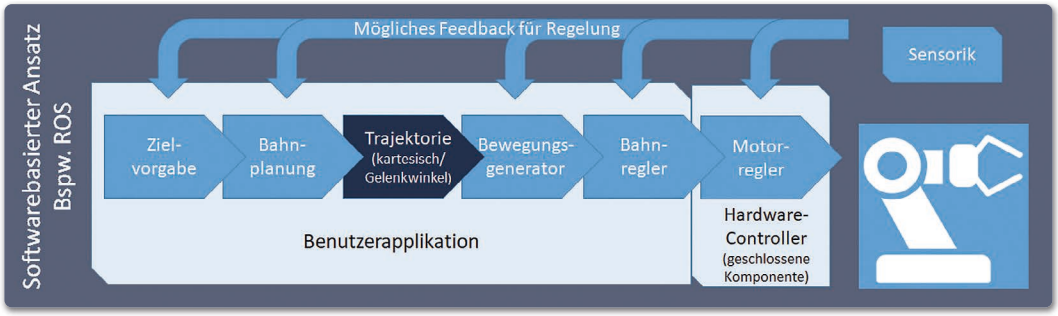
Im industriellen Umfeld wird bisher vorrangig mit den Hersteller-Sprachen gearbeitet, um ein System „aus einem Guss“ zu erreichen. In der Servicerobotik und Forschung hingegen kommen eigene, frei definierbare Controller zum Einsatz, so dass der Roboter hauptsächlich als Aktuator mit hardware-naher Regelung, aber ohne eigene Intelligenz, zu betrachten ist. Aktuell gibt es jedoch mehrere Robotik-Anbieter, die den zweiten Ansatz in der Industrie etablieren möchten.

Daher ist es wichtig, die Unterschiede einmal näher zu betrachten:

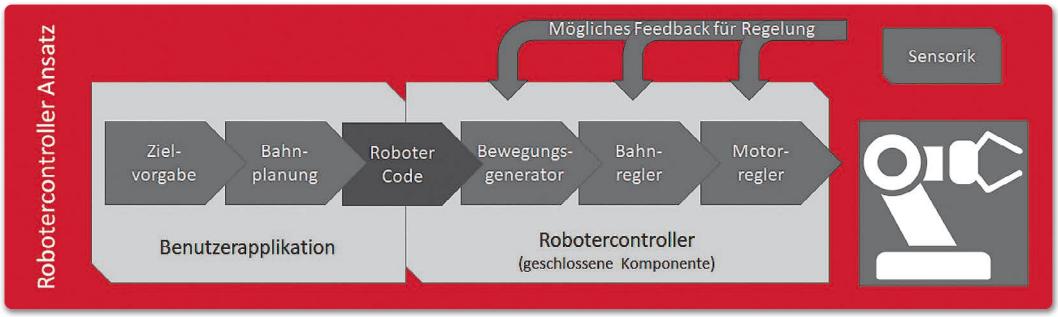
Die Anforderungen

In der Automatisierungstechnik herrscht durch die gute Planbarkeit des Robotereinsatzes oftmals ein Top-Down-Ansatz in der Programmierung und Ausführung. Dieser verlangt nach wenig Autonomie und damit auch geringeren Freiheitsgraden für den Roboter, da dessen Aufgaben klar definiert sind: Der Roboter kann im Allgemeinen jeden Arbeitsschritt Planen, die Bewegungsbahnen berechnen und ohne Unterbrechung bzw. Anpassung ausführen. Beispielsweise kann eine

Autor:
Andreas Hermann,
Senior Team Leader
Advanced Robotics
ArtiMinds Robotics GmbH
www.artiminds.com/de



Softwarebasierter Ansatz mittels individuellem Controller



Robotercontroller-basierter Ansatz mittels nativer Programmierung

kamerabasierte Objekterkennung zu einem diskreten Zeitpunkt eine Objektposition an das Roboterprogramm übermitteln, dieses berechnet eine Bewegung zum Objekt und überlagert nur deren letzten Abschnitt mit einer Kraftüberwachung.

Die Servicerobotik verlangt hingegen fast immer nach einem regelungsbasierten Ansatz der Bewegungen, da die Roboter sich meist in einer wenig strukturierten Umgebung bewegen und somit wesentlich höhere Unsicherheit herrscht. Daher kommt in der Wissenschaft oftmals ein Schichtenmodell mit bidirektionalem vertikalen Informationsfluss zum Einsatz, das dem Roboter wesentlich mehr Autonomie und Anpassungsfähigkeit auf allen Ebenen bietet. Beispiele sind hier eine verhaltensbasierte Entscheidungsfindung oder eine Echtzeit-Regelung für das Servoing (verfolgen eines dynamischen Zieles).

Die Vorteile

Ein individueller Controller unterliegt nur wenigen Randbedingungen und erlaubt somit eine freie Systemarchitektur. Somit kann die Hard- und Software (inklusive des Betriebssystems und der Programmiersprach) beliebig gewählt werden, um deren Vorteile zu nutzen oder vorhandene Vorarbeiten ohne Portierung einzubringen. Dies erlaubt insbesondere die Verwendung beliebiger Sensoren, die mit dem Roboter von Haus aus nicht kompatibel wären. Einschränkungen der Regelung ergeben sich einzig aus der Dynamik der Aktoren und der Schnittstelle des Roboters. Somit sind auch mehrfach überlagerte Regelungsansätze möglich (bspw. Nullraum-Regelung oder die Berücksichtigung harter und weicher Randbedingungen während der Ausführung).

Vorteile der nativen Programmierung

Für die Industrie überwiegen jedoch meist die Vorteile, die eine native Programmierung mitbringt: Eine umfassende Garantie und Supportmöglichkeiten durch den Roboterhersteller. Dies ist möglich, da die Ausführung auf dem originalen Robotercontroller stattfindet. Die Programmierung erfolgt mit herstellereigenen Tools und Sprachen. Zusatzhardware ist zertifiziert oder nur über getestete Protokolle möglich. Somit kann jederzeit ein deterministisches Verhalten sichergestellt werden, auch was die Dynamik des Roboters und somit die Taktzeit betrifft. Hard- und Softwarekomponenten sind optimal aufeinander abgestimmt, um bestmögliche Performance zu erreichen und gleichzeitig sind sichere Limits klar vorgegeben. Weiterhin ist durch das klar definierte Set an erprobten Befehlen zur Steuerung / Regelung eine vollständige Dokumentation und die Möglichkeit für Support bei Problemfällen gegeben.

Einschränkungen und Risiken

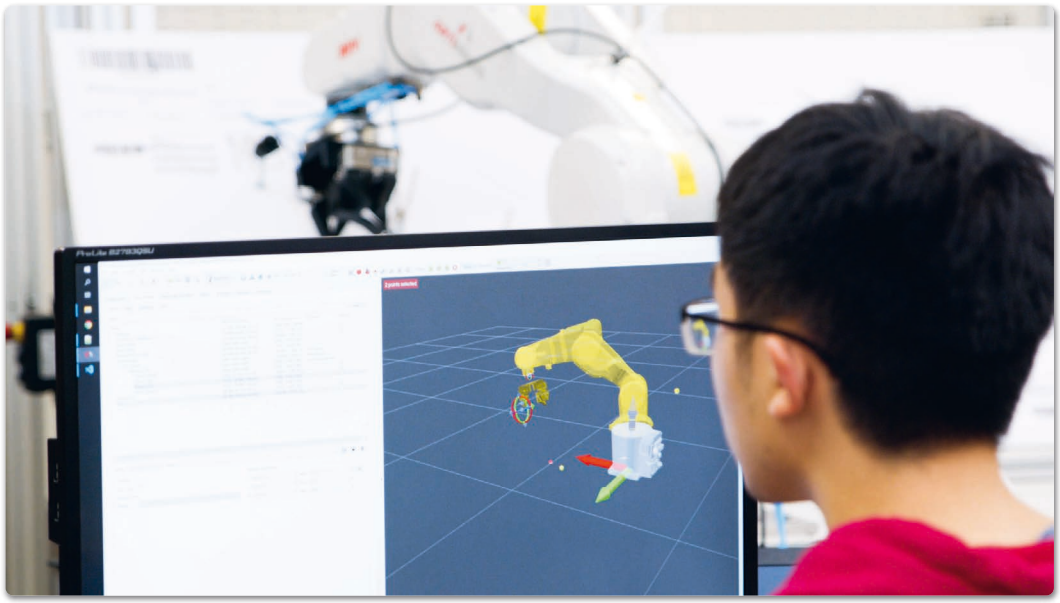
Die Nutzung einer durch den Roboterhersteller vorgegebenen Umgebung erfordert Fachkenntnisse in der Programmierung und das Auseinandersetzen mit dessen eventuellen Unzulänglichkeiten und Einschränkungen. Das Gesamtsystem kann nur das, was der Roboterhersteller vorsieht und zulässt. Dies bedeutet meist auch, dass der Roboter zur Laufzeit nur das kann, was zur Programmierzeit angedacht wurde.

Andererseits bietet ein individueller Controller sehr großes Potential für Fehlfunktionen, da ein vielschichtiges System mit mehreren Komponenten involviert ist, bei denen eine Kompatibilität in eigener Verantwortung liegt. Somit ist hier ein sehr tiefes Verständnis für die Hardware und ihre Dynamik notwendig. Liegt diese nicht vor, besteht die Möglichkeit, die Hardware durch fehlerhafte Ansteuerung zu überlasten, wenn dieser zu viel Dynamik abverlangt wird. Dadurch gestaltet es sich im Allgemeinen auch wesentlich schwieriger, sich die Sicherheit eines solchen System zertifizieren zu lassen.

Fazit

Natürlich erlauben viele Industrieroboter auch einen Mischbetrieb, durch den es möglich wird, eigene Sensorik oder externe Regler für spezielle Teilszenarien anzubinden. So ist es häufig möglich, Korrektur-Offsets auf einzelne, sonst statische Trajektorien aufzubringen (Überlagerung) und diese auch noch auf sicherheitsrelevante Maximalwerte zu überwachen. Auf diese Weise können die Vorteile beider Herangehensweisen genutzt und die Nachteile lediglich für kleine Programmabschnitte in Kauf genommen werden.

Zudem finden sich auch spezielle herstellerunabhängige Softwarelösungen am Markt, die zwar nativen Code generieren, aber auf einer Template-basierten Programmierung beruhen. D.h. programmiert wird nicht mehr klassisch per Zeilencode, sondern per Drag-and-Drop von vordefinierten Funktionsbausteine. Dies minimiert einerseits den Programmieraufwand und ermöglicht das Programmieren von sensoradaptiven Roboteranwendungen auch ohne dediziertes Expertenwissen. ◄



Programmierung eines Roboters