1 Geräteplattform ›QIAsymphony SP‹

# Symphonie aus **C++ und Lua**

**Skriptsprache mit kleinem Interpreter.** Um große Softwareprogramme überschaubar und leicht änderbar zu halten, bedienen sich Programmierer einer Skriptsprache. Kleine, in sich abgeschlossene Module, auch Skripte genannt, sollen sich leicht schreiben und mit anderen Programmiersprachen verbinden lassen. Lua, bisher besonders beliebt bei Programmierern von Computerspielen, hat sich jetzt auch in einem komplexen Mechatronikprojekt für die Diagnostik bewährt. [ME101758]

CLAUS KÜHNEL
DANIEL ZWIRNER

■ Die molekulare Diagnostik beruht auf dem Nachweis von Nukleinsäuren (DNA oder RNA) zum Beispiel aus Blut- und Gewebeproben. Die Nukleinsäuren in den Proben müssen nach der Entnahme stabilisiert und so extrahiert und aufgereinigt werden, dass die nachfolgende Diagnostik nicht nur zuverlässig, sondern auch reproduzierbar ist. Der von Qiagen entwickelte ›QIAsymphony SP‹ (Bild 1) zur Probenaufreinigung spart Zeit und Kosten bei der Probenhandhabung, was zu einer hohen Leistungssteigerung führt.

## Die Hardwarestruktur

Der QIAsymphony SP ist mit zahlreichen Aktoren für das xyz-Portal mit Pipettierfunktion, Sensoren zur Prozessüberwachung und einem integrierten Steuerungsrechner mit grafischer Benutzerschnittstelle (GUI) ausgestattet. Es handelt sich um ein komplexes, mechatronisches System, dessen Software in der Sprache C++ geschrieben ist.

Bild 2 zeigt die vereinfachte Systemarchitektur. Die CPU und der Touchscreen mit grafischer Benutzerschnittstelle bilden den zentralen Rechnerkern. Über CAN/CANopen sind mehrere lokale Controller miteinander verbunden, die diverse DC-Motoren, Schrittmotoren oder Elektromagnete steuern und Sensoren einschließlich 2D-Barcode-Kameras abfragen, um der CPU Informationen zur Verfügung zu stellen beziehungsweise deren Kommandos in den lokalen Controllern umzusetzen. Über Ethernet-TCP/IP ist das Gerät mit einem Host verbunden und lässt sich so in ein Netzwerk integrieren.

### KONTAKT

QIAGEN Instruments AG,
CH-8634 Hombrechtikon,
Tel. +41 55 /2 54 -21 25,
Fax +41 55 /2 54 -23 00,
www.qiagen.com

## Die Softwarestruktur

Das System basiert auf dem Betriebssystem ›Embedded Linux‹. Im Schichtenmodell (Bild 3) ist die Softwarearchitektur schematisch dargestellt. Der abzuarbeitende (biologische) Prozess liegt als XML-Datei vor und wird in der Schicht ›Biologische Funktionen‹ abgearbeitet. In dieser Schicht befinden sich weitere Komponenten, die das störungsfreie Abarbeiten der Proben garantieren (Inventory Manager, Batch Manager). Über die GUI auf dem Touchscreen kommuniziert der Benutzer mit dem Gerät. Die Proben und die eingesetzte Labware (Tips, Plates, Cartridges) identifiziert das System optisch, beispielsweise anhand von 1D- oder 2D-Barcodes.

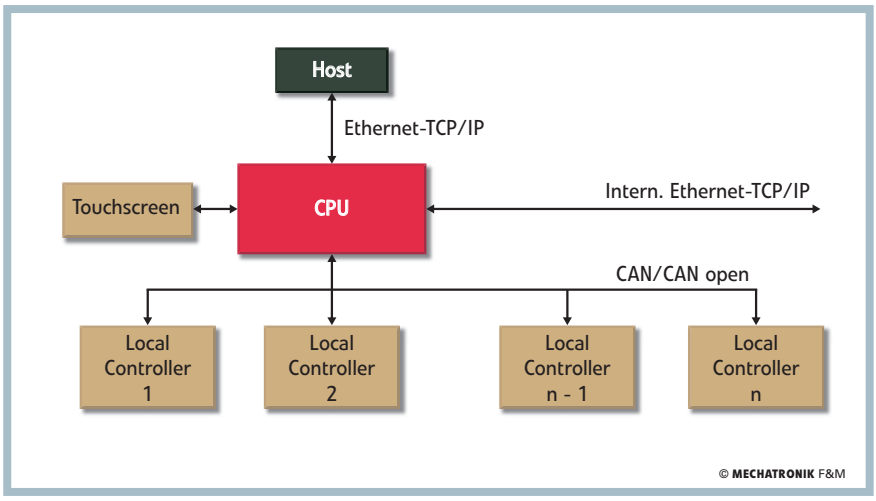
Der Load Check stellt sicher, dass das Gerät ausreichend mit Labware für die abzuarbeitenden Proben ausgerüstet ist, und stellt diese Daten dem Inventory Manager in der Datenbank bereit.

Zwischen den biologischen Funktionen und der eigentlichen Hardwareabstraktion liegt noch die Zwischenschicht ›Logi-

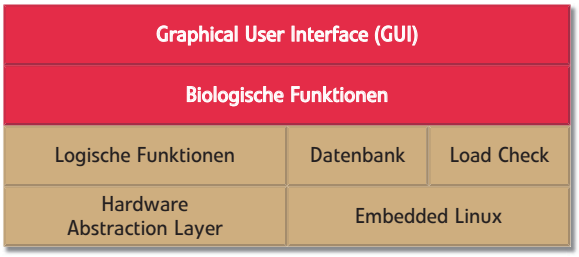
sche Funktionen‹, die eine weitere Kapselung der Gerätefunktionalität vornimmt. Bei der Entwicklung von Hard- und Software waren sechs Disziplinen involviert:

- Biologen für die Entwicklung der zu automatisierenden Prozesse und die Prozessintegration,
- Kunststoffingenieure für die Entwicklung der Labware, also Kunststoffspritzgussteile für den Einmalgebrauch (Tips),
- Maschinenbauingenieure für die Entwicklung der Gerätemechanik,
- Elektronikingenieure für die Entwicklung der Steuerungselektronik inklusive Sensorik und Antriebe,
- Softwareingenieure für die Entwicklung der hardwarenahen Schichten,
- Softwareingenieure und Informatiker für die Entwicklung der prozessnahen Schichten (Applikation) und der GUI.

An einem Standort wurde die Mechanik, Elektronik und hardwarenahe Software entwickelt. An einem zweiten Standort fand die Applikationsentwicklung, die GUI-Entwicklung und die Hostanbindung statt. Die Basisfunktionalität wurde in Funktionsmustern und Prototypen implementiert, bevor die Applikation und die GUI für die Inbetriebnahme und Tests zur Verfügung standen. Das erforderte ein Tool, mit dem sich die Basisfunktionalität testen lässt.




**2 Vereinfachte Systemarchitektur**




**3 Vereinfachte Softwarearchitektur**

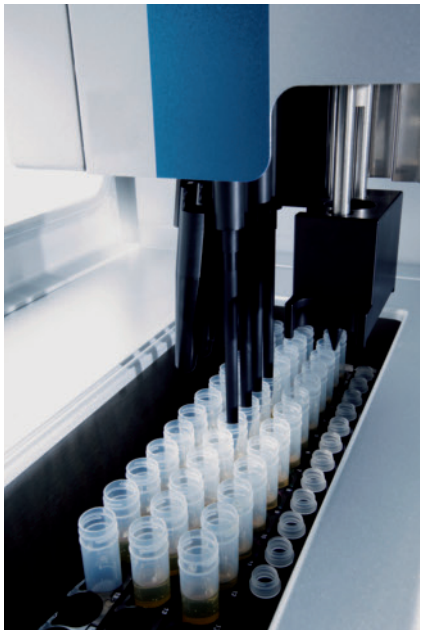
## C und Lua bilden ein Team

Lua (portugiesisch Mond) ist eine Skriptsprache, die in andere Programme eingebunden werden kann, um diese leichter weiterentwickeln und warten zu können. Eine großer Vorteil von Lua ist die kleine Größe des kompilierten Skript-Interpreters [1, 2]. Lua wurde 1993 von der Computer Graphics Technology Group in Rio de Janeiro, Brasilien, entwickelt und ist eine freie Software unter MIT-Lizenz. Lua-Programme werden vor der Ausführung in Bytecode übersetzt [3].

Der Lua-Interpreter kann über eine C-Bibliothek angesprochen werden, die auch eine API für die Laufzeitumgebung des Interpreters für Aufrufe vom C-Programm aus beinhaltet. Mittels API lassen sich Programmteile in C und Lua schreiben, während Variablen und Funktionen in beiden Richtungen erreichbar sind. Eine Funktion in Lua kann eine Funktion in C aufrufen und umgekehrt. Lua ist in ANSI-C implementiert und unterstützt sowohl die funktionale als auch die objektorientierte Programmierung.

Am Anfang des Projekts war klar, dass der Roboter neben der eigentlichen Applikation einen einfachen Skript-Interpreter haben soll, mit dem beliebige Aktionen ausgeführt werden können. Um den Entwicklungsaufwand für einen spezifischen Interpreter zu vermeiden, wurde nach verfügbaren Lösungen gesucht. Die in [4] beschriebenen Erfahrungen führten zu Lua. Ein Nachmittag reichte, um Lua zu integrieren und das erste Kommando für die Roboteransteuerung zu implementieren.

**4 Probenrack im Sampledrawer**



**5 Pipettierkanal mit 200-µl-Tip**

## Kalibrieren und Testen

Nach weiteren Erfahrungen hat Qiagen ganz auf Lua gesetzt. Dem Test der Low-Level-Steuerung der Maschine folgten Lua-Anbindungen für weitere Komponenten. Heute stehen für alle Schichten unterhalb der GUI spezifische Lua-Packages bereit. Danach verifizierte das Testcenter einzelne Module mit Lua. Es entstanden erste Lua-Skripte, um den Roboter zu kalibrieren. In der Qualitätskontrolle werden heute mithilfe von Lua-Skripten fertige Roboter kalibriert und getestet. ›

› Um die Kommandozeilen-Schnittstelle zu verbergen, folgte die Entwicklung einer einfachen grafischen Bibliothek, basierend auf VT102-Kommandos und einer Touchscreen-Anbindung. Daraus entstanden mehrere Applikationen, die in der Produktion als Fertigungshilfsmittel dienen oder bei unseren Lieferanten als Test- und Programmiersoftware eingesetzt werden. Bei Fertigungshilfsmitteln werden meist nur beschränkte Dialoge geführt, für die in der Regel ein kleiner, interner Touchscreen ausreicht.

Der Zusammenfassung in [4] unter der Überschrift ›Lua ist nicht von dieser Welt‹ lässt folgende Einschätzung zu: Lua ist einfach anzuwenden, aber die einfache Syntax verhüllt die Mächtigkeit, denn Lua unterstützt Objekte (ähnlich wie Perl), Metatables machen den Table Type erweiterbar und die C-API erlaubt die perfekte Integration und Erweiterungen zwischen Skripten und der Hostsprache. Lua wurde in Programmumgebungen, die in C, C++, C#, Java oder Python geschrieben sind, integriert. Bevor man eine Konfigurationsdatei oder ein Ressource-Format und einen Parser dafür erzeugt, sollte man es mit Lua versuchen.

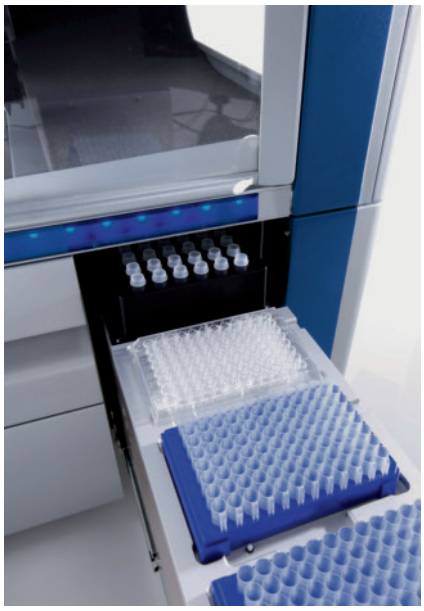
## Leicht integrierbar

In die Applikation wurde Lua in zwei Schritten integriert: Integrieren des Lua-Interpreters und Erweitern des Interpreters um applikationsspezifische Befehle. Der Lua-Interpreter lässt sich anhand des Standalone Interpreters (src/lua.c), der im Lua-Package mitgeliefert wird, implementieren. Anschließend muss er mit der Applikation verlinkt werden. Die Ein- und Ausgabe funktioniert über ›`stdin`‹ und ›`stdout`‹. Unter www.mechatronik.info/programmlistings steht ein Beispielprogramm (Listing 1) zur Verfügung.

Im Interpreter kann jetzt die Funktion `xna.init()` mit keinem, einem oder zwei Parametern aufgerufen werden. Der Interpreter ist später erweitert worden, sodass die Ein- und Ausgabe auch via Netzwerk (Telnet) möglich ist, da der Applikation `stdin` und `stdout` im fertigen Produkt nicht mehr zur Verfügung stehen. Die Erweiterung des Interpreters war einfach, weil alle Komponenten, die an Lua angebunden werden sollten, als Singleton (Einzelstück) vorliegen. Für jede angebundene Komponente hat Qiagen eine eigene Bibliothek erstellt. Heute existieren unter anderem Bibliotheken für die Inventory-Funktionen (`inv`), für den Load Check (`lch`) zum Testen der Beladung, für Logging-Funktionen (`logger`) zum Erstellen entsprechender Log Files sowie für die Kommunikation mit den Hardwaremodulen des Geräts (`xna`).

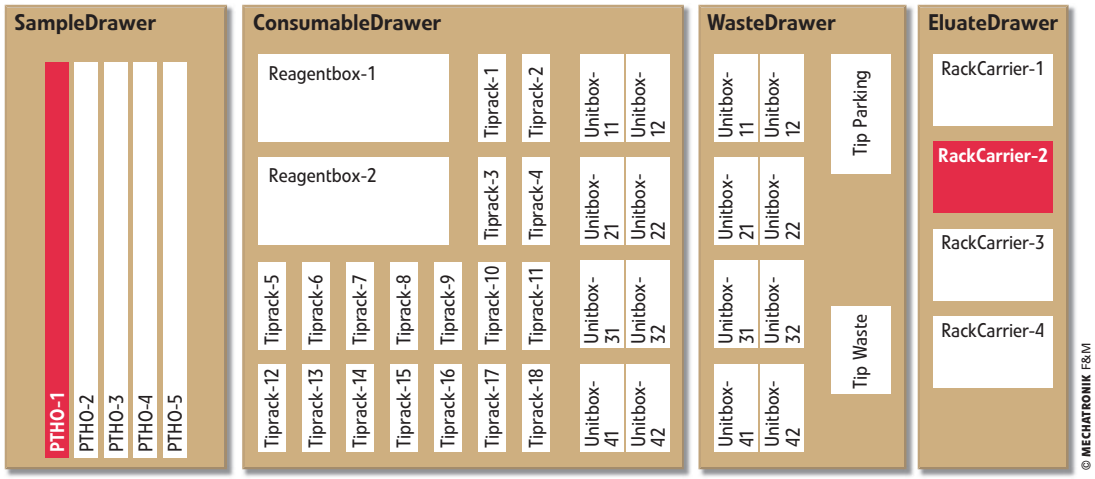
## Skripte bewähren sich in der Praxis

Die Verwendung von Lua zum Steuern von Grundfunktionen des Geräts soll beispielhaft am Flüssigkeitstransfer von der Probenausgabeschublade (SampleDrawer) zur Probenausgabeschublade (EluateDrawer) aufgezeigt werden. Bild 4 zeigt den SampleDrawer mit einem eingeschobenen Probenrack (rot markiert). Im Programmbeispiel wären die äußeren drei Stellplätze mit sogenannten Primary Tubes, die die zu transferierenden Flüssigkeiten beinhalten, bestückt. Der eigentliche Flüssigkeitstransfer erfolgt mit einem Pipettor durch Aspirieren (Aufnehmen) und Dispensieren (Abgeben) der Flüssigkeit, nachdem der Pipettierkopf über den Worktable bewegt worden ist. Bild 5 zeigt den Pipettierkopf beim Aufnehmen von Flüssigkeit aus Primary Tubes. Die Flüssigkeiten werden in unserem Beispiel direkt in eine im EluateDrawer platzierte Microtiter Plate (Bild 6) transferiert.



**6 Microtiter Plate im EluatDrawer**

Bild 7 zeigt den Worktable schematisch. Neben den bereits erwähnten Schubladen sind hier noch Schubladen für Verbrauchsmaterialien (ConsumableDrawer), wie Tips, Cartridges und Reagenzien, sowie für die zu entsorgenden Komponenten (WasteDrawer) zu sehen. Die Positionen, die für den zu beschreibenden Flüssigkeitstransfer infrage kommen, sind rot markiert.




**7 Schematische Darstellung des Worktables mit vier ›Schubladen‹**

Das Skript (siehe Listing 2; unter www.mechatronik.info/programmlistings) beginnt mit einer Initialisierung (`init()`) und einem anschließenden Verriegeln der Schubladen (`start()`). Durch das Verriegeln sind die Schubladen auch in xy-Richtung arretiert, eine Voraussetzung für das

| Name | Source | Params | | |
|---|---|---|---|---|
| Yellow | SampleDrawer$PTHO-0$0 | liquid1.params | | |
| | | liquidClass | aspParams | dispParams |
| Blue | SampleDrawer$PTHO-0$1 | liquid2.params | | |
| | | liquidClass | aspParams | dispParams |
| Oil | SampleDrawer$PTHO-0$2 | oil.params | | |
| | | liquidClass | aspParams | dispParams |

**A Beschreibung der Proben in der Tabelle `reagents = {}` im Listing 2**

präzise Positionieren. Der Flüssigkeitstransfer soll so vorgenommen werden, dass vom Pipettor ein Tip aufgenommen wird (`getTips()`). Mit diesem wird über eine kapazitive Messung der Füllstand der drei Proben (`YELLOW, BLUE, OIL`) gemessen (`lldLiquid()`). Diese Füllstände werden dem Inventory Manager übergeben. Anschließend erfolgt der Flüssigkeitstransfer (`transferLiquid()`) von den drei Eingabepositionen zu einer identischen Ausgabeposition, was nichts anderes bedeutet als ein Mischen der drei Flüssigkeiten. Nach dem letzten Flüssigkeitstransfer wird der Tip (in Waste) abgeworfen (`putTips()`), bevor am Ende des Skripts die Schubladen entriegelt werden (`stop()`).

Das Listing 2 beginnt mit Vereinbarungen. `useChannels = {1, 0, 0, 0}` beschreibt, dass hier nur der erste Pipettierkanal von insgesamt vier verwendet wird. Ein `tipType = „Tip200“` kennzeichnet die Tips mit einem Volumen von 200 µl. Der `welloffset = 0` beschreibt kein Weiterführen zur nächsten Position bei der Flüssigkeitsabgabe, das heißt, alle drei Flüssigkeiten werden in der gleichen Position abgegeben.

Nach den Konstantenvereinbarungen für die Flüssigkeiten in der Eingabezone werden die Zuordnungen zu den Rackpositionen in der Tabelle `reagents = {}` festgehalten. Tabelle A listet die Parametersätze für die Kennzeichnung des präzisen Pipettiervorgangs einer Flüssigkeit. Das Listing 3 im Internet dient als Beispiel für die Parametersätze der Flüssigkeit Oil.

Aufbauend auf diesen Funktionen lässt sich der beschriebene Flüssigkeitstransfer zusammenstellen. Der Abschnitt `Main` in Listing 2 zeigt die einzelnen Aufrufe. Soll beispielsweise nach jedem Pipettiervorgang der benutzte Tip abgeworfen und ein neuer aufgenommen werden, dann gestaltet sich dieser Vorgang wie in Listing 4 im Internet gezeigt (siehe unten).

Lua ist erfolgreich in das Qiagen-Projekt integriert worden. Bei ähnlich gelagerten Projekten würden die Autoren diesen Weg wieder beschreiten. ■

## FAZIT

### Einfache, aber mächtige Syntax

Die Softwareentwicklung für Geräte in der Laborautomatisierung ist komplex und unterliegt in vielen Fällen strengen regulatorischen Anforderungen. Wichtig für eine effektive Softwareentwicklung ist es deshalb, auf einer weitgehend stabilen Hardware und Firmware aufsetzen zu können. Die Scriptsprache Lua ermöglichte ein flexibles System für den Test der tieferen Schichten (Bild 3) der Software und der Hardwaremodule des Laborgeräts zu einem frühen Zeitpunkt. Die Syntax von Lua ist unkompliziert und verhüllt die Mächtigkeit dieser Scriptsprache. Alle gewonnenen Erkenntnisse des Projekts ›Probenaufreinigung‹ sind nahtlos in die Fertigungshilfsmittel der Produktion von Qiagen und bei Lieferanten eingeflossen.


## Autoren

Dr. CLAUS KÜHNEL ist Associate Director Electronic Engineering bei Qiagen Instruments und
DANIEL ZWIRNER ist als Software-Ingenieur für die hardwarenahe Software, Linux und die Einführung von Lua zuständig.


## Literatur


1 LUA Wikipedia http://de.wikipedia.org/wiki/Lua; LUA Homepage: www.lua.org; LUA User Website http://lua-users.org; LuaForge: http://luaforge.net/
2 Lerusalimschy, R.: ›Programmieren mit Lua‹; Open Source Press, 2006, ISBN 978-3937514222
3 Lerusalimschy, R.; De Figueiredo, L. H.; Celes, W.: ›Lua 5.1 Reference Manual‹; 2006; ISBN 978-8590379836
4 Streicher, M.: ›Embeddable scripting with Lua; Lua offers high-level abstraction without losing touch with the hardware‹; www.ibm.com/developerworks/linux/library/l-lua.html


## www.mechatronik.info

Alle genannten Programmlistings
1 Erweiterung des Lua-Interpreters
2 Quelltext der Datei ›test-mix-under-oil-200-1.lua‹
3 Quelltext der Datei ›oil.params‹
4 Flüssigkeitstransfer mit Tipwechsel
finden Sie im Internet unter:
www.mechatronik.info/programmlistings