Software-Updates im Produktlebenszyklus:

Always change a running system

(Bild: Shutterstock)

Die Elektronikentwicklung ist abgeschlossen, das Gerät im Einsatz. Jetzt gilt es sicherzustellen, dass es den Anforderungen eines langjährigen Lebenszyklus gerecht wird. Die Grundlagen dafür müssen bereits im Design gelegt werden – und dabei spielen Software und Update-Fähigkeit eine ebenso wichtige Rolle wie die Hardware selbst.

Von Axel Berghoff

Product Change Management ist ein anerkanntes Thema in Bezug auf Hardware. Die geschickte Auswahl der Bauteile beim Design ist entscheidend für Änderungsaufwand und Zukunftsfähigkeit der Elektronik. Doch neben der Hardware wächst die Bedeutung von Software und Update-Fähigkeit: Mit der zunehmenden Vernetzung sind immer mehr Systeme im Internet exponiert. Zudem steigt die Bedeutung der erhobenen Daten und damit die Gefahr vor unerlaubten Zugriffsversuchen. Gleichzeitig vereinfacht die hohe Standardisierung von Protokollen und Hardware den Zugang zu Systemen, sobald eine Sicherheitslücke ausgenutzt wurde. Die steigende Zahl der gefundenen Sicherheitsrisiken – Spectre und Meltdown haben es ja sogar in die Aufmerksamkeit der Massenmedien geschafft – zeigt eindrucksvoll, wie sich die Bedrohungslage in den letzten Jahren verändert hat.

„Never change a running system" gilt nicht mehr

Unter diesen Umständen müssen immer mehr elektronische Geräte über ihren gesamten Lebenszyklus hinweg ein Höchstmaß an Security bieten und ermöglichen, auf Sicherheitslücken und Bedrohungen zu reagieren. „Never change a running system" darf nicht mehr gelten, wenn Sicherheitslücken nicht verhindert werden können und geschlossen werden müssen. In dieser Situation müssen die Voraussetzungen geschaffen werden, entsprechende

Rahmenbedingungen für das Software-Lifecycle-Management

Voraussetzung für das Lifecycle-Management der Software ist das Vorliegen einer BSP-Spezifikation, die den gesamten Funktionsumfang der Plattform beschreibt. Es wird eine automatisierte Testumgebung verwendet, mit der die komplette Funktion des Systems entsprechend der BSP-Spezifikation geprüft werden kann. Die Tests umfassen in erster Linie die auf den Boards angelegten Schnittstellen, Treiber und Verbindungen. Kundenapplikationen werden in der Regel nicht in den Test aufgenommen. Die Standardtests umfassen „gängige" Schnittstellen. Besondere Schnittstellen oder spezielle Protokolle können durch Erweiterung der Prüfspezifikation individuell aufgenommen werden; evtl. ist dafür die Erstellung spezieller Test-Hardware erforderlich.

Für die Tests ist das auf Jenkins basierende System für die Continuous Integration mit der Testumgebung für automatische Hardware-Tests verknüpft. Damit eignet sich das Setup optimal zur kontinuierlichen Integration von Standard-Board-Support-Packages sowie von kundenspezifisch angepassten BSPs.

Patches schnell und ohne großen Aufwand zu implementieren. Phytec trägt dem Rechnung und setzt für seine Board Support Packages weitgehend auf die Vorteile von Mainline-Linux. Der Open-Source-Ansatz sorgt für stabilen, vielfach erprobten Code, für schnelle Bug- und Security-Fixes und die Pflege und Weiterentwicklung von Treibern durch die Community. Dabei spielt die Aktualität der verwendeten Software eine große Rolle: Häufig sind Patches zum Schließen von Sicherheitslücken nur für die aktuellsten Betriebssystemversionen verfügbar. Ältere Versionen folgen erst verspätet, wenn sie überhaupt noch mit Patches versorgt werden.

Vom Hersteller-BSP zur Mainline-Unterstützung

Im x86-Bereich ist es schon seit vielen Jahren üblich, dass neue Bausteine bereits bei Markteinführung mit Mainline-Linux-Implementierungen angeboten werden. Mainline bedeutet, dass der Linux-Kernel ohne Modifikationen/Patches verwendet werden kann. Bei ARM-basierten Controllern hingegen ist es üblich, dass ein neuer Controller zunächst mit einem Hersteller-BSP (Board Support Package) ausgestattet ist. Darunter wird eine Linux-Implementierung des Chipherstellers verstanden, die in der Praxis tausende Patches gegen Mainline enthält und oft auch mit proprietärem Code angereichert ist, etwa zur Ansteuerung von Einheiten zur Grafikbeschleunigung. Auch für ARM gibt es Mainline-Implementierungen, aber aufgrund der vielfältigen Peripherie-Konfigurationen dauert es meistens sehr lange, bis ein Controller mit seinem vollständigen Feature-Set unterstützt wird. Zur Pflege der Chip-spezifischen Patches ist häufig das Insiderwissen des Chiphersteller erforderlich. Trotz Open Source entsteht faktisch eine Abhängigkeit von der Pflege des Betriebssystems durch den Chiphersteller. Verschärfend kommt hinzu, dass der Chiphersteller sein Interesse an der Betriebssystempflege nach ein paar Jahren verlieren wird, spätestens dann, wenn die Neudesigns mit dem entsprechenden Controller deutlich zurückgehen. Aufgrund des hohen Pflegeaufwands der vielen Patches und weil dieser sehr weitgehend am Hersteller hängt, ist es um die Aktualität der Hersteller-BSPs im Allgemeinen nicht besonders gut bestellt.

Aber es gibt gute Nachrichten: Die Linux-Community schafft es, in relativ kurzer Zeit den Umfang der Feature-Unterstützung in Mainline für neue Controller deutlich zu verbessern. Alternative Ansätze für proprietäre Codeanteile (z. B. ETNAVIV für die Grafikbeschleunigung) finden weite Unterstützung und damit eine hohe Verbreitung. Als zweite gute Nachricht kann die zunehmende Akzeptanz (oder besser gesagt die zunehmende Einsicht) der Halbleiterhersteller für Mainline genannt werden. Es ist damit zu rechnen, dass die Halbleiterhersteller ihre Aktivitäten rund um Mainline deutlich ausbauen. Dies ist die logische Konsequenz des geschilderten Szenarios.

Die Frage, ob ein Unternehmen, das ein neues Produkt entwickelt, gleich auf Mainline setzt oder mit dem Hersteller-BSP startet, hängt schlichtweg von der Unterstützung der Features ab, die in der Anwendung benötigt werden. Es kann aktuell durchaus zwischen einem und drei Jahren dauern, bis die Mainline-Unterstützung wirklich einen Controller in allen Facetten optimal unterstützt. In dieser Zeit wird das Hersteller-BSP typischerweise sehr gut gepflegt (wenn auch, wie ausgeführt, ggf. in älteren Versionen). Praktisch gesehen führt für ein nachhaltiges Patch-

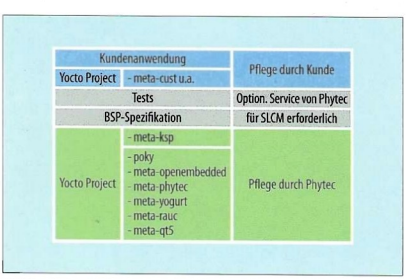

Bild 1. Schichtenaufbau eines Board Support Packages.

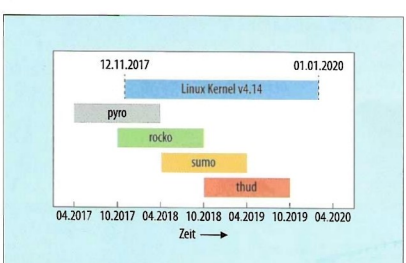

Bild 2. Release-Timeline für Linux-Kernel und Yocto-Versionen.

Updatefähigkeit: Was die Norm fordert

Die IEC 62443 beschäftigt sich im aktuell vorliegenden Stand in Part 2-3 mit dem Thema „Patch Management in the Industrial Automation Control System Environment". Hier wird explizit im normativen Rahmen von der Notwendigkeit der Update-Fähigkeit von Systemen gesprochen. Bringt man sehr vereinfacht die Details des Normenvorschlags auf diesen Punkt, dann lässt sich eine Gedankenkette aufbauen:

Systeme müssen auf Gefahren vorbereitet sein, die erst noch entstehen. Ein Ansatz könnte es sein, das betroffene System von vorneherein durch externe Maßnahmen zu schützen (Isolierung, externe Firewall). Der Normenvorschlag befasst sich im Wesentlichen mit der Ausgangsbasis, dass Commercial-off-the-shelf (COTS)-Technologie Einzug in Industrial Automation and Control Systems (IACS) gehalten hat und leitet hieraus die Notwendigkeit eines Patch-Managements auf Firmware-Ebene ab, insbesondere auch Bootloader und Betriebssystem.

Nun setzt Patch-Management das Vorhandensein von Patches voraus. Patches für Closed-Source-Betriebssysteme können nur vom Hersteller der Closed Source bereitgestellt werden. Verliert dieser das Interesse an einer (alten) Version, gibt es dafür keine Patches. Wenn es neue Versionen gibt, kann der Hersteller Forderungen an die Hardware stellen, die von der im Feld befindlichen Hardware nicht erfüllt werden. Patch-Management würde dann einen Tausch der Hardware bedingen.

Die Konsequenz: Nach Auffassung des Autors dieses Artikels ist die Forderung nach einem nachhaltigen Patch-Management effizient nur mit Open-Source-Lösungen zu erreichen. Man kann durchaus soweit gehen, zu behaupten, dass nachhaltiges Patch-Management zwangsläufig den Einsatz von Open-Source-Lösungen voraussetzt.

Management kein Weg an einem Umstieg auf Mainline vorbei. Es ist also nur eine Frage der Zeit, wann der Einstieg in Mainline erfolgen soll.

Software-Lifecycle-Management als Service

Der Aufwand, um Systeme im Feld ständig auf dem aktuellsten Software-Stand zu halten, kann enorm hoch sein. Die Risiken beim Ausrollen der Software müssen in den Griff gebracht werden. Als Lösung bietet Phytec seinen Kunden das Software-Lifecycle-Management – kurz SLCM – als individuelles Serviceangebot an (siehe **Kasten**). Das Ziel ist, über den gesamten Produktlebenszyklus hinweg zeitnah aktuelle Versionen des Kernels und der mittels Yocto erzeugten Distribution zu testen und bereitzuhalten, sodass sie im Bedarfsfall schnell und unkompliziert ausgerollt werden können. Dazu wird das kundenspezifische Board Support Package in einer Testfarm integriert. Entsprechend der mit dem Kunden vereinbarten Update-Strategie wird die Software auf dem Laufenden gehalten – in der Regel mit jährlichen Major Updates des Yocto Projects und Updates der LTS-Kernel-Version (long time stable) im zweijährigen Rhythmus. Außerdem werden kontinuierlich Security- und Bugfixes eingespielt. In der Testfarm wird die Hardware mit automatischen Nightly-Builds getestet, sodass etwaige Konflikte schnell erkannt und zeitnah behoben werden können. Gleichzeitig wird die Übereinstimmung der BSPs mit den BSP-Spezifikationen in der Testfarm kontinuierlich überprüft. Sämtliche Resultate werden in Testprotokollen festgehalten.

Durch dieses Vorgehen hat der Kunde zu jedem Zeitpunkt Zugang zu einer aktuellen und getesteten BSP-Version, ohne die Risiken kontinuierlicher Updates im Feld. Routinemäßig oder sobald eine relevante Sicherheitslücke erkannt wird, kann der Kunde das BSP mit seinen eigenen Software-Applikationen testen. Positiver Nebeneffekt ist die klare Trennung von BSP, Middleware und Applikationssoftware, mit der die einzelnen Schichten im Bedarfsfall individuell behandelt werden können, ohne dass sich Fehler durch nicht berücksichtigte Abhängigkeiten ergeben. In der Summe wird das zügige Ausspielen auf die im Feld befindlichen Geräte so deutlich vereinfacht.

Deployment von Software-Updates vorbereiten

Zusätzlich unterstützt werden Software-Updates durch die Vorbereitung des RAUC (Robust Auto-Update Controllers) in allen aktuellen BSPs von Phytec. Der Update-Client sorgt für die zuverlässige Installation von signierten BSP-Updates auf den Embedded-Systemen und wird von Yocto im Meta-RAUC Layer unterstützt. Auf dem Host-System können

Hersteller-BSP vs. Mainline in der Phytec-Praxis

Als Modulhersteller setzt Phytec auf Linux und ist dabei ein unbedingter Verfechter der Mainline-Unterstützung. Deshalb hat Phytec zum Beispiel maßgeblich das Mainlining des i.MX27 finanziert und ist Gründungsmitglied der OSADL-Genossenschaft, die beispielsweise die Unterstützung von Echtzeit-Features im Linux-Mainline-Kernel vorantreibt.

Selbstverständlich steht die Wirtschaftlichkeit der Produkte für den Modulhersteller im Vordergrund. Die Abwägung zwischen Investment in Mainline einerseits und der optimalen Anpassung eines Hersteller-BSP muss für jeden Controller individuell getroffen werden und steht in direktem Kontext zur jeweiligen Qualität der zur gegebenen Zeit vorhanden Mainline-Version.

Phytec setzt dabei auf eine Kombination aus früher Marktpräsenz und Nachhaltigkeit. In der frühen Phase steht das Hersteller-BSP im Fokus; die Arbeiten der Phytec sind aber stets perspektivisch auf Mainline ausgerichtet. In der Praxis zeigt sich das an der Tatsache, dass für einige Controller gleich zwei Varianten von Linux (über einen gewissen Zeitraum) gepflegt werden: eine Hersteller-BSP-Implementierung und eine Mainline-Version.

mittels des Tools BSP-Updates erstellt, geprüft und modifiziert werden. Phytec unterstützt Kunden sowohl bei der Implementierung der Update-Mechanismen als auch beim Schaffen einer entsprechenden Infrastruktur – von der RAUC-Konfiguration über das Einrichten von Cloud-Services bis hin zum Schutz der Hardware vor dem Aufspielen von Schadsoftware. Dazu bietet das Unternehmen z.B. Hardening an und implementiert Secure Boot. Außerdem berät Phytec seine Kunden bezüglich weiterer Voraussetzungen für die Erfüllung von Security-Anforderungen im Design von Hardware und Software – vom Schlüsselhandling über den Umgang mit Zertifikaten bis hin zum Einrichten und Managen der Cloud-Umgebung für das Ausspielen der Updates.

In einer vernetzten Welt müssen industrielle Systeme mit immer neu entstehenden Sicherheitsherausforderungen schritthalten können. Wer ein Sicherheitsleck nicht schließen kann, hat verloren. Open Source und Mainline schaffen die Basis, Systeme im Feld aktuell halten zu können. Die Reaktionszeit zwischen Bekanntwerden einer Gefahr und dem Ausrollen eines Updates muss so kurz wie möglich bleiben. Software-Lifecycle-Management als ein kontinuierlicher Prozess ist der Schlüssel, Systeme im Feld aktuell zu halten und auf Gefahren in kürzester Zeit mit stabilen Fixes reagieren zu können. *jk*

Axel Berghoff arbeitet seit mehr als 20 Jahren für Phytec Messtechnik. Er betreut Großkunden und strategisch wichtige Kunden, mit denen er gemeinsam optimale Projektlösungen erarbeitet – vom Einsatz von Standardmodulen bis hin zu Komplettentwicklungen mit umfassenden Leistungen im Bereich Software und Lifecycle-Management.
axel.berghoff@phytec.de