LINUTRONIX

Failure Safe Update

- Sie suchen eine Möglichkeit für ein sicheres Update ihres Embedded Systems?
- Sie wollen auch das Betriebssystem updaten?
- Sie wollen sogar den Bootloader updaten?
- Sie wollen flexibel sein bei der Wahl des Update Mediums?
- Sie wollen einen gesicherten Datenaustausch?
- Sie wollen einen schnellen Bootvorgang?

Embedded Systeme haben spezielle Anforderungen

Bei Embedded Systemen werden das Betriebssystem und alle Anwendungen fest im Gerät gespeichert. Dies ist solange kein Thema, solange die Software nicht geändert werden muss. Was aber, wenn die Applikation oder das Betriebssystem geändert werden muss? Oder der Bootloader zu ändern ist? Man könnte meinen, das wäre doch kein Problem. Man lädt einfach die neue Software und schreibt sie in den (Flash-)Speicher. Im Gegensatz zu Anwendungen in der IT Welt kann man aber nicht immer jemand mit IT Wissen zum Gerät schicken, da es entweder zu viele Geräte sind oder die Geräte sich an einem Ort befinden, den man physikalisch nur mit hohem Aufwand aufsuchen kann.

Auch müssen bei Embedded Geräten verschiedene Boot-Devices berücksichtigt werden - SD-card oder USB-Stick oder Netzwerk oder was auch immer nutzbar ist. Und was, wenn sofort eine Benutzerschnittstelle angezeigt werden muss? Oder eine spezielle Schnittstelle während des Bootens für den Service gescannt werden soll. Oder verschiedene Firmwaren geladen werden müssen, z. Bsp. für eine weitere CPU oder ein FPGA.

Problematisch wird es in dem Moment, in dem der Schreibvorgang in das Flash nicht funktioniert. Und im Anschluss daran das Gerät nicht mehr nutzbar ist. Einschicken zum Hersteller? JTAG anschließen? Undenkbar und mit viel zu hohen Kosten (Produktionsausfall etc.) verbunden.

Die Embedded Welt benötigt daher auf Grund Ihrer spezifischen Anforderungen auch spezifische, auf sie zugeschnittene Lösungen.


Choose the right tool ...

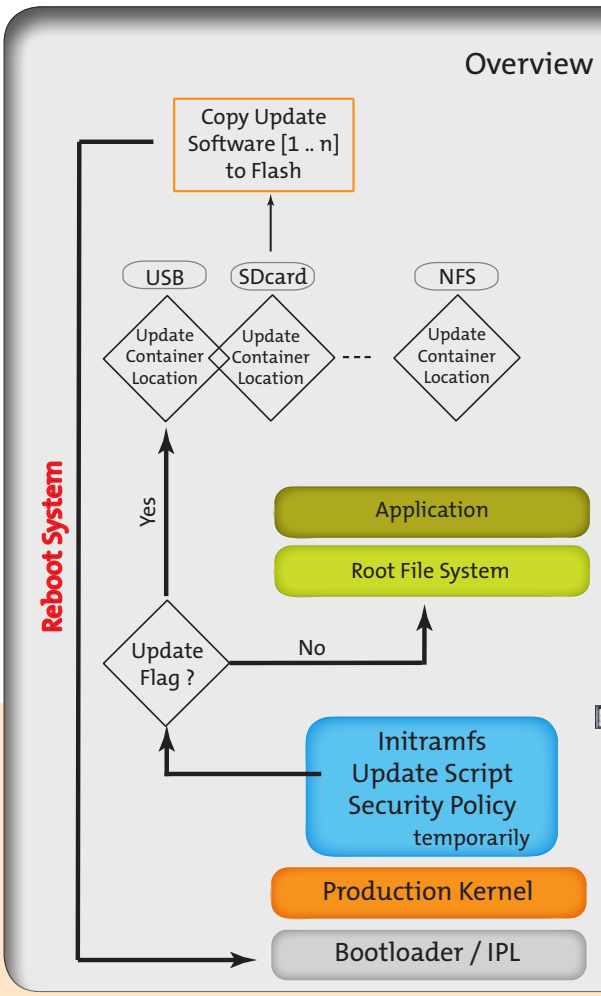
Ein Update muss einfach durchführbar sein. Der Nutzer einer Maschine ist in der Regel kein IT Spezialist. Jeder Nutzer ist daher dankbar, wenn es einfach geht. Und die Vorgehensweise kann sich auch von Fall zu Fall unterscheiden. Je nachdem ist der Ansatz, alles zentral einzuspielen optimal, oder jedes Gerät muss einzeln auf den neuesten Stand gebracht werden.

Aber diese Entscheidungen überlassen wir Ihnen und Ihren Kunden.Genau wie wir es Ihnen überlassen, ob Sie nun ein Update mittels eines einzigen Images durchführen oder aufgeteilt in mehrere Images oder per Paketmechanismus. Für jeden der genannten Ansätze gibt es viele Gründe dafür oder dagegen. Wir liefern Ihnen das Werkzeug, damit Sie Ihre Entscheidung so umsetzen können, wie es für Sie richtig ist.

Bootloader oder Linux?

Der scheinbar einfachste Weg, ein Update durchzuführen ist es, den Bootloader hierfür zu nutzen. Dies geht mit wenig Programmieraufwand und schaut auf den ersten Blick sehr zielführend aus. Der Bootloader selbst kann hierbei nicht upgedated werden, aber das ist in diesem Fall gewollt, da systembedingt.

Oft vergessen wird aber, dass im Bootloader nicht dieselbe Unterstützung vorhanden ist wie in Linux. Beispielsweise werden nicht alle Schnittstellen oder alle Protokolle unterstützt. Das Problem ist lösbar, indem die fehlende Software (Treiber, Protokolle etc.) für den Bootlaoder ein weiteres Mal entwickelt wird. Eine (teure und auch) fehlerträchtige Angelegenheit. Auch ist die Bedienbarkeit des Bootloaders, d.h. die User-Schnittstelle (GUI), nur rudimentär, d.h. nur als Kommandozeile, vorhanden.

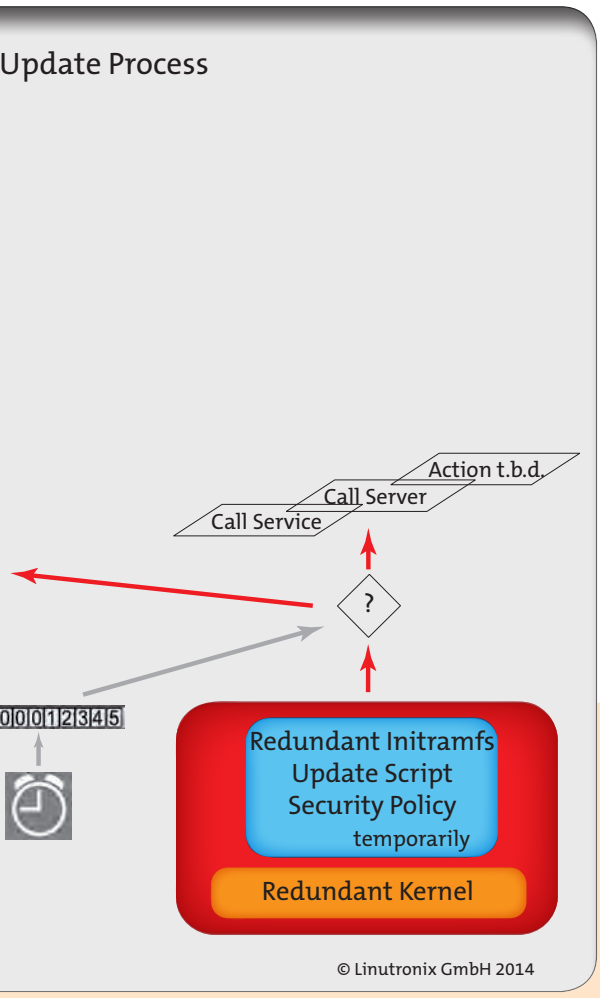


Die Integration einer komplexeren Updatelogik ist ebenfalls nicht vorgesehen.

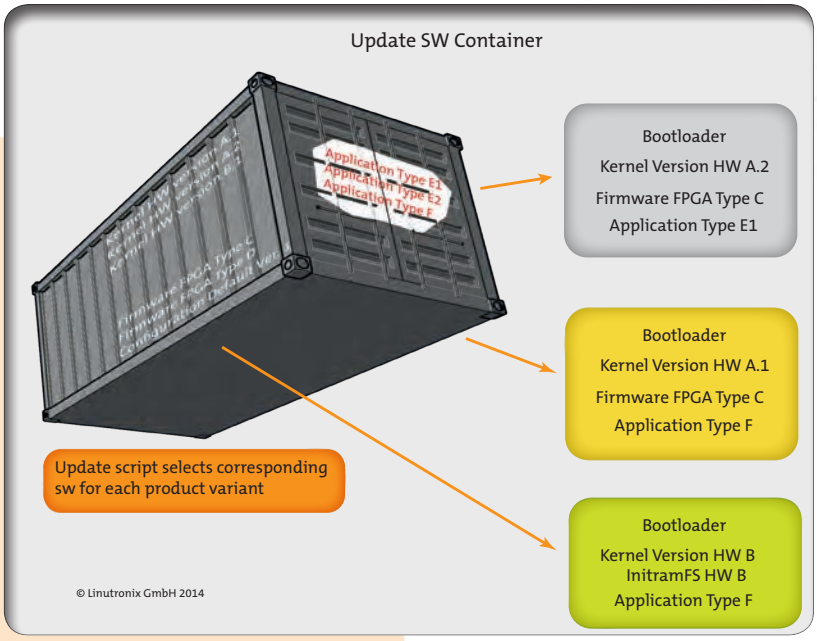
Lösung mit Linux

So vielfältig und unterschiedlich der Updatevorgang auch zu sein scheint, letztendlich lässt er sich immer auf folgendes generisches Schema reduzieren: Es wird zuerst geschaut, ob ein Update durchzuführen ist und dann, in einem zweiten Schritt, wird das Update von einem wo auch immer sich befindenden Medium auf das System geladen. In einem dritten und letzten Schritt wird dann das Update in das eigentliche Bootmedium kopiert und steht anschließend bei einem Neustart zur Verfügung.

Das Prinzip ist einfach, die Umsetzung vielfältig, da jeder seine eigenen Anforderungen hat. Wir haben, um uns und Ihnen die Arbeit einfacher zu machen, ein Framework geschaffen, mit dessen Hilfe wir die unterschiedlichsten Anforderungen berücksichtigen können.



Die Daten für das Update werden in einen Container gepackt. Dabei kann es sich um Daten für den Bootloader, das Betriebssystem, die Anwendung(en) oder auch um Firmware für zum Beispiel ein FPGA handeln. Hierbei können verschiedene Software Versionen für unterschiedliche Systemvarianten in einem



Container transportiert werden. Es spielt keine Rolle, ob die Software als „Image“ oder als „Paket“ gespeichert wird. Eine XML Beschreibung liefert die notwendigen Informationen zu den einzelnen Datenpaketen mit. Die Beschreibung enthält unter anderem Informationen, zu welcher Hardwareversion ein einzelnes Update gehört, zu welcher Komponente der Hardware und um welche SW-Version es sich handelt. Ein einziges Update (Container) kann also für viel Hardware- wie auch Softwareversionen genutzt werden. Dies erlaubt eine einfache Handhabung von Updates.

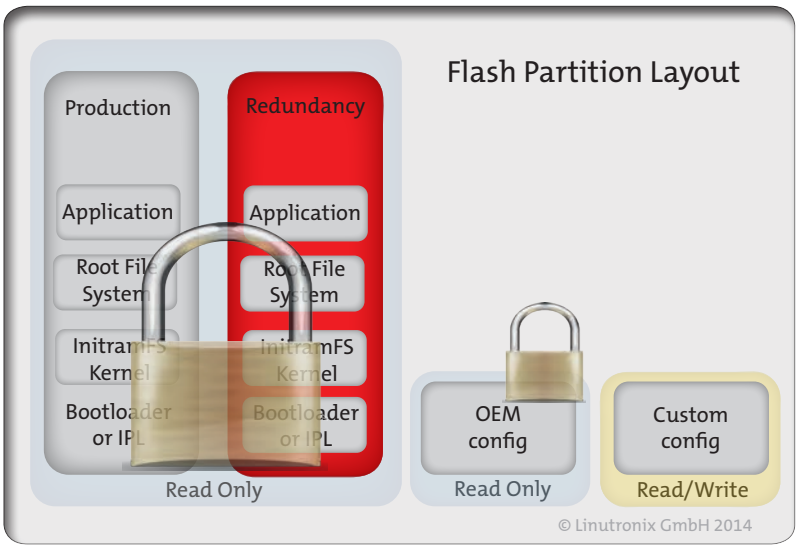
Ein auf dem Linuxsystem laufendes Dienstprogramm prüft beim Start, ob ein Update erfolgen soll oder nicht. Ist kein Update gewünscht, dann wechselt das Betriebssystem das Rootfilesystem und nutzt ab diesem Moment das Produktivsystem und startet die Anwendung(en). Soll jedoch ein Update durchgeführt werden, dann verbleibt Linux im InitramFS. Die möglichen Quellen für das Update (siehe auch weiter vorne) werden auf das Vorhandensein eines Containers gescannt. Sobald ein gültiger Container gefunden wird, wird die XML Beschreibung ausgewertet und die einzelnen Komponenten des Containers, die für diese spezifische Hardware gedacht sind, an die entsprechenden Speicherorte geschrieben.

Hierzu wird der Inhalt in den Arbeitsspeicher kopiert und in einem 2. Schritt von dort auf das eigentliche Bootmedium geschrieben. Sollte das Update nicht zur Gänze in den Speicher passen, dann wird dieser Vorgang in entsprechend passende Teilstücke aufgeteilt.

Nach Abschluss des Schreibens in das Bootmedium stehen zwei Alternativen zur Auswahl, wie weiter verfahren werden soll. Es kann nun entweder in das Produktivsystem gewechselt werden oder es wird ein System Neustart initiiert und damit die neue Software sofort zum Einsatz gebracht.

Redundanz und Watchdog

Bei Embedded Geräten besteht die Gefahr, dass das Schreiben auf das Bootmedium fehlerhaft sein kann (z. Bsp. durch plötzlichen Verlust der Stromversorgung). Oder die neue Software wurde nur teilweise oder fehlerhaft übermittelt. Das Ergebnis ist das Gleiche: das System kann mit der neuen Software nicht gebootet werden.



Aus diesem Grunde sind das Betriebssystem und die Anwendung oftmals doppelt auf dem Gerät vorhanden (siehe Bild oben). Zumindest das Betriebssystem. Denn dann erkennt ein Watchdog, wenn der Bootvorgang fehlschlägt und startet in diesem Falle beim nächsten Bootvorgang das redundante System. Das weitere Vorgehen in diesem Falle erfolgt gemäß der Festlegung des Nutzers. So kann versucht werden, das Update ein zweites Mal einzuspielen und erst beim wiederholten Scheitern wird der Benutzer um Hilfe gebeten bzw. informiert oder das System stoppt und gibt eine entsprechende Fehlermeldung auf dem Bildschirm (sofern einer vorhanden ist) aus. Hier sind die Möglichkeiten nahezu unerschöpflich und können jedem denkbaren Szenario angepasst werden.

Soll auch der Bootloader selber ersetzbar sein, so muss auch dieser redundant vorhanden sein. Das korrekte Verhalten wird auch in diesem Falle von einem Watchdog überprüft. Nur ist nun der Bootloader doppelt vorhanden und wenn erkannt wird, dass der originäre Loader nicht funktioniert, wird der redundante Bootloader gestartet. Die genaue Implementierung dieses Verhaltens hängt u.a. von der gewählten CPU ab.

Secure Boot & Secure System

Das Thema Systemintegrität (also die Unversehrtheit) eines Embedded Systems wird, vor allem im Zeitalter von Internet of Things, immer wichtiger. Die möglichen Maßnahmen zur Sicherung der Integrität sind vielfältig. Sie reichen von der Signaturüberprüfung beim Laden von Kernelmodulen bis hin zu Secure Boot. Hier wird nur ein signierter Kernel plus das dazugehörige System vom Bootloader geladen und gestartet. Der Bootloader selber enthält dabei das sog. Wurzelzertifikat. Soll das System auch noch gegen Angriffe mit physikalischer Zugriffsmöglichkeit geschützt werden, so ist der Einsatz spezieller Hardware (TPM – trusted platform module) notwendig.

Der hier vorgestellte Updatemechanismus unterstützt diese Security Methoden bei Bedarf dadurch, dass er nur das Nachladen von signierten Modulen oder Images erlaubt. Somit ist auch im Falle eines Updates sichergestellt, dass nur autorisierte Software zum Einsatz gelangt.

Bootzeit

Alle gezeigten Möglichkeiten haben entweder keinen oder nunur einen verschwindend kleinen Anstieg der Bootzeit zur Folge. Fastboot mit Zeiten im einstelligen Sekundenbereich sind damit realisierbar.

Haben wir Ihr Interesse geweckt? Wollen Sie mehr wissen? Rufen Sie einfach an, oder senden Sie uns eine Email.

07/2015_V1.1

LINUTRONIX GMBH
Auf dem Berg 3 | D-88690 Uhldingen - Mühlhofen
Telefon +49 7556 4521 890 | Fax +49 7556 919 886
info@linutronix.de | www.linutronix.de

LINUTRONIX
LINUX FOR INDUSTRY