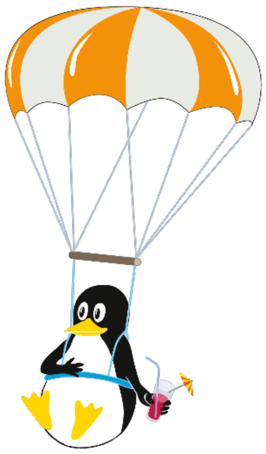


Enterprise class update for Embedded Systems

Fail Safe, Secure Update

This page remains intentionally blank

Main features

Modular

No fix layout, easy adaption to your needs

Container

1010011 1001101 0101010

Container & App Update

Security

1010011 1001101 0101010

X.509 support
Signed Update
Encrypted Update

Ruggedness

1010011 1001101 0101010

Atomic update, complete redundancy

Delta Update

1010011 1001101 0101010

Reduces data volume, best for CDN

Flexible

1010011 1001101 0101010

Supports various update media

Gateway

1010011 1001101 0101010

Delivery gateway for underlying nodes

Expandable

1010011 1001101 0101010

Supports new update mechanism (TUF, Uptane)

This page remains intentionally blank

Content

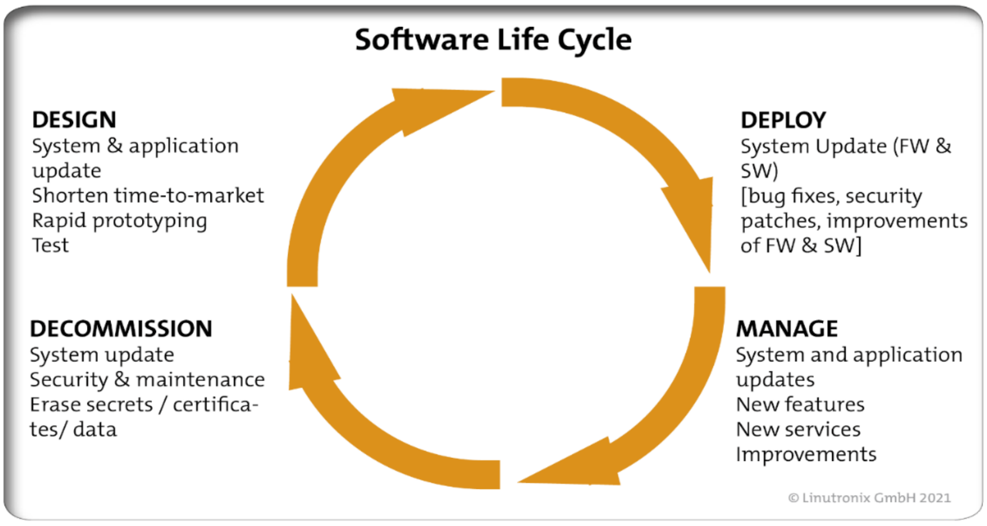
(OTA) Updates

IIoT device manufacturers have recognized the benefits of open-source operating systems and applications, but have also found that maintaining and servicing the devices can be a real challenge.

IIoT devices are not only quick to develop, they are just as easy to distribute in large numbers around the world. But that also means you need a solution to manage all those devices. Preferably, devices from different manufacturers. Maintaining these devices can easily become a chore if you don't have the right tools.
How can a solution for these requirements look like? How to bring updates for the OS as well as the application securely to the devices without needing physical access? An OTA update plus the associated device management is the solution. And this is where our Linutronix approach comes into the game.

Over-the-air (OTA) updates, or remote updates, are now common practice in the age of the IIoT.

And not only when regulatory requirements such as NIST, KRITIS, ETSI EN303645 or IEC62443 dictate this. A software update can also be part of a business model. In the case of smartphones, PCs or IT software, it is common to be able to later buy functions via a software update. This approach is slowly being adopted in the typical areas of embedded devices. Primarily the long lifetimes of 10 or more years of a typical embedded device cause the necessity for updates.



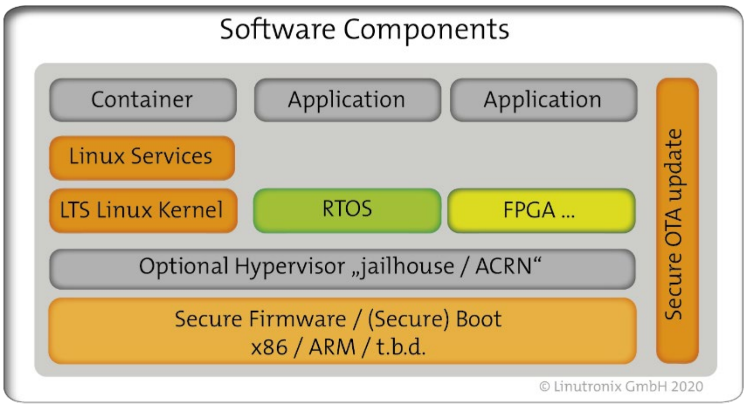
Software Lifecycle

A modern update system must support the complete lifecycle (see picture above) of the software of an embedded device. A closer look at the update software identifies the two components server and target system (client). Both must fulfill different requirements.

On the client or respectively on the device side, the challenges lie in particular in the very strongly diversified hardware and software situation. In addition, there is the question of physical accessibility of the devices, which is not always available or only at very high cost. This results in very high and diverse requirements for an OTA solution, which must also be customizable.

Which software components of a client are supposed to be accessible by an update?

The picture below (software components) shows a typical software structure on a modern system. We see here the firmware, consisting of boot loader and operating system as well as possible components such as the hypervisor, an FPGA code or the operating system for other cores. In addition, there is the application software, which today is often available in the



Software components to be updated

form of containers. An update solution should be able to exchange all these components, if possible, independently of each other. Security in terms of the target's data integrity should be maintained.

The update management should also be able to support a fleet management. In this case, the update is first delivered to only a few devices (the so-called canary devices) in order to evaluate the quality. Only when the percentage of successful installations is high enough, the update is delivered to a freely configurable number of devices (the "fleet").

Device management goes even further. Here, it is a matter of recording the status of the devices and having them available in "real time". Measures can then be derived and implemented from these findings. This can be in the area of predictive maintenance, new functions that are subsequently sold (sale after delivery) or the detection of security problems.

Customers want to see the following features, for example:

- Rolling out updates - per device, per group, according to freely selectable criteria
- Successive roll-out - depending on the success of the previous campaign
- Monitoring of the device health (utilization, memory consumption, uptime, etc.)
- Operating system and application updates, secure and reliable
- Application management: Installation, decommissioning, configuration
- Applications loggings
- Device specific telemetry info
- On demand specific debug support for OS and application (deep inside inspection)
- Decommissioning support for the complete device incl. secrets

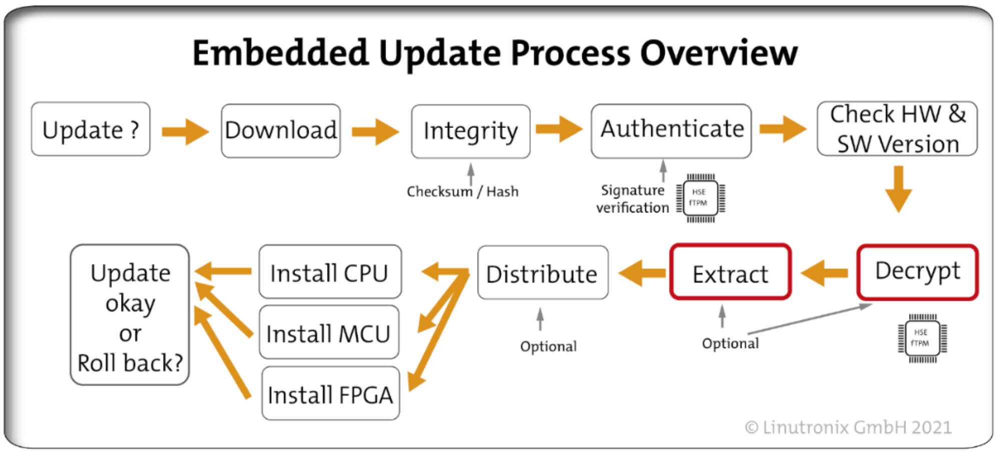
All this is accomplished by our OTA solution, which is based on open-source software.

Update Workflow

The demands on an update system that can be derived from this situation can be summarised as follows:

- Authentication of the update server (security endpoint feature)
- Signed and optionally encrypted updates (data integrity)
- Verification of each update locally before download
- No import of outdated, faulty updates (downgrade)
- Power failure security and robustness against faulty data transfers
- Secure update even if device is not always connected to the network
- Rollback option, also for downstream devices
- Multiple images for multiple units in one update
- Forwarding of updates to downstream devices (FPGA, CPUs ...)
- Optimal use of memory
- Optimal use of existing bandwidth (CDN-friendly systems) thanks to delta updates
- Update as images, as partition updates, as packages and as files
- Feedback update and information exchange to the management server

We have implemented these requirements on the basis of an open source tool, which has received appropriate extensions from us. Our tool can be easily integrated into Yocto or Debian based build environments. An update then runs as shown below.



The most important component is the rollback at the end of the process, if the update was not successful. It is not important here whether the rollback is enabled thanks to a completely redundant approach (A/B update), an Android-like approach (with an update or rescue kernel) or a delta update with archive on the device. It is important to always put the device in a functional state.

For the adjustment to the specific conditions it is recommended, if various actions can be carried out before and after the installation. These so-called pre- and post-install scripts can be freely created with our OTA approach.

If you want to read the complete document, please go to our download page: https://linutronix.de/download/download.php