About me

- Software Engineer at Siemens Corporate Technology
- Embedded Linux Corporate Competence Center
  - OSS contributors
  - Part of the OSS community (4 speakers in this event)
- Interested in enabling embedded devices to ride the „container“-wave
Agenda

- Why updating?
- Architecture
- Components
- Firmware updates
- Conclusion
State of Update Affairs in Industrial Domains

- Devices are either
  - disconnected or
  - in isolated networks
- On-site updates
- Very long life
- Difficult to reach
- Infrequent updates
... but the Internet of Things is coming

- Trend: More and more devices getting connected (including industrial products)
- Number of devices to manage explodes ⇒ remote management required
- Attack surface increases due network exposure ⇒ updates frequency will increase due to security issues
- There’s always a bug to fix
- Additional expectations due to technology exposure:
  - Easy features addition
  - Easy bugfixing
10000 Feet Architecture

Backend

hawkBit

Device

SWUpdate

Why updating?
10000 Feet Architecture

Why updating?

Architecture

Components

Firmware updates

Conclusion
Workflow

1. Devices poll requesting artifacts (firmware updates)
2. Backend can reply
   - either no updates available
   - or list of updates with download URLs
Workflow

1. Devices poll requesting artifacts (firmware updates)
2. Backend can reply
   - either no updates available
   - or list of updates with download URLs
3. Device downloads updates
4. Device processes updates
5. Device report success/failure
SWUpdate

- Neither convincing OSS nor commercial alternatives back then
- Make vs. “buy” OSS decision
- Developed and open-sourced by a well-established player in the OSS arena DENX with experience in industrial domains
- Convincing feature set
  - Full power of Linux userspace for updates
  - Extensible
  - Good integration with U-Boot, support for others possible
  - …
- Battle-proof software
- … ⇒ Easy decision for “buy”
**hawkBit**

- Neither convincing OSS nor commercial alternatives back then
- Make vs. “buy” OSS decision
- Originally developed by Bosch and released as OSS under the umbrella of the IoT working group of the Eclipse Foundation
- Bosch as Siemens in industrial domains
- Shifting from device-managed provisioning to remote-managed provisioning
- Convincing present and future feature set:
  - Easy integration via REST-APIs
  - Direct or indirect devices connection
  - External artifacts repository
  - Reporting and monitoring
- Young project working on stabilization and new features
- … ⇒ Decision for “buy”
Firmware definition

- Linux base system that makes system runnable
Supported update strategies

- Trade-off between time and space
- Two extremes in IoT domain:
  - With two firmware partitions (best for time, worst for space)
  - With one firmware partition (best for space, worst for time)
- Hybrid solutions possible by reducing firmware storage footprint
- Minimalistic firmware images reduce the differences between both with regard to space
Two Firmware Partitions

- No device bricking and rollback possibility (only 1 version back)
- Double firmware storage footprint
- Minimal downtime (usually only reboot)
- Update cancellation in case erroneous/manipulated images, keeping working version
Two Firmware Partitions: Start

hawkBit

Bootloader
Blue

Firmware v1
SWUpdate

Others
Two Firmware Partitions: Download and Flashing

- **Bootloader**
  - Firmware v1
  - SWUpdate
  - Others

- **Blue**
  - Firmware v1
  - SWUpdate

Why updating?
- Architecture
- Components
- Firmware updates
- Conclusion
Two Firmware Partitions: Checks and Activation

- Bootloader
  - Firmware v1
    - SWUpdate
  - Others

hawkBit

- Bootloader
  - Firmware v1
    - SWUpdate
  - Flashing...
  - Others

- Bootloader
  - Green
  - SWUpdate
  - Firmware v2
  - Others

Why updating? Architecture Components Firmwares updates Conclusion

Unrestricted © Siemens AG 2016
Page 17 2016.10.11
Silvano Cirujano Cuesta
Two Firmware Partitions: End
One Firmware Partition

- No device bricking
- Single firmware storage footprint
- Relatively long downtime
- Only rescue from erroneous/manipulated images is recovery mode

Bootloader

Lightweight Linux (LWL) + SWUpdate

Firmware

Others (applications, data, config...)

Why updating?  Architecture  Components  Firmware updates  Conclusion
One Firmware Partition: Start

Why updating?

Architecture

Components

Firmware updates

Conclusion

Unrestricted © Siemens AG 2016

Page 20 2016.10.11

Silvano Cirujano Cuesta

Corporate Technology
One Firmware Partition: Download and Flashing

Why updating?
Architecture
Components
Firmware updates
Conclusion
One Firmware Partition: Checks and Activation

### Why updating?
- Architecture
- Components
- Firmware updates
- Conclusion

**hawkBit**

Bootloader → Lightweight Linux

1. **Firmware v1**
2. **Firmware v2**

Lightweight Linux

SWUpdate

Flashing...

Others

---

**Why updating?**

- Architecture
- Components
- Firmware updates
- Conclusion

Silvano Cirujano Cuesta
Conclusion

- 100% Open Source Software
- Open communities
  - We were welcomed in both
- Current focus on firmware updates, but software provisioning in general possible
- Modular architecture:
  - hawkBit can be contacted by other clients, via other protocols, …
  - SWUpdate can be extended to support other servers, via other protocols, …
- Future features:
  - Split preparation/realization to fit into maintenance windows
  - Synchronization of Device and Backend
  - Asymmetric key signatures
Contact Information

Silvano Cirujano Cuesta
Software Engineer
Corporate Technology
Embedded Linux Corporate Competence Center

E-mail: silvano.cirujano-cuesta@siemens.com

Internet
siemens.com/corporate-technology

Intranet
intranet.ct.siemens.com