SWUpdate

Updating an Embedded System
• Me:
  - Software Engineer at DENX, Gmbh
  - U-Boot Custodian for NXP's i.MX
  - Focus on Linux embedded
  - Author of FOSS SWUpdate
Do we update?
Local Update
Push Software

Software Update

Software Upload
Update Firmware: Browse... No file selected.

Reboot Device

Status Messages

Network
Pull Software

Network
Deployment systems
Rescue system

Network
Requirement of updater ES

- Power-off safe
- Must not brick the device
- Atomic: must not apply half an update
- Secure
  - Signing images and verification of images
  - Prevent that device can be hijacked
Requirements - 2

- Remote unattended update
- Update of bootloader, kernel, filesystem
- Failsafe, Apply / rollback system updates
- It should take care of most important law

Murphy’s Law

If it can go wrong, it will go wrong.

As much as possible!
Components to be update

- Bootloader
- Kernel
- Root filesystem
- System Application
- FPGAs
- Microcontroller, etc.
- Configuration
Double-copy

Bootloader

Standby copy

Running copy

SWUpdate

Config Data, etc.
Combine methods

SPI NOR Flash

- Ramdisk
- Kernel for SWUpdate
- Env
- U-Boot

~4MB

8MB

Copy-A

Copy-B

NAND / eMMC / SD
SWUpdate

- Project started end 2014
- GPLv2, client library LGPLv2
- Often delivered together with BSP
- In the meantime:
  - ~40 developers sent contribution
  - Release cycle 3 months
  - One of Yocto updater:
  - Used by many devices in field

Deeds, not words!
Features - Basis

- Atomic update
- Embedded media
  - eMMC, SD
  - Raw NAND, UBI, NOR, SPI-NOR
- Single image (SWU) for multiple devices
- Power-Off safe
- Hardware-Software check

Deeds, not words!
Features - Interfaces

- Local Interface
- Remote interface / OTA
  - integrated web server (PUSH mode)
  - Backend: integrated REST client connector to hawkBit (PULL Mode)
  - remote server download (PULL Mode)
  - Custom interface (client library, LGPL)

Deeds, not words !
Features - Extended

● Integrated LUA interpreter
  – modular with plugins in LUA

● Embedded Buildsystems
  – Integrated in Yocto with meta-swupupdate
  – Officially supported by Buildroot

● Support for bootloader
  – U-Boot
  – GRUB

● Small footprint

Deeds, not words!
Features - next

- Fallback with bootloaders
- Image updater and file updater
- Interface to report progress
- Uses Kbuild for configuration
- Streaming without temporary copies

Deeds, not words!
Features - Security

- HTTPS protocol
- Use Certificates for server verification
- Signed Images
- Encrypted artifacts
- Privilege separation
  - Installer usually runs as root
  - Network processes runs on different user

Deeds, not words!
Structure SWU image

- CPIO format for simplicity
- `sw-description` describes update
- Images data / artifacts
sw-description

- Describe how to install a release
- Different parser
  - libconfig (default)
  - JSON
  - Custom (LUA)
  - Example: XML parser using LUAExpat
SWUpdate’s architecture

- SWUpdate
  - LUA Interpreter
  - Utilities
  - Progress
  - Notifier
  - Tracer / LOG
  - Handler manager
  - RAW
  - UBI
  - MTD
  - JSON Parser
  - LIBCONFIG Parser (default)
  - Custom Parser (LUA)
  - BootEnv
  - Archive
  - Installer Thread

SWUpdate IPC

Local

Suricatta

WebServer

Downloader

Custom Process
Config

Swupdate Configuration

Arrow keys navigate the menu. <Enter> selects submenus --- (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <? > for Help, </> for Search.

Legend: [*] built-in [ ] excluded <M> module < > module capable

Swupdate Settings --->
Bootsloader (U-Boot, ..) (None) --->
[*] Enable image downloading
[*] Allow to add sha256 hash to each image
[ ] Enable verification of signed images
[ ] Images can be encrypted with a symmetric key
Suricatta --->
[*] Enable webserver
  Webserver Features --->
  Archival Features --->
  Parser Features --->
  Image Handlers --->
Sw-description : structure

Software = {
    Version = “1.0.0”;
}

myhw = {
    hardware-compatibility : [ “1.0”, “1.1”, “1.3”];
}

images : ( {
    filename = “rootfs.ext4.gz”;
    device = “/dev/mmcblk0p1”;
    type = “raw”;
});

files : ( {
    filename = “archive.tgz”;
    type = “archive”;
    Path = “/usr/share/myapp”;
});

scripts : ( {
    filename = “postinstall.sh”;
    type = “shellscript”;
});
One image for multiple devices

Software = {
    Version = "1.0.0";
}

hmi = {
    hardware-compatibility : [ "1.0", "1.1", "1.3" ];
    images : (
        { 
            ........
        }
    )
}

TypeA-1 = {
    Hardware-compatibility : [ "2.1", "2.2", "3.3" ];
    images : (
        { 
            ........
        }
    )
}

Header
Target: HMI
Target: TypeA-1
software =
{
    version = "0.1.0";
    myhw = {
        hardware-compatibility: [ "1.0"];
        stable : {
            copy1 : {
                images: ( {
                    filename = "core-image-full-cmdline-twister.ubifs";
                    type = "ubivol";
                    volume = "rootfs1";
                    sha256 = "@core-image-full-cmdline-twister.ubifs";
                }),
                {
                    filename = "ulmage-twister.bin";
                    type = "flash";
                    device = "/dev/mtd10";
                    sha256 = "@ulmage-twister.bin";
                });
            scripts: ( {
                    filename = "test.lua";
                    type = "lua";
                    sha256 = "@test.lua";
                });
        uboot: ( {
            name = "nandroot";
            value = "rootfs1";
        },
        {
            name = "kernelpart";
            value = "kernel1";
        } );
    };
};
collections: {
  images: (
    {
      filename = "core-image-full-cmdline-twister.ubifs";
      type = "ubivol";
      volume = "rootfs2"
      installed-directly = true;
      sha256 = "@core-image-full-cmdline-twister.ubifs";
    },
    {
      filename = "ulmage-twister.bin";
      type = "flash";
      device = "/dev/mtd11";
      sha256 = "@ulmage-twister.bin";
    }
  ),
  scripts: (
    {
      filename = "test.lua";
      type = "lua";
      sha256 = "@test.lua";
    }
  ),
  uboot: ( 
    {
      name = "nandroot";
      value = "rootfs2";
    },
    {
      name = "kernelpart";
      value = "kernel2";
    }
  )
};
Handlers

- flash devices in raw mode (both NOR and NAND)
- UBI volumes
- Archives (tarballs,..)
- raw devices, such as a SD Card partition
- U-Boot environment
- LUA scripts
- Shell scripts
- Remote handler

But you can also create your own ...
Embedded Script

- Executive part of sw-description
- Description changed at runtime
- Use cases for Embedded Script:
  - Check if an update is allowed
  - Set Partitions
  - Pre-install script
Rollback

- Together with U-Boot “bootcounter”
- Increment count in bootloader
- Reset after successful update / boot
- If reboots and count > threshold
  - Bootloader knows update / boot failed
  - Bootloader loads alternate boot
Security: Signed images

Yocto BuildSystem

Meta-swupdate

Signed Image

Authentication Key

Update Agent (SWUpdate)

Public Key
Security: Encrypted images

Yocto BuildSystem

Meta-swupdate

Encrypted Artifact

Symmetric Key

Update Agent (SWUpdate)

Network

SWU
Suricatta mode

Installer

Intermediate Layer

Backend Agent X (who knows ?)

Backend Agent 1 (Hawkbit)

Network
Automatic SWU Image build

- meta-swupdate to build swupdate and swu
- Rescue image recipe
- Provides a class to automatically generate and sign a release image SWU
Creating own SWU

DESCRIPTION = "Example Compound image for beaglebone"
SRC_URI_beaglebone = "file://sw-description"

inherit swupdate

LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COREBASE}/LICENSE;md5=4d92cd373abda3937c2bc47fbc49d690 
file://${COREBASE}/meta/COPYING.MIT;md5=3da9cfcbb788c80a0384361b4de20420"

# IMAGE_DEPENDS: list of Yocto images that contains a root filesystem
# it will be ensured they are built before creating swupdate image
IMAGE_DEPENDS = ""

# SWUPDATE_IMAGES: list of images that will be part of the compound image
# the list can have any binaries - images must be in the DEPLOY directory
SWUPDATE_IMAGES = "
core-image-full-cmdline
"

# Images can have multiple formats - define which image must be
# taken to be put in the compound image
SWUPDATE_IMAGES_FSTYPES[core-image-full-cmdline] = ".ext3"

COMPATIBLE = "beaglebone"
SWUpdate Roadmap

- Extend community
- SWUpdate as Updater Gateway
- Dynamic LUA Handlers / new Handlers
- Hardware Keys / TPM for decryption
- Delta update
- Chain Handlers for single artifact
- Add other backends, support multiple servers
- A new modern Website
Questions

http://sbabic.github.io/swupdate/
swupdate@googlegroups.com