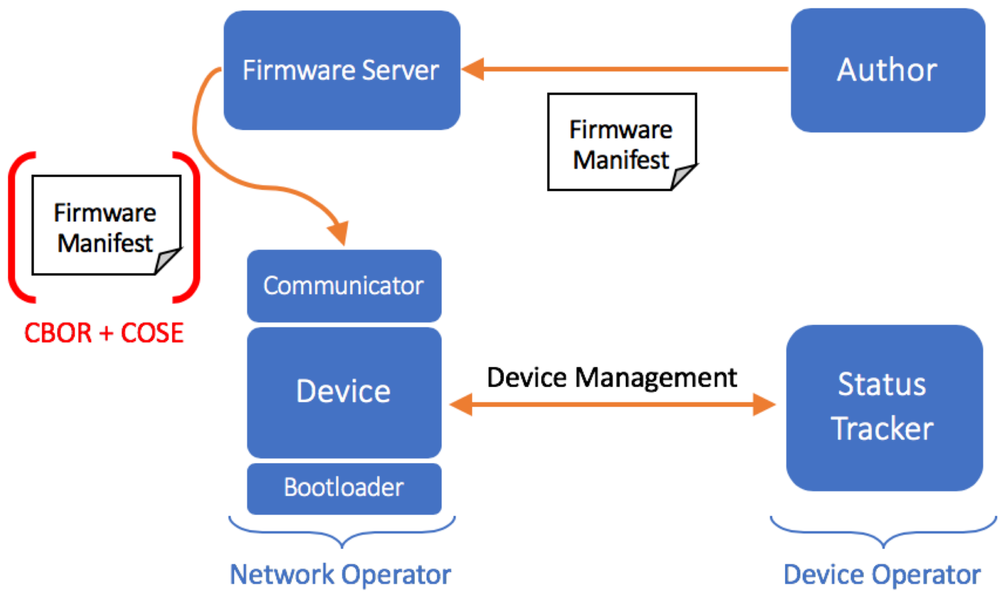
SOFTWARE UPDATES FOR IOT

Concept

The SUIT working group is chartered to develop firmware update solutions that can be implemented into Internet of Things (IoT) devices; especially those with limited RAM and flash memory, such as **~10 KiB RAM** and **~100 KiB flash.**



Background

Recent attacks on IoT devices have taken advantage of poor device configuration (i.e. the Mirai botnet generated a 600Gbps DDOS using CCTV cameras). It has also been reported that software updates have effectively bricked devices without user consent (e.g. Nest); similarly the lack of firmware updates has caused broken APIs (e.g. Samsung Smart Fridge).

There is no modern interoperable approach allowing secure updates to firmware in IoT devices. Work on **RFC8240** provides a summary of the state of the art.

SUIT defines a *manifest form* to specify what the firmware images contain. The format is currently defined at: **tools.ietf.org/html/draft-moran-suit-manifest**

```
{
    "manifestFormatVersion" : 1,
    "nonce" : "c31211d1ff88",
    "conditions" : {
        "vendorId" : "fa6b4a53-d5ad-5fdf-be9d-e663e4d41ffe",
        "classId" : "1492af14-2569-5e48-bf42-9b2d51f2ab45"
    },
    "payloadInfo" : {
        "format" : {
            "type" : "binary"
        },
        "size" : 16,
        "digestAlgorithm" : "SHA-256",
        "digests" : {
         "raw" : "c31211d1ff88f77a5aaf653677895bfca769f06da198a8fa71156aa64acd695d",
         "ciphertext" : "f7e59db5d5ef2b6bbb732dec2e8ef33c285224cf7bad235910e402b5f5249c22"
        }
    }
}
```

suitmanifest.json

It contains the version of the manifest format, a manifest description, the payload description, the vendor and the model names. Not present in the example it also may have directives, dependencies and extensions.

The payload information contains the format, the size, the storage identifier a message digest as well the digest algorithm used; SHA-256 in this case.

The payload format serialized in CBOR and signed in COSE; the assumption is that the Firmware Server is capable of signing and encoding the manifest with its private key.

```
00000000: 8a01 f646 c312 11d1 ff88 1a5b 4a7a be82  ...F.......[Jz..
00000010: 8201 50fa 6b4a 53d5 ad5f dfbe 9de6 63e4  ..P.kJS.._....c.
00000020: d41f fe82 0250 1492 af14 2569 5e48 bf42  .....P....%i^H.B
00000030: 9b2d 51f2 ab45 f6f6 f6f6 8781 0110 f6f6  .-Q..E..........
00000040: 8101 a201 5820 c312 11d1 ff88 f77a 5aaf  ....X .......zZ.
00000050: 6536 7789 5bfc a769 f06d a198 a8fa 7115  e6w.[..i.m....q.
00000060: 6aa6 4acd 695d 0358 20f7 e59d b5d5 ef2b  j.J.i].X ......+
00000070: 6bbb 732d ec2e 8ef3 3c28 5224 cf7b ad23  k.s-....<(R$.{.#
00000080: 5910 e402 b5f5 249c 22f6                 Y.....$.".
```

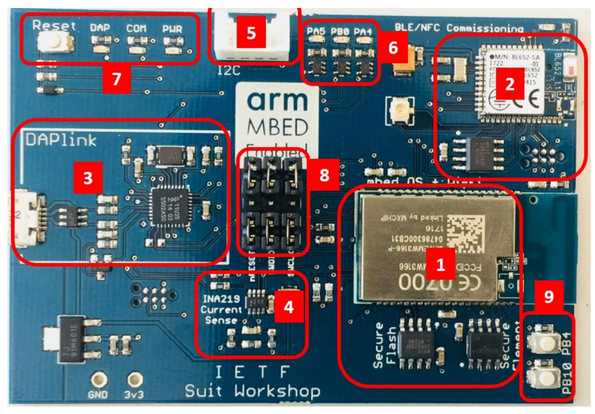
suitmanifest.cbor

Implementation

A basic SUIT implementation requires the capability to:

- [x] Generate a Manifest (JSON)
- [x] Encode it in Constrained Binary Representation format (CBOR).
- [x] Sign it with COSE (RFC8252).
- [x] Send it to the device.
- [ ] Verify the Manifest on the device.
- [x] Reboot and flash.

The hardware has been customized by ARM for the Software Updates work. It features an Atmel Secure element and connectivity over Wifi and BLE. Leds are used for visual feedback.



1. Wi-Fi Subsystem
   - MXChip EMW3166 WiFi Module
   - MX25R3235 32Mbit SPI flash
   - Atmel ATECC608A Secure element
2. BLE/NFC Commissioing subsystem
   - Laird BL652 BLE module
   - MX25R3235 32Mbit SPI flash
3. DAPlink interface (LPC11U35)
4. INA219 I2C Current sensor
5. Grove connector for I2C interface
6. G,R,B LEDs on EMW3166 IO
7. DAPLink reset and status LEDs
8. Jumpers to configure DAPlink target
9. Push buttons on EMW3166 (PB10, PB4)

Mbed supports it on arm PELION they also also provide an online compiler to implement the device logic: **https://os.mbed.com/compiler/**

It is possible to run locally the **ARM**mbed CLI, to facilitate the work there is a docker image available:

```
~$ docker pull jaime/mbed-cli
~$ docker run -it --entrypoint=/bin/bash jaim/mbed-cli:latest
```

It is also possible to run locally the RIOT environment on docker:

```
~$ docker pull riot/riotbuild
~$ git clone https://github.com/RIOT-OS/RIOT.gitocker
~$ make BUILD_IN_DOCKER=1 -C examples/hello-world/ ~$ BOARD=frdm-k64f
```

A basic update module manages firmware transfer over **CoAP**, verification and storage is done in Flash within the non-running image slot.

Conclusion

This work provides a standard and interoperable way to so Firmware updates. It is currently being standardized and prototyped on various IoT OSs. Now it is a good time to prepare IoT solutions for the coming of this technology.

References

http://git.io/fxh6D

Jaime Jiménez {jaime.jimenez@ericsson.com}