New Zephyr features
LWM2M / FOTA Framework

Marti Bolivar <marti.bolivar@linaro.org>
David Brown <david.brown@linaro.org>
Ricardo Salveti <ricardo.salveti@linaro.org>
Mike Scott <michael.scott@linaro.org>

LEADING
COLLABORATION
IN THE ARM
ECOSYSTEM

Zephyr and LTD
“changing at an alarming pace”

Since Zephyr v1.0.0 (February 8)

Version	Changesets	Changed lines
1.1.0	481	15,798(+) 9,205(-)
1.2.0	482	37,176(+) 11,044(-)
1.3.0	456	37,216(+) 15,452(-)
1.4.0	473	289,223(+) 14,585(-)
1.5.0	970	179,485(+) 18,732(-)
1.6.0	3,188	1,706,583(+) 131,887(-)
1.7.0	1,839	1,126,493(+) 336,555(-)
1.8.0	1,692	858,467(+) 239,250(-)

FOTA at last Connect

- MCUBoot talk given by David
  - Introduce the bootloader
  - Discuss the Zephyr port, work with Runtime, etc
  - A more detailed update was given earlier today in SFO17-118, so just some Zep updates to discuss today
- hawkBit keynote demo
  - FOTA updates of devices which live-streamed temperature data
  - An updated demo was given at today's keynote

Zephyr work since then

- Firmware over the air (FOTA) framework
- LWM2M subsystem and example applications
- Today's keynote demos
  - hawkBit
  - LWM2M
  - BT Mesh lights

Work in progress, in collaboration with other groups and companies.

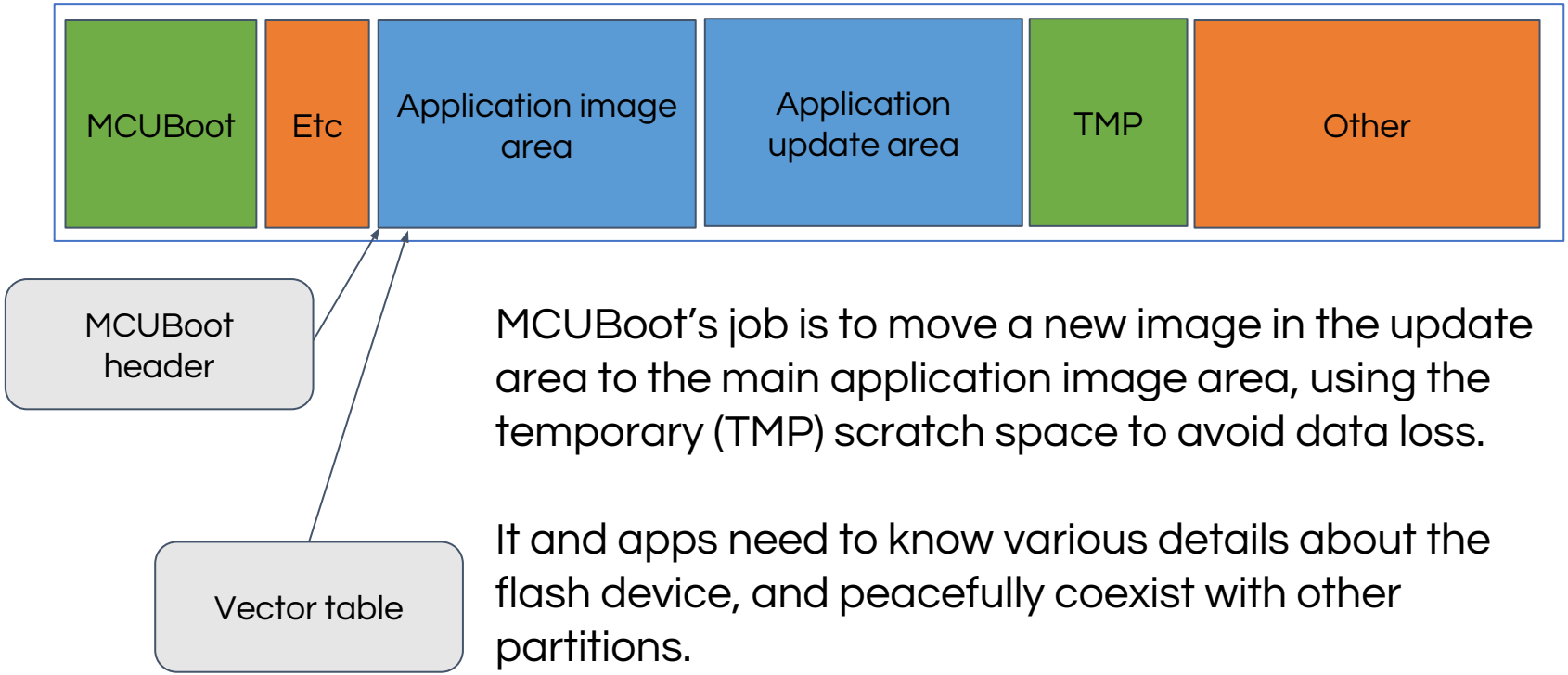
MCUBoot + FOTA framework

Desiderata

- Should be possible to add SoC or board support for mcuboot without changing mcuboot repository
- Should be possible to portably share system configuration
  - The bootloader,
  - Applications which are chain-loaded by it
  - Bootloader and application build systems (including flashing and debug)

In short, mcuboot should be “easy”.

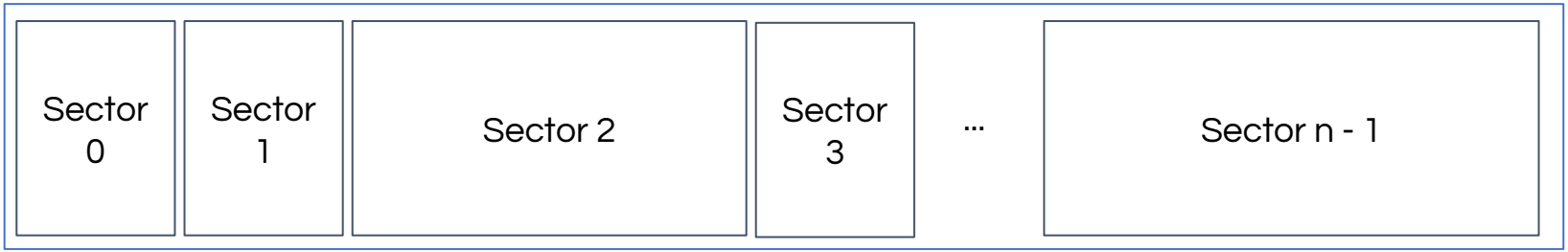
Booting with MCUBoot



MCUBoot's job is to move a new image in the update area to the main application image area, using the temporary (TMP) scratch space to avoid data loss.

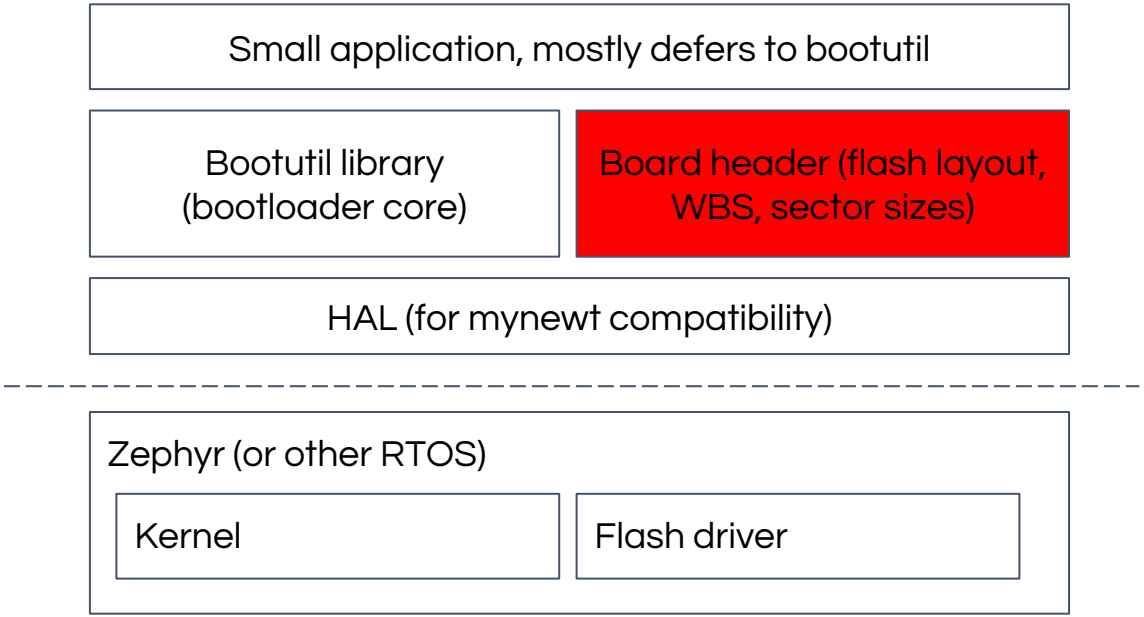
It and apps need to know various details about the flash device, and peacefully coexist with other partitions.

Flash specifics to know about (apps and bootloader)



- Map between sectors and “partitions” / image areas
- Nonuniform sector sizes (e.g. STM32)
- Widely varying “write block sizes” (1, 4, 8 common)

Support issues on Zephyr



One header file per board with flash layout, sector “size”, etc.

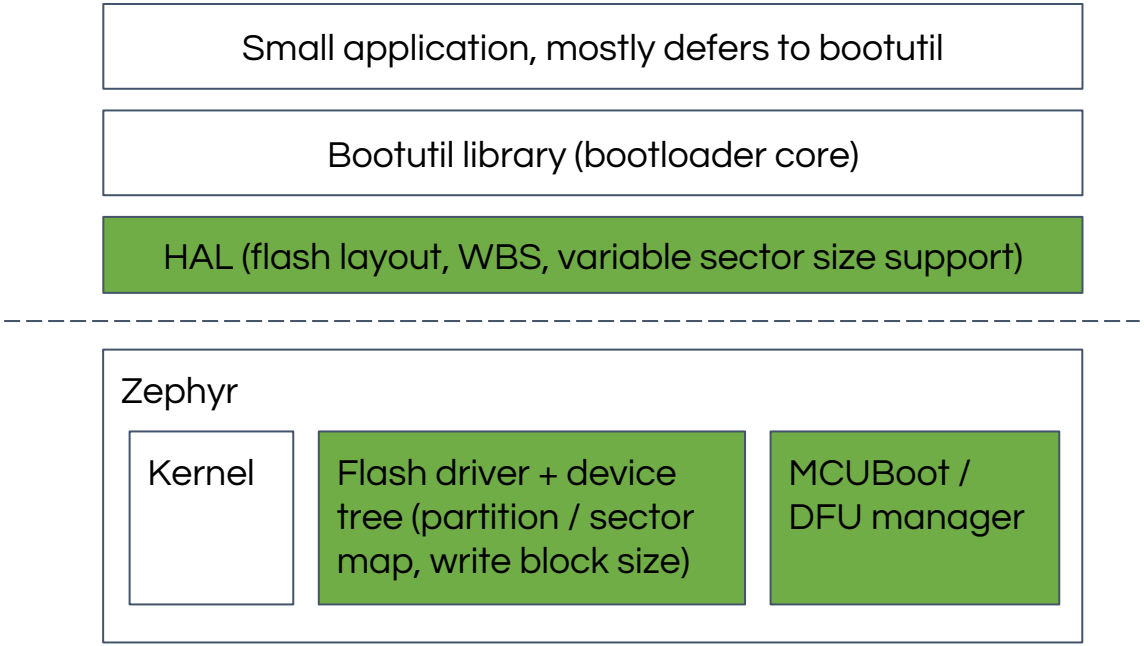
This information is duplicated again in every application that needs to manage FOTA updates with mcuboot.

This is tedious and error-prone. Also, what happens when you want to change the config?

Partitions in DTS now, but more remains for writing apps.

Other issues around flashing, etc.

MCUBoot with Zephyr + “FOTA framework”



MCUBoot’s HAL will allow target-specific hacks, but supports obtaining everything from the OS.

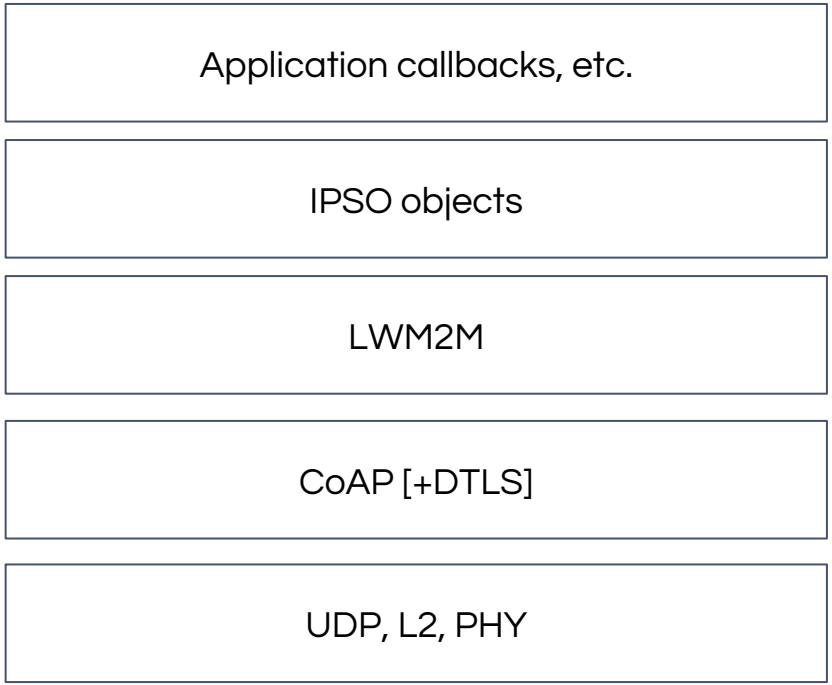
Target-specific behavior from DTS and the flash driver.

MCUBoot manager allows apps and bootloader to start sharing code.

Makefile.exports for build systems, but more work needed to make this easy.

LWM2M

LWM2M from 10,000 ft (3048 m)



LWM2M and IPSO

- LWM2M: 18 base objects managed via OMA Working Groups with room for expansion up to object id 1023. Security (0), Server (1), Device (3), Firmware (5) ...
- IPSO: Hundreds of standard and vendor-defined objects starting with object id 2048. Examples: Light control (3311) used in keynote demo, also presence (3302), humidity (3304), accelerometer (3313), magnetometer (3314), …
- These well-defined objects / resources increase interoperability.

More details in the OMA registry.

IPSO Light Control Object Resources

Resource	Name	Read/write	Number	Required?	Type
5850	On/Off	RW	Single	Mandatory	Boolean
5851	Dimmer	RW	Single	Optional	Integer
5852	On Time	RW	Single	Optional	Integer
5805	Cumulative active power	R	Single	Optional	
5820	Power Factor	R	Single	Optional	Float
5706	Colour	RW	Single	Optional	String
5701	Sensor Units	R	Single	Optional	String

IPSO Light Control in Source Form

```
static struct lwm2m_engine_obj_field fields[] = {
        OBJ_FIELD_DATA(LIGHT_ON_OFF_ID, RW, BOOL),
        OBJ_FIELD_DATA(LIGHT_DIMMER_ID, RW, U8),
        OBJ_FIELD_DATA(LIGHT_ON_TIME_ID, RW, S32),
        OBJ_FIELD_DATA(LIGHT_CUMULATIVE_ACTIVE_POWER_ID, R,
                       FLOAT32),
        OBJ_FIELD_DATA(LIGHT_POWER_FACTOR_ID, R, FLOAT32),
        OBJ_FIELD_DATA(LIGHT_COLOUR_ID, RW, STRING),
        OBJ_FIELD_DATA(LIGHT_SENSOR_UNITS_ID, R, STRING),
};
```

Instance-specific data declarations

```
/* Initialize instance resource data. */
INIT_OBJ_RES_DATA(res[avail], i, LIGHT_ON_OFF_ID,
        &on_off_value[avail], sizeof(*on_off_value));
INIT_OBJ_RES_DATA(res[avail], i, LIGHT_DIMMER_ID,
        &dimmer_value[avail], sizeof(*dimmer_value));
  [...]
INIT_OBJ_RES_DATA(res[avail], i, LIGHT_SENSOR_UNITS_ID,
        units[avail], LIGHT_STRING_SHORT);
```

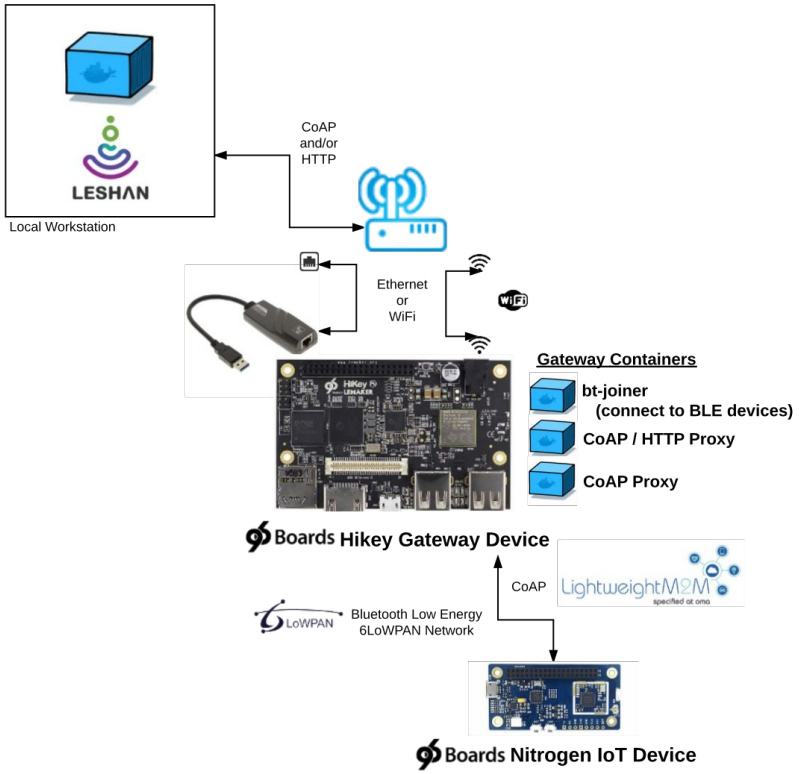
Application level

```
/* Resource initialization */
    lwm2m_engine_set_string("3/0/0", CLIENT_MANUFACTURER);
    lwm2m_engine_set_string("3/0/1", CLIENT_MODEL_NUMBER);
   [...]

/* Engine startup */
    ret = lwm2m_rd_client_start(&client, CONFIG_NET_APP_PEER_IPV6_ADDR,
                                CONFIG_LWM2M_PEER_PORT, CONFIG_BOARD,
                                rd_client_event);
```

End to End Demo System

- Control an LED and do a FOTA update via LWM2M
- Canned Docker containers and build system glue to get moving quickly
- http://ltd-docs.readthedocs.io/en/ltd-17.09/iotfoundry/lwm2m-howto.html



How to play with the Zephyr LWM2M Client locally

- Download the latest Leshan Demo Server build and start
  - $ wget https://hudson.eclipse.org/leshan/job/leshan/lastSuccessfulBuild/artifact/leshan-server-demo.jar
  - $ java -jar ./leshan-server-demo.jar
  - Open a web browser to: http://localhost:8080
- Run the Zephyr LWM2M Client via QEMU
  - Follow "Getting Started" Guide
  - Follow "Networking with QEMU" Docs
  - $ make -C samples/net/lwm2m_client run

Zephyr LWM2M Roadmap

In Zephyr v1.9

- LWM2M Engine to manage registering smart object and handling read/write/create/delete operations.
- Registration device client state machine (DTLS not supported yet)
- 4 OMA LWM2M objects: Security, Server, Device and Firmware
- Support for firmware update via both direct resource write and pull via URL resource.

Coming in the future

- Migrate to new Zephyr CoAP APIs.
- LWM2M Engine to support DTLS.
- Registration client to support bootstrap state machine.
- Add more OMA standard smart objects: Access Controls, Location …
- Add more IPSO Smart Objects.
- Optimization!