Firmware Updates from RIOT to RIOT

a practical guide

Benjamin Valentin

ML!PA Consulting GmbH

September 5, 2022

Use case

- Gateway device which is not always online, collects data from sensor network
- When uplink is available, sends out collected data, downloads updates for all nodes
  - needs to cache data & update files
  - many nodes of the same type

OR

- Offline updates: Network is not online at all, updates served via SD card

What do we need

- everything needed available in RIOT:
  - VFS layer
    can store files on and fs on and MTD device
  - GCoAP filesever
    can serve files from VFS via CoAP (like `aiocoap-fileserver`)
  - SUIT updates
    can fetch updates from CoAP and VFS sources

Mounting a filesystem

- Pseudo-Module `vfs_default` selects default file system(s) defined by board (same as `netdev_default`)
- Board defines mount points via `VFS_AUTO_MOUNT()` XFA macro
- Mount point naming convention:
  - flash devices get mounted to `/nvm0`, `/nvm1`, ...
  - SD cards get mounted to `/sd0`, `/sd1`, ...
  - `VFS_DEFAULT_DATA` can be used to write portable code
- Pseudo-Module `vfs_auto_format` formats device if mount fails

Mounting a filesystem

littlefs2 on NOR flash

```
/* configure MTD device */
static mtd_spi_nor_t spi_nor_dev = {
    .base = {
        .driver = &mtd_spi_nor_driver,
        .page_size = 256,
        .pages_per_sector = 16,
    },
    .params = &_spi_nor_params,
};

/* configure mount point */
VFS_AUTO_MOUNT(littlefs2, VFS_MTD(spi_nor_dev), VFS_DEFAULT_NVM(0), 0);
                  ↑                ↑                ↑            ↑
            use littlefs2     get mtd_t    mount to /nvm0   1st mountpoint
```

Mounting a filesystem

FAT on SD card

```
/* configure MTD device */
static mtd_sam0_sdhc_t sdhc_dev = {
    .base = {
        .driver = &mtd_sam0_sdhc_driver,
    },
    .state = {
        .dev = SDHC1,
        .cd  = GPIO_PIN(PD, 20),
        .wp  = GPIO_PIN(PD, 21),
    },
};

/* configure mount point */
VFS_AUTO_MOUNT(fatfs, VFS_MTD(sdhc_dev), VFS_DEFAULT_SD(0), 1);
                 ↑                ↑           ↑              ↑
             use FAT        get mtd_t     mount to /sd0   2nd mountpoint
```

Mounting a filesystem

- MTD device is SPI NOR
- default fs is littlefs2

Makefile.dep

```
# default to using littlefs2 on the external flash
ifneq (,$(filter vfs_default,$(USEMODULE)))
  USEPKG += littlefs2
  USEMODULE += mtd
endif

# define default MTD devices
ifneq (,$(filter mtd,$(USEMODULE)))
  USEMODULE += mtd_spi_nor
endif
```

Mounting a filesystem

List mountpoints

```
> vfs df
Mountpoint                 Total          Used     Available      Capacity
/sd0                     1948544         16239       1932305         0%
/nvm0                       8192            55          8137         0%
```

List files

```
> ls /nvm0
./
../
fdb_kvdb1/
fdb_tsdb1/
fw/
image.bin    57344 B
old_image.bin    52224 B
range/
total 2 files
```

Serving updates with GCoAP

- GCoAP fileserver can be registered as handler for a CoAP endpoint
- takes path to directory as argument
- (optional) write support (PUT, DELETE)

Provide `/sd0/fw` as `/fw`

```
static const coap_resource_t _resources[] = {
    { "/fw", COAP_GET | COAP_MATCH_SUBTREE,
             gcoap_fileserver_handler, "/sd0/fw" },
    /* app specific endpoints */
    { "/cmd", COAP_GET | COAP_PUT, _coap_cmd_handler, NULL },
    { "/log", COAP_PUT, _coap_log_handler, NULL },
};
static gcoap_listener_t _listener = {
    .resources = _resources,
    .resources_len = ARRAY_SIZE(_resources),
};
gcoap_register_listener(&_listener);
```

Download Firmware Update from GCoAP fileserver

Using SUIT worker thread

```
char url[CONFIG_SOCK_URLPATH_MAXLEN];
int len = snprintf(url, sizeof(url),
                   "coap://[fdea:dbee:f::1]/fw/%s.suit", RIOT_BOARD);

suit_worker_run();
suit_worker_trigger(url, len);
```

Download files from CoAP server to VFS

- `nanocoap_vfs_get_url()` / `nanocoap_vfs_get()`: Download file from CoAP
- `nanocoap_vfs_put_url()` / `nanocoap_vfs_put()`: Upload file to CoAP
- `nanocoap_link_format_get_url()` / `nanocoap_link_format_get()`: iterate CoRE Link Format directory listing [1]
  - mirror directory
- `ncget` / `ncput` shell commands

[1] pending review

Install Firmware from SD card

- Load an update from SD card instead of CoAP server
- Enable the `suit_transport_vfs` module
- Set `SUIT_COAP_ROOT = file:///sd0/fw` in the app `Makefile`

Using SUIT worker thread

```
char url[CONFIG_SOCK_URLPATH_MAXLEN];
int len = snprintf(url, sizeof(url),
                   "file:///sd0/fw/%s.suit", RIOT_BOARD);

suit_worker_run();
suit_worker_trigger(url, len);
```

Other new SUIT features

- keys now stored in `$(XDG_DATA_HOME)/RIOT/keys`

Encrypted Private key

- set `SUIT_SEC_PASSWORD` when creating and using the key
- use `password_protect_key.py` to encrypt existing key

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIGbMFcGCSqGSIb3DQEFDTBKMCkGCSqGSIb3DQEFDDAcBAj0LvHymDZtUAICCAAw
DAYIKoZIhvcNAgkFADAdBglghkgBZQMEASoEEBLwoScnJ/I1AygLzR6VDc8EQBjj
xCoHzJB2/lSif2A85qj1LsEpmiFvqceE6PPZFakWYx/86uNbnUabmETbo6DyPt/n
hRQhOOmKUvpx9gnHaek=
-----END ENCRYPTED PRIVATE KEY-----
```

Multiple Public Keys [a]

[a] pending review

- development firmware accepts both dev key and production key
- `SUIT_SECS="devel.pem production.pem"`

Demo