Installation de Mongoose OS et d'une application de démonstration

Nous développerons en C directement en ligne de commande, ce qui nous permettra d'économiser la place prise par l'interprète Javascript de la version « développement Web » de Mongoose OS.

Droits d'accès au device

```
xterm
$ ls -la /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 Nov 27 10:56 /dev/ttyUSB0
$ sudo usermod -aG dialout $USER
```

Vous ajouterez le groupe auquel appartient le `/dev/ttyUSBx` à votre utilisateur.

Site de l'OS : `https://mongoose-os.com`

```
xterm
$ sudo add-apt-repository ppa:mongoose-os/mos
$ sudo apt update
$ sudo apt install mos
$ mos --help
```

Installation de **docker** avec transfert des droits d'exécution à l'utilisateur :

```
xterm
$ sudo apt install docker.io
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

Pour que votre entrée dans le groupe `docker` soit prise en compte, il vous faut vous déconnecter/reconnecter.

ATTENTION

L'installation de docker s'accompagne de la reconfiguration de votre firewall.
Faites attention que les « *policies* » en `FORWARD` laisse bien passer votre trafic en provenance de votre Raspberry Pi.

```
xterm
$ sudo iptables -t filter -P FORWARD ACCEPT
```

Compilation d'un firmware

Installation d'une application de **démonstration** :

xterm
```
$ git clone https://github.com/mongoose-os-apps/empty my-app
```

Vous éditerez le manifeste de l'application de démonstration (fichier « `mos.yml` »):

```
author: mongoose-os
description: A Mongoose OS app skeleton
version: 1.0

libs_version: ${mos.version}
modules_version: ${mos.version}
mongoose_os_version: ${mos.version}

# Optional. List of tags for online search.
tags:
  - c

# List of files / directories with C sources. No slashes at the end of dir names.
sources:
  - src

# List of dirs. Files from these dirs will be copied to the device filesystem
filesystem:
  - fs

build_vars:
  MGOS_MBEDTLS_ENABLE_ATCA: 1

config_schema:
  - ["debug.level", 3]
  - ["sys.atca.enable", "b", true, {title: "Enable the chip"}]
  - ["i2c.enable", "b", true, {title: "Enable I2C"}]
  - ["sys.atca.i2c_addr", "i", 0x60, {title: "I2C address of the chip"}]
  - ["wifi.ap.enable", "b", false, {title: "Enable"}]
  - ["wifi.sta.enable", "b", true, {title: "Connect to existing WiFi"}]
  - ["wifi.sta.ssid", "s", "IoT", {title: "SSID"}]
  - ["wifi.sta.pass", "s", "12344321", {title: "Password", type: "password"}]
  - ["http.enable", true]
  - ["http.listen_addr", ":443"]
  - ["http.ssl_cert", "ecc.crt.pem"]
  - ["http.ssl_key", "ATCA:0"]
libs:
  - origin: https://github.com/mongoose-os-libs/boards
  - origin: https://github.com/mongoose-os-libs/ca-bundle
  - origin: https://github.com/mongoose-os-libs/rpc-service-config
  - origin: https://github.com/mongoose-os-libs/rpc-service-fs
  - origin: https://github.com/mongoose-os-libs/rpc-uart
  - origin: https://github.com/mongoose-os-libs/atca
  - origin: https://github.com/mongoose-os-libs/wifi
# Used by the mos tool to catch mos binaries incompatible with this file format
manifest_version: 2017-09-29
```

Ici, on demande à l'ESP8266 de:

- ▷ *activer le composant ATECC608 ;*
- ▷ *se connecter au point d'accès SSID: IoT, PWD: 1234321 ;*
- ▷ *activer un serveur http protégé par TLS utilisant un certificat basé ECC dont la clé privée est gérée par le composant ATECC608 ;*

***Compilation** de l'application de démonstration :*

xterm
```
$ cd my-app
$ mos build --local --platform esp8266
$ mos flash
$ mos console
```

L'utilisation de l'option `--local` permet d'installer un **container** pour disposer du compilateur et des bibliothèques nécessaires à la compilation de Mongoose OS (dans le cas contraire votre application est compilée dans le Cloud...)⟹cela peut prendre du temps lors de la première compilation.

*Pour **automatiser la procédure** de compilation et de flashage :*

xterm
```
mos build --local --platform esp8266 && mos flash && mos console
```