Faculté des Sciences & Techniques — Université de Limoges

Master 2nde année

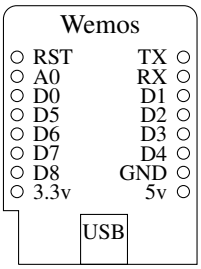
TMC

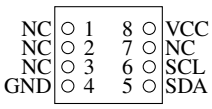
TP n°4

MQTT avec connexion sécurisée par authentification du client

Interconnexion ESP8266/ATECC608A

On utilisera le bus I²C, c-à-d 3 fils : GND, SCL et SDA.





ESP8266	ATECC508A
3.3v	VCC
GND	GND
D5	SCL
D6	SDA

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_latest
$ 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-service-atca
  - 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
  - origin: https://github.com/mongoose-os-libs/http-server
# 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
```

Sur le Raspberry Pi, où vous aurez installé le `script_ap`*:*

xterm

```
pi@raspberrypi:~ $ ./script_ap
hostapd: no process found
net.ipv4.ip_forward = 1
Configuration file: /tmp/hostapd_config
Using interface wlan0 with hwaddr b8:27:eb:8d:bc:f1 and ssid "IoT"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
dnsmasq: started, version 2.85 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6
no-Lua TFTP conntrack ipset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 10.33.33.100 -- 10.33.33.150, lease time 1h
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: read /etc/hosts - 5 addresses
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: associated
wlan0: AP-STA-CONNECTED ec:fa:bc:5e:fa:cd
wlan0: STA ec:fa:bc:5e:fa:cd RADIUS: starting accounting session D056AB159699B069
wlan0: STA ec:fa:bc:5e:fa:cd WPA: pairwise key handshake completed (RSN)
dnsmasq-dhcp: DHCPDISCOVER(wlan0) ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPOFFER(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPDISCOVER(wlan0) ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPOFFER(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPREQUEST(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPACK(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd esp8266_5EFACD
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: disassociated
wlan0: AP-STA-DISCONNECTED ec:fa:bc:5e:fa:cd
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: disassociated
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: disassociated
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: disassociated
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: disassociated
wlan0: STA ec:fa:bc:5e:fa:cd IEEE 802.11: associated
wlan0: AP-STA-CONNECTED ec:fa:bc:5e:fa:cd
wlan0: STA ec:fa:bc:5e:fa:cd RADIUS: starting accounting session 03AF9DBE0F7F5D0C
wlan0: STA ec:fa:bc:5e:fa:cd WPA: pairwise key handshake completed (RSN)
dnsmasq-dhcp: DHCPDISCOVER(wlan0) ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPOFFER(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPREQUEST(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd
dnsmasq-dhcp: DHCPACK(wlan0) 10.33.33.103 ec:fa:bc:5e:fa:cd esp8266_5EFACD
```

Configuration du composant ATECC608A : installation de la clé privée

`https://mongoose-os.com/docs/mongoose-os/userguide/security.md#atecc608a-crypto-chip`

L'activation et la configuration du composant ATECC608A est déja ***faite*** *et* ***bloquée*** *sur les composants distribués.*

Elle autorise :

- ▷ la répartition de la mémoire flash du composant entre les différents stockages, « *slots* » et usages : *Slots 0-3 are ECC slots with ECDH enabled. They can be generated on the device or rewritten using key in slot 4, which itself can be reset at any time.*
- ▷ l'installation **à volonté** de la clé privée associée à un certificat pour réaliser signature et authentification ; *Cette possibilité est nécessaire dans le cas d'un composant utilisé à des fins pédagogiques...*

Il faut juste une **clé d'installation** qui permet « *d'ouvrir* » l'ATECC608 afin d'y installer la clé privée associée à un certificat.

Pour la création et l'installation de la **clé d'installation** dans l'ATECC508 :

xterm

```
$ openssl rand -hex 32 > slot4.key
$ mos -X atca-set-key 4 slot4.key --dry-run=false
Using port /dev/ttyUSB0

ATECC608A rev 0x6002 S/N 0x0123d4df07023967ee, config is locked, data is locked

Slot 0 is a ECC private key slot
Parsed EC PRIVATE KEY
Data zone is locked, will perform encrypted write using slot 4 using slot4.key
Writing block 0...
SetKey successful.
```

Création et Installation du certificat associé au serveur Web ainsi que de sa clé privée dans l'ATECC608

Pour la création de la clé privée ECC :

xterm
```
$ openssl ecparam -out ecc.key.pem -name prime256v1 -genkey
```

Pour l'installation de la clé privée ECC dans l'ATECC grâce à la clé d'installation :

xterm
```
$ mos -X atca-set-key 0 ecc.key.pem --write-key=slot4.key --dry-run=false
```

Pour la création d'un certificat « *auto-signé* » pour le serveur Web :

xterm
```
$ openssl req -new -subj  "/C=FR/L=Limoges/O=TMC/OU=IOT/CN=mqtt.tmc.com"  -sha256
-key ecc.key.pem -text -out ecc.csr.tmpl
$ openssl x509 -in ecc.csr.tmpl -text -out ecc.crt.pem  -req -signkey ecc.key.pem
-days 3650
```

On utilise la clé installée dans le « slot 4 » pour autoriser l'installation de cette clé dans le composant.

Vous copierez le fichier contenant le certificat « `ecc.crt.pem` » dans le sous-répertoire « `fs` » de votre application et la recompiler/reflasher.

Vous pouvez également installer le certificat dans le « *filesystem* » de l'ESP8266 sans recompilation :

xterm
```
$ mos put ecc.crt.pem
$ mos ls
Using port /dev/ttyUSB0
conf0.json
ecc.crt.pem
index.html
```

Test de la connexion sécurisée avec authentification au travers de l'ATECC608

xterm

```
$ openssl s_client -connect 10.33.33.146:443
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com
verify error:num=18:self signed certificate
verify return:1
depth=0 C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com
verify return:1
---
Certificate chain
 0 s:C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com
   i:C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIBlDCCAToCCQCNEPoguhTQgzAKBggqhkjOPQQDAjBSMQswCQYDVQQGEwJGUjEQ
MA4GA1UEBwwHTGltb2dlczEMMAoGA1UECgwDVE1DMQwwCgYDVQQLDANJT1QxFTAT
BgNVBAMMDG1xdHQudG1jLmNvbTAeFw0xNzEyMDIxOTEwNTRaFw0yNzExMzAxOTEw
NTRaMFIxCzAJBgNVBAYTAkZSMRAwDgYDVQQHDAdMaW1vZ2VzMQwwCgYDVQQKDANU
TUMxDDAKBgNVBAsMA0lPVDEVMBMGA1UEAwwMbXF0dC50bWMuY29tMFkwEwYHKoZI
zj0CAQYIKoZIzj0DAQcDQgAEUMuSZezP+mCX2nffWvObSazns/d7lTfoR0HnihgR
75gA7UXpiNrRlMH8tO8Y4ntDa9APwk0r4anChvjLyXoGhzAKBggqhkjOPQQDAgNI
ADBFAiEAsIes4Bg6V/Yke9A/VhIJI9e6nnCxSdzi6S+jQy7IiacCIBGnifB1OXvM
NUu0xo+w8ldpmfz+AY6/vK7Yc2JGiQoM
-----END CERTIFICATE-----
subject=C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com

issuer=C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com

---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: ECDSA
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 732 bytes and written 409 bytes
Verification error: self signed certificate
---
New, TLSv1.2, Cipher is ECDHE-ECDSA-AES128-GCM-SHA256
Server public key is 256 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-ECDSA-AES128-GCM-SHA256
    Session-ID: 1003B68B0638F14F5535F6D8EDD82D443DF3879DBF7B2B36BE84C94AAF5CD7E6
    Session-ID-ctx:
    Master-Key:
44F44612D9745AB4D50E616348C5F7D3FBD10479F4CE959226B18BAA74CFADF3F75A848000C904DF74F44A24464F71D8
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1606474929
    Timeout   : 7200 (sec)
    Verify return code: 18 (self signed certificate)
    Extended master secret: yes
---
```

Le contenu du certificat installé dans l'ATECC608A (qui correspond à celui reçu lors de la connexion) :

xterm

```
$ openssl x509 -in ecc.crt.pem -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            8d:10:fa:20:ba:14:d0:83
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com
        Validity
            Not Before: Dec  2 19:10:54 2017 GMT
            Not After : Nov 30 19:10:54 2027 GMT
        Subject: C = FR, L = Limoges, O = TMC, OU = IOT, CN = mqtt.tmc.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:50:cb:92:65:ec:cf:fa:60:97:da:77:df:5a:f3:
                    9b:49:ac:e7:b3:f7:7b:95:37:e8:47:41:e7:8a:18:
                    11:ef:98:00:ed:45:e9:88:da:d1:94:c1:fc:b4:ef:
                    18:e2:7b:43:6b:d0:0f:c2:4d:2b:e1:a9:c2:86:f8:
                    cb:c9:7a:06:87
                ASN1 OID: prime256v1
                NIST CURVE: P-256
    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:21:00:b0:87:ac:e0:18:3a:57:f6:24:7b:d0:3f:56:
         12:09:23:d7:ba:9e:70:b1:49:dc:e2:e9:2f:a3:43:2e:c8:89:
         a7:02:20:11:a7:89:f0:75:39:7b:cc:35:4b:b4:c6:8f:b0:f2:
         57:69:99:fc:fe:01:8e:bf:bc:ae:d8:73:62:46:89:0a:0c
```

Meanwhile on the ESP8266...

xterm

```
$ mos console
Using port /dev/ttyUSB0
[Nov 27 12:05:15.295] main.c:5                Tick uptime: 46.54, RAM: 53600, 43760 free
[Nov 27 12:05:16.295] main.c:5                Tock uptime: 47.54, RAM: 53600, 43760 free
[Nov 27 12:05:16.400] mg_lwip_net_if.c:373    0x3ffef094 conn 0x3fff15dc from
10.33.33.254:55318
[Nov 27 12:05:16.406] mg_net.c:526            0x3ffef094 0x3fff134c 1073680756 0x10
[Nov 27 12:05:16.412] mg_net.c:532            0x3fff134c tcp://10.33.33.254:55318
[Nov 27 12:05:16.425] mg_ssl_if_mbedtls.c:30  0x3ffef094 ciphersuite:
TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
[Nov 27 12:05:16.485] ATCA:16 ECDH gen pubkey ok
[Nov 27 12:05:16.581] ATCA:0 ECDSA sign ok
[Nov 27 12:05:16.587] mgos_mongoose.c:66      New heap free LWM: 38080
[Nov 27 12:05:17.189] ATCA:16 ECDH ok
[Nov 27 12:05:17.203] mgos_http_server.c:180  0x3fff134c HTTP connection from
10.33.33.254:55318
[Nov 27 12:05:17.208] mgos_mongoose.c:66      New heap free LWM: 37400
[Nov 27 12:05:17.295] main.c:5                Tick uptime: 48.54, RAM: 53600, 41356 free
[Nov 27 12:05:18.295] main.c:5                Tock uptime: 49.54, RAM: 53600, 41364 free
[Nov 27 12:05:19.295] main.c:5                Tick uptime: 50.54, RAM: 53600, 41364 free
[Nov 27 12:05:20.046] mg_ssl_if_mbedtls.c:202 0x3fff134c TLS connection closed by peer
[Nov 27 12:05:20.296] main.c:5                Tock uptime: 51.54, RAM: 53600, 43568 free
[Nov 27 12:05:21.295] main.c:5                Tick uptime: 52.54, RAM: 53600, 43568 free
```

Pour réduire la quantité d'information de déboguage obtenue sur la console vous pouvez changer la valeur de 3 vers 2 dans le manifeste « `mos.yml` *» :*

```
  - ["debug.level", 3]
```