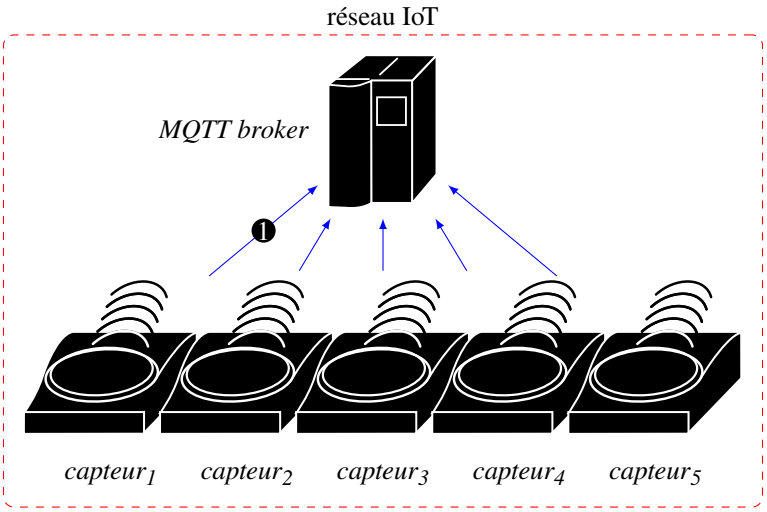
MQTT

IoT, capteurs et MQTT



Chaque capteur :

- ▷ dispose d'une **identité unique** et correspond à un ESP8266 ;
- ▷ devient un « *publisher* », ❶, par rapport au « *broker* » installé sur le Raspberry Pi.

Installation de Mongoose OS et d'une application de démonstration

MONGOOSE OS

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.

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

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

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

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

Compilation de l'application de démonstration :

xterm
```
$ cd my-app
$ mos build --arch esp8266
$ mos flash
$ mos console
$ mos wifi essid password
```

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

xterm
```
mos build --arch esp8266
mos flash; mos wifi essid password; mos console
```

Combinaison avec le Raspberry Pi

1. Vous installerez et configurerez sur le Raspberry Pi :
   - ▷ pour la connexion WiFi des ESP8266 :
     - ⋆ le point d'accès logiciel « `hostapd` » ;
       `https://raspberry-projects.com/pi/software_utilities/wifi-access-point`
     - ⋆ le serveur DHCP permettant de prendre en charge les clients WiFi connectés ;
     - ⋆ vous choissirez un SSID unique comme « *INSA123123* » et une PSK « *rnfrnfrnf* » que vous partagerez avec l'ESP8266 pour qu'il puisse s'y connecter.
   - ▷ Vous installerez le serveur MQTT « *mosquitto* » :

```
xterm
$ sudo apt install mosquitto mosquitto-clients
```

Pour activer la protection d'accès au serveur MQTT par mot de passe, vous ajouterez dans le fichier « `/etc/mosquitto/mosquitto.conf` » :

```
allow_anonymous false
password_file /etc/mosquitto/mosquitto_passwd
```

Vous utiliserez la commande « `mosquitto_passwd` » pour créer le contenu du fichier password.
`https://mosquitto.org/man/mosquitto_passwd-1.html`

```
xterm
$ sudo mosquitto_passwd -c /etc/mosquitto/mosquitto_passwd <user_name>
```

Pour s'abonner à un topic MQTT :

```
xterm
$ mosquitto_sub -h localhost -p 1883 -t mon_topic -u <user_name> -P
<password>
```

2. L'ESP8266 va se comporter comme un client MQTT en se connectant au serveur MQTT du Raspberry Pi jouant le rôle de concentrateur :
   - ◇ il va transmettre à intervalle régulier une mesure, en tant que « *publisher* » ;
     Ici, le message « Hello ! ».
   - ◇ sur le « *topic* » « `/esp8266` » ;
     Vous pourrez contrôler que cela fonctionne en vous connectant sur le serveur MQTT en tant que « subscriber » :

```
xterm
$ mosquitto_sub -h 10.20.30.1 -p 1883 -u toto -P secret -t '/esp8266'
Hello !
Hello !
```

   - ◇ une application exécutée sur le Raspberry Pi pourra récupérer les mesures une par une, par exemple en sortie de la commande « `mosquitto_sub` ».

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

config_schema:
  - ["debug.level", 3]
  - ["mqtt.enable", "b", true, {title: "Enable MQTT"}]
  - ["mqtt.server", "s", "iot.serveur.com:8883", {title: "MQTT server"}]
  - ["mqtt.pub", "s", "/esp8266", {title: "Publish topic"}]
  - ["mqtt.user", "s", "insa", {title: "User name"}]
  - ["mqtt.pass", "s", "rnfrnf", {title: "Password"}]

cdefs:
  MG_ENABLE_MQTT: 1

# List of libraries used by this app, in order of initialisation
libs:
  - 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-mqtt
  - origin: https://github.com/mongoose-os-libs/rpc-uart
  - origin: https://github.com/mongoose-os-libs/wifi

# Used by the mos tool to catch mos binaries incompatible with this file format
manifest_version: 2017-05-18
```

Le code source de l'application Mongoose OS :

```
#include <stdio.h>

#include "mgos.h"
#include "mgos_mqtt.h"

static void my_timer_cb(void *arg) {
  char *message = "Hello !";
  mgos_mqtt_pub("/esp8266", message, strlen(message), 1, 0);
  (void) arg;
}

enum mgos_app_init_result mgos_app_init(void) {
mgos_set_timer(2000, MGOS_TIMER_REPEAT, my_timer_cb, NULL);
  return MGOS_APP_INIT_SUCCESS;
}
```

Pour compiler, flasher l'ESP8266 :

xterm

```
$ mos build --local --arch esp8266
$ mos flash; mos wifi INSA123 rnfrnfrnf; mos console
```

Ici, le réseau WiFi est de SSID « `INSA123` » et le mot de passe « `rnfrnfrnf` ».