Louis Allain - Sujet IOT : Synchronisation d'un ensemble de feux de circulation

Ce document reprend le cheminement décrit dans le sujet avec à chaque fois une explication du principe de fonctionnement, une description du programme en pseudo-code et une réalisation en langage Lua. Chaque partie à été testée sur un ESP32 en simulant les autres feux via un client MQTT sur un ordinateur.

De plus, le programme complet reprenant l'ensemble des parties est donné en annexe.

Q1 - Synchronisation sans qu'un feu puisse "sortir" du système

Principe :

Pour synchroniser le fonctionnement des feux, chaque feu devra connaître le nombre total des feux en fonctionnement. Pour cela, lorsqu'un feu rentre dans le système, il se déclare en publiant dans un sujet MQTT. Ainsi, chaque feu recevant cette déclaration, incrémente le nombre de feux qu'il connaît.

De plus, à chaque fois qu'un feu reçoit la déclaration d'un nouveau feu, il publie également dans un autre sujet MQTT, le nombre de feu qu'il connaît. De ce fait à chaque fois qu'un feu reçoit un nombre de feux provenant des autres noeuds du système, celui-ci va regarder si le nombre de feux qu'il connaît est inférieur, et si c'est le cas alors il va remplacer le nombre de feu qu'il connaissait par le nombre de feux reçu. Ainsi, tous les nœuds du système vont posséder le bon nombre de feux au final.

Pour ce qui est de la synchronisation en elle-même, chaque feu va publier toutes les quatre secondes dans un sujet MQTT son identifiant unique. A la réception d'un tel message, chaque feu maintient un tableau d'identifiants représentant les feux qui sont prêts pour ce cycle. En plus, chaque feu va vérifier si la taille de ce tableau est égale au nombre de feux. Si c'est le cas alors chaque feu peut changer de couleur et peut réinitialiser le tableau des identifiants des feux prêts pour initialiser un nouveau cycle de quatre secondes.

Programme :

En pseudo-code :

On suppose l'existence des sujets MQTT suivants :

- compteurFeux
- nombreFeux
- feuxPret

Initialisation d'un feu :

- Variables :
  - Nombre de feux connu : "nb_feux" = 0
  - Nombre de feux prêts pour le cycle courant : "nb_feux_prêts" = 0
  - Tableaux des feux prêts "feux_prêts" : = { }
- Le feu s'abonne aux sujet "compteurFeux", "nombreFeux" et "feuxPrêt"
- Le feu publie dans le sujet "compteurFeux" le nombre 1

Méthode exécutée à la réception d'un message depuis le sujet "compteurFeux" :

- Ajoute à la variable "nb_feux" la valeur reçue
- Publie "nb_feux" sur le sujet "nombreFeux"

Méthode exécutée à la réception d'un message depuis le sujet "nombreFeux" :

- Si le nombre de feux que je connais est inférieur au nombre de feux que je reçois alors je mets à jour mon nombre de feux par la valeur reçue.

```
Si (nb_feux < nb_feux_reçu) Alors nb_feux = nb_feux_reçu Finsi
```

Méthode exécutée à la réception d'un message depuis le sujet "feuxPret" :

- Si l'uuid reçu du feu n'appartient pas au tableau "feux_prêts" alors incrémente la variable globale "nb_feux_prêts" de 1 et ajoute l'uuid reçu au tableau "feux_prêts". Si cette variable est égale à la variable globale "nb_feux" alors change la couleur du feu et réinitialise "nb_feux_prêts" et "feux_prêts".

```
Si (uuid_reçu ∉ feux_prêts) Alors
   nb_feux_prêts = nb_feux_prêts +1
   Ajoute uuid_reçu à feux_prêts
Finsi

Si (nb_feux_prêts == nb_feux) Alors
   Si(feu == VERT) Alors feu = rouge Sinon feu = vert Finsi.
   nb_feux_prêts = 0
Finsi
```

Boucle de synchronisation :

- Publie toutes les 4 secondes l'identifiant unique (uuid) du feu dans le sujet "feuxPret" pour indiquer qu'il est prêt à changer de couleur

En Lua :

- Initialisation :

```
pio.pin.setdir(pio.OUTPUT, pio.GPIO2) -- rouge
pio.pin.setdir(pio.OUTPUT, pio.GPIO4) -- vert

currentColor = "v" -- couleur du feu ("v" pour vert et "r" pour rouge)
nb_feux = 0 -- nombre de feu que l'on connaît
nb_feux_prets = 0 -- nombre de feu prets à changer de couleur pour ce cycle
feux_prets = {} -- tableaux des feux prêts pour ce cycle

TOPIC_COMPTEUR_FEUX = "/e1602246/compteurFeux"
TOPIC_NOMBRE_FEUX = "/e1602246/nombreFeux"
TOPIC_FEUX_PRET = "/e1602246/feuxPret"

-- Clients mqtt au broker
client = mqtt.client("e1602246-1", "mini.arsaniit.com", 1883, false)
clientSub = mqtt.client("e1602246-2", "mini.arsaniit.com", 1883, false)

-- uuid du feu
uuid = "feu1"

-- Booléen indiquant si la boucle doit tourner ou pas
running = false
```

- Définition de quelques fonctions d'aide :

```
-- Fonction se connectant à un réseau wifi
initSTA = function()
        net.wf.setup(net.wf.mode.STA, "SSID", "KEY")
        net.wf.start()
end

-- Fonction permettant de savoir si un élément fait parti d'un tableau
has_value = function(tab, val)
        for index, value in ipairs(tab) do
            if value == val then
                return true
            end
        end

        return false
end
```

```
-- Change de couleur, si rouge alors vert et vice versa
switchColor = function()
        if(currentColor == "v") then
                currentColor = "r"
                pio.pin.setlow(pio.GPIO4)
                pio.pin.sethigh(pio.GPIO2)
        else
                currentColor = "v"
                pio.pin.setlow(pio.GPIO2)
                pio.pin.sethigh(pio.GPIO4)
        end
end
```

- Fonctions “callback” des abonnements MQTT :

```
-- Méthode exécutée lors de la réception d'un message depuis
-- le sujet TOPIC_COMPTEUR_FEUX
onCompteurFeux = function(length, msg)
        -- modifie le nombre de feux connus
        nb_feux = nb_feux + msg
        -- publie sur le sujet TOPIC_NOMBRE_FEUX le nombre de feux connus
        clientSub:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)
end

-- Méthode exécutée lors de la réception d'un message depuis
-- le sujet TOPIC_NOMBRE_FEUX
onNombreFeux = function(length, msg)
        if(nb_feux < msg) then nb_feux = msg end
end

-- Méthode exécutée lors de la réception d'un message depuis
-- le sujet TOPIC_FEUX_PRET
onFeuxPret = function(length, msg)

        -- vérifie si c'est un nouveau feu qui est pret avant
        -- d'incrémenter le nombre de feux prêts
        if(has_value(feux_prets, msg) == false) then
                nb_feux_pret = nb_feux_pret + 1
                table.insert(feux_prets, msg)
        end

        if(nb_feux_pret == nb_feux) then
                switchColor()
                -- Réinitialisation
                feux_prets = {}
                nb_feux_pret = 0
        end
end
```

- Fonction exécutée toutes les 4 secondes :

```
-- Fonction exécutée périodiquement
boucle = function()
        client:connect("", "")
        while(running == true) do
                if(client:connected() == false) then client:connect("", "") end
                -- publie pour indiquer qu'il est prêt à changer de couleur
                client:publish(TOPIC_FEUX_PRET, uuid, mqtt.QOS0)
                tmr.delayms(4000)
                -- pause lorsque tous les feux sont prêts
                -- débloqué dans la fonction onFeuxPret
                while(nb_feux_pret == nb_feux) do
                        tmr.delayms(10)
                end
        end
        client:disconnect()
end
```

- Fonction permettant de lancer un feu :

```
initFeu = function(color)

        -- Connexion au réseau wifi
        initSTA()

        -- Initialise la couleur du feu au depart
        currentColor = color
        pio.pin.setlow(pio.GPIO2)
        pio.pin.setlow(pio.GPIO4)
        if(currentColor == "v") then pio.pin.sethigh(pio.GPIO4)
        else pio.pin.sethigh(pio.GPIO2) end

        -- Initialise les variables
        nb_feux = 0
        nb_feux_pret = 0
        feux_prets = {}

        -- Abonnements
        clientSub:connect("", "")
        clientSub:subscribe(TOPIC_COMPTEUR_FEUX, mqtt.QOS0, onCompteurFeux)
        clientSub:subscribe(TOPIC_NOMBRE_FEUX, mqtt.QOS0, onNombreFeux)
        clientSub:subscribe(TOPIC_FEUX_PRET, mqtt.QOS0, onFeuxPret)

        -- Déclare son existence en publiant dans le sujet TOPIC_COMPTEUR_FEUX
        client:connect("", "")
        client:publish(TOPIC_COMPTEUR_FEUX, 1, mqtt.QOS0) -- ie. +1 feu
        client:disconnect()

        -- Commence la boucle
        running = true
        th1 = thread.start(boucle) -- dans un thread pour les callbacks
end
```

Q2 - Ajout de la possibilité pour un feu de sortir du système

Principe :

Pour qu'un feu puisse sortir et rentrer du système en cours de fonctionnement, celui va devoir le déclarer en publiant dans un sujet MQTT, ainsi chaque feu recevant cette déclaration décrémente le nombre de feux qu'il connaît. Ainsi tous les feux recevant cette déclaration vont mettre à jour leur nombre de feux qu'ils connaissent.

Maintenant, pour qu'un feu puisse re-rentrer dans le système, il faut qu'il maintienne le même nombre de feux que tous les autres nœuds. Pour cela, chaque nœud (encore en fonctionnement) publie périodiquement le nombre de feux qu'ils connaissent pour permettre aux feux rentrant dans le système de mettre à jour leur nombre de feux.

Enfin pour qu'un feu re-rentre dans le système, il suffit d'appeler la fonction "initFeu" à nouveau.

Programme :

En pseudo-code :

Ajout : Arrêt du feu :

- Le feu publie dans le sujet "compteurFeux" le nombre -1

Modification : Boucle de synchronisation :

- Publie toutes les 4 secondes l'identifiant unique (uuid) du feu dans le sujet "feuxPret" pour indiquer qu'il est prêt à changer de couleur
- Publie toutes les 2 secondes le nombre de feu qu'il connaît afin de permettre aux feux qui rentrent dans l'ensemble d'être au courant

En Lua :

- Ajout de la fonction "StopFeu" :

```lua
-- Arrête le feu (pour maintenance par exemple)
stopFeu = function()

        -- Arrête la boucle
        running = false

        -- Extinction du feu
        pio.pin.setlow(pio.GPIO2)
        pio.pin.setlow(pio.GPIO4)

        -- Déclare que le feu n'existe plus en publiant dans le sujet TOPIC_COMPTEUR_FEUX
        if(client:connected() == false) then client:connect("", "") end
        client:publish(TOPIC_COMPTEUR_FEUX, -1, mqtt.QOS0) -- ie. -1 feu
end
```

Modification de la boucle :

```
-- Fonction exécutée périodiquement
boucle = function()
        client:connect("", "")
        while(running == true) do


                if(client:connected() == false) then client:connect("", "") end
                -- publie le nombre de feux connu pour la mise à jour des feux rentrants
                client:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)
                tmr.delayms(2000)
                if(client:connected() == false) then client:connect("", "") end
                -- publie le nombre de feux connu pour la mise à jour des feux rentrants
                client:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)

                -- publie pour indiquer qu'il est prêt à changer de couleur
                client:publish(TOPIC_FEUX_PRET, uuid, mqtt.QOS0)

                tmr.delayms(2000)

                -- pause lorsque tous les feux sont prêts
                -- débloqué dans la fonction onFeuxPret
                while(nb_feux_pret == nb_feux) do
                        tmr.delayms(10)
                        if(client:connected() == false) then client:connect("", "") end
                        -- re-publie "prêt" dans le cas où un feu sort du système
                        client:publish(TOPIC_FEUX_PRET, uuid, mqtt.QOS0)

                end
        end
        client:disconnect()
end
```

Q3 - Gestion des pannes

Principe :

Pour anticiper les pannes, il faut compter le nombre de cycles qu'un feu a effectué sans avoir changé de couleur. Dans ce cas, la valeur du nombre de cycles qu'un feu effectue avant de détecter une panne est 2. Lorsque cette valeur est atteinte, alors chaque feu décrémente son nombre de feu connu de 1 en supposant qu'un seul feu tombe en panne. Dans le cas où plusieurs feux tomberaient en panne, le nombre de cycle qu'un feu devra effectuer avant de changer de couleur est donc de 2 fois le nombre de feux tombés en panne.

Programme :

En pseudo-code :

Ajout : Initialisation d'un feu :

- Variable "k" permettant de compter le nombre de cycles qu'un feu effectue avant de détecter une panne.

Modification : Méthode exécutée à la réception d'un message depuis le sujet "feuxPret" :

- Lorsque l'on reçoit un message provenant de nous-même (l'uuid reçu correspond à l'uuid du feu) alors on incrémente la variable "k".
  Si "k" vaut 2, qui est le nombre de cycles avant qu'une panne soit détectée, alors on décrémente le nombre de feux connus ainsi on fait disparaître le feu en panne. Si plusieurs feux tombent en panne dans un intervalle court alors il faudra n cycles avant que le fonctionnement reprennent normalement où n = 2 fois nombre de feux tombés en panne.

En Lua :

- Ajout : initialisation :

```
k = 0 -- sert à compter le nombre de cycles avant la détection d'une panne
```

- Modification : méthode de callback "onFeuxPret" :

```
-- Méthode exécutée lors de la réception d'un message depuis
-- le sujet TOPIC_FEUX_PRET
onFeuxPret = function(length, msg)

        -- lorsqu'un feu est "bloqué", il continue de publier dans le sujet TOPIC_FEUX_PRET
        -- ainsi, on peut compter le nombre de cycle qu'il effectue
        if(uuid == msg) then
                 -- Gestion des pannes :
                k = k + 1 -- incrémente le nombre de cycles
                if(k == 2) then -- nombre de cycles avant détection d'une erreur
                        k = 0
                        -- décrémente le nombre de feux car on a été bloqué pendant
                        -- 2 cycles donc au moins un feu est en panne
                        nb_feux = nb_feux - 1
                        clientSub:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)

                end
        end

        -- vérifie si c'est un nouveau feu qui est pret avant
        -- d'incrémenter le nombre de feux prêts
        if(has_value(feux_prets, msg) == false) then
                nb_feux_pret = nb_feux_pret + 1
                table.insert(feux_prets, msg)
        end

        if(nb_feux_pret == nb_feux) then
```

```
                switchColor()
                -- Réinitialisation
                feux_prets = {}
                nb_feux_pret = 0
        end
end
```

Q4 - Reprise après une panne

Pour cela, il n'y a rien à modifier, il faut appeler à nouveau la méthode "initFeu".

Annexe 1 - Programme complet

```
pio.pin.setdir(pio.OUTPUT, pio.GPIO2) -- rouge
pio.pin.setdir(pio.OUTPUT, pio.GPIO4) -- vert

currentColor = "v" -- couleur du feu ("v" pour vert et "r" pour rouge)
nb_feux = 0 -- nombre de feu que l'on connait
nb_feux_pret = 0 -- nombre de feu qui ont indiques qu'ils etaient prets a changer de couleur
feux_prets = {} -- tableaux des feux prêts pour ce cycle

k = 0 -- sert à compter le nombre de cycles avant la détection d'une panne

TOPIC_COMPTEUR_FEUX = "/e1602246/compteurFeux"
TOPIC_NOMBRE_FEUX = "/e1602246/nombreFeux"
TOPIC_FEUX_PRET = "/e1602246/feuxPret"

-- Crer un client mqtt au broker
client = mqtt.client("e1602246-1", "mini.arsaniit.com", 1883, false)
clientSub = mqtt.client("e1602246-2", "mini.arsaniit.com", 1883, false)

-- booléen indiquant si la boucle doit tourner ou pas
running = false

-- uuid du feu
uuid = "feu1"

-- Fonction activant la wifi
initSTA = function()
        net.wf.setup(net.wf.mode.STA, "SSID", "KEY") net.wf.start()
end

-- fonction permettant de savoir si un élément fait parti d'un tableau
has_value = function (tab, val)
        for index, value in ipairs(tab) do
            if value == val then
                return true
            end
```

```
        end

        return false
end

-- change de couleur, si rouge alors vert et vice versa
switchColor = function()
        if(currentColor == "v") then
                currentColor = "r"
                pio.pin.setlow(pio.GPIO4)
                pio.pin.sethigh(pio.GPIO2)
        else
                currentColor = "v"
                pio.pin.setlow(pio.GPIO2)
                pio.pin.sethigh(pio.GPIO4)
        end
end

-- Fonction exécutée périodiquement
boucle = function()
        client:connect("", "")
        while(running == true) do
                if(client:connected() == false) then client:connect("", "") end
                -- publie le nombre de feux connu pour la mise à jour des feux rentrants
                client:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)
                tmr.delayms(2000)
                if(client:connected() == false) then client:connect("", "") end
                -- publie le nombre de feux connu pour la mise à jour des feux rentrants
                client:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)
                -- publie pour indiquer qu'il est prêt à changer de couleur
                client:publish(TOPIC_FEUX_PRET, uuid, mqtt.QOS0)
                tmr.delayms(2000)

                -- pause lorsque tous les feux sont prêts
                -- débloqué dans la fonction onFeuxPret
                while(nb_feux_pret == nb_feux) do
                        tmr.delayms(10)
                        if(client:connected() == false) then client:connect("", "") end
                        -- re-publie "prêt" dans le cas où un feu sort du système
                        client:publish(TOPIC_FEUX_PRET, uuid, mqtt.QOS0)
                end
        end
        client:disconnect()
end

-- arrête le feu (pour maintenance par exemple)
stopFeu = function()

        -- Arrête la boucle
        running = false

        -- Extinction du feu
```

```
        pio.pin.setlow(pio.GPIO2)
        pio.pin.setlow(pio.GPIO4)

        -- Déclare que le feu n'existe plus en publiant dans le sujet TOPIC_COMPTEUR_FEUX
        if(client:connected() == false) then client:connect("", "") end
        client:publish(TOPIC_COMPTEUR_FEUX, -1, mqtt.QOS0) -- ie. -1 feu
end

initFeu = function(color)

        -- Wifi
        initSTA()

        -- Initialise la couleur du feu au depart
        currentColor = color
        pio.pin.setlow(pio.GPIO2)
        pio.pin.setlow(pio.GPIO4)
        if(currentColor == "v") then
                pio.pin.sethigh(pio.GPIO4)
        else
                pio.pin.sethigh(pio.GPIO2)
        end

        -- Initialise les variables globales
        nb_feux = 0
        nb_feux_pret = 0

        -- uuid des feux pret en cours
        feux_prets = {}

        -- Abonnements
        clientSub:connect("", "")
        clientSub:subscribe(TOPIC_COMPTEUR_FEUX, mqtt.QOS0, onCompteurFeux)
        clientSub:subscribe(TOPIC_NOMBRE_FEUX, mqtt.QOS0, onNombreFeux)
        clientSub:subscribe(TOPIC_FEUX_PRET, mqtt.QOS0, onFeuxPret)

        -- Déclare son existance en publiant dans le sujet TOPIC_COMPTEUR_FEUX
        client:connect("", "")
        client:publish(TOPIC_COMPTEUR_FEUX, 1, mqtt.QOS0) -- ie. +1 feu
        client:disconnect()

        -- Commence la boucle
        running = true
        th1 = thread.start(boucle)
end

-- Méthode exécutée lors de la réception d'un message depuis
-- le sujet TOPIC_COMPTEUR_FEUX
onCompteurFeux = function(length, msg)
        -- modifie le nombre de feux connus
        nb_feux = nb_feux + msg
        -- publie sur le sujet TOPIC_NOMBRE_FEUX le nombre de feux connus
```

```
        clientSub:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)
    end

-- Méthode exécutée lors de la réception d'un message depuis
--  le sujet TOPIC_NOMBRE_FEUX
onNombreFeux = function(length, msg)
        if(nb_feux < msg) then nb_feux = msg end
end

-- Méthode exécutée lors de la réception d'un message depuis le sujet TOPIC_FEUX_PRET
onFeuxPret = function(length, msg)

        if(uuid == msg) then
                 -- Gestion des pannes :
                k = k + 1
                if(k == 2) then
                        k = 0
                        -- décrémente le nombre de feux car on a été bloqué pendant
                        -- 2 cycles donc au moins un feu est en panne
                        nb_feux = nb_feux - 1
                        clientSub:publish(TOPIC_NOMBRE_FEUX, nb_feux, mqtt.QOS0)

                end
        end


        -- vérifie si c'est un nouveau qui est pret avant
        -- d'incrémenter le nombre de feux prêts
        if(has_value(feux_prets, msg) == false) then
                nb_feux_pret = nb_feux_pret + 1
                table.insert(feux_prets, msg)
        end

        if(nb_feux_pret == nb_feux) then
                k = 0
                switchColor()
                -- Réinitialisation
                feux_prets = {}
                nb_feux_pret = 0
        end
end
```