|  |
| --- |
| ELECTIVES  Workshop : IOT  2023-24 |

# Suivi de modification

|  |  |  |
| --- | --- | --- |
| Date de modification | Auteur | Objet |
| 10/04/2021 | TPS | Première écriture de la démarche + screens |
| 08/03/2024 | Claire PERROT | Mise à jour du code + ajout de détails |

# Pré-Requis

A préparer avant le début du workshop :

* + CISCO Packet Tracer V8

# Actions techniques

##### IoT dans Packet Tracer

Ressources :

* Install Packet Tracer : voir netacad <https://www.netacad.com/portal/resources/packet-tracer>

L'outil choisi pour les simulations est Cisco Packet Tracer. Cet outil est utilisé depuis de nombreuses années pour former les étudiants aux réseaux Cisco. La principale force de l'outil est l'offre d'une variété de composants réseau qui simulent un réseau réel. Les périphériques doivent être interconnectés et configurés afin de créer un réseau. Dans la version 7 de l'outil, Cisco a introduit des fonctionnalités IoT, et il est maintenant possible d'ajouter au réseau des dispositifs intelligents, des composants, des capteurs, des actionneurs et aussi des dispositifs qui simulent des microcontrôleurs comme Arduino ou Raspberry Pi. Tous les appareils IoT peuvent fonctionner avec des programmes standard ou être personnalisés en les programmant avec Java, Python ou Blockly.

Cela fait de Cisco Packet Tracer un outil idéal pour créer des simulations d’infrastructure IoT.

Nous souhaitons donc créer une infrastructure composée de plusieurs éléments que nous pourrons résumer à ce schéma :

Texte de remplacement généré par une machine :
PC et mobile 
Serveur IOT 
Gestion des appareils 
Internet 
Passerelle (Gateway) 
Objets connectés 

Tâchons d’en comprendre les éléments.

* Quel est le rôle de la passerelle ?

Elle joue le rôle de pare feu

* Quel est le rôle du Cloud ?

Accéder a internet

* Quels sont la ou les choses les plus importantes pour l'utilisateur ?

Maintenant bien au clair avec ces différents éléments de notre infrastructure, nous pouvons commencer à la réaliser dans l’outil Packet tracer. Pour cela nous commencerons par Texte de remplacement généré par une machine :
2SSO-24TT 
SwitchO ajouter un switch

On va ajouter un switch 2960-24TT

On va ajouter un serveur que l'on nommera "ServerIoT"

On va ajouter plusieurs Objets Connectés :

* Un capteur de température
* Un capteur de passage photoélectrique
* Une lampe

 Que l'on nommera respectivement

* IoTTempérature
* IoTTripDetector
* IoTLight

Maintenant que tous nos éléments sont placés dans notre scène, nous allons dans un premier temps les relier via des câbles pour tous les connecter entre eux via le switch.

(Les câbles ne seront plus utilisés plus tard lorsque nous implémenterons les connexions sans fil)

Texte de remplacement généré par une machine :
loTTempérature 
Specifications 
MODULES 
PT -UNV-PWR-ADAPTER 
The IOT power adapteL 
Config 
Attributes 
Zoom In 
Physical Device Vieu 
Advanced 

Si vous essayez de connecter le capteur de température ou la lampe, vous noterez que c'est impossible car initialement, la liaison est configurée avec un module sans fil.

Texte de remplacement généré par une machine :
loTLight 
Specifications 
MODULES 
PT -UNV-PWR-ADAPTER 
The IOT power adapteL 
Config 
Attributes 
Zoom In 
Physical Device Vieu 
Original Size 
Zoom Out 
Advanced 

Pour la changer, rendez-vous sur l'interface "Advanced" de votre objet

Texte de remplacement généré par une machine :
loTLight 
Specifications 
1/0 Confi 
Network Adapter 
Network Adapter 2 
Digital Slots 
Analog Slots 
USB Ports 
Bluetooth 
Desktop 
LI sage 
Physical 
Config 
Thing Editor 
None 
Programming 
PT CFE 
PT CFE 
PT CGE 
PT -IOT-NM-IW 
PT -IDT -NM-IW-AC 
Show 
Smart Device 
Attributes 
o 
Component 
Advanced 

Puis dans "l'I/O Config" changez le connecteur réseau sur "PT-IOT-NM-1CFE" (vous noterez dans la vue physique le changement)

Texte de remplacement généré par une machine :
loTLight 
Specifications 
1/0 Config 
MODULES 
PT -UNV-PWR-ADAPTER 
The IOT power adapteL 
Config 
Zoom In 
Thing Editor 
Programming Attributes 
Physical Device Vieu 
Advanced 

Effectuez la manipulation pour la lampe et le capteur de température. Et une fois fait reliez vos 3 objets à votre switch

Nous attribuerons ensuite statiquement les adresses IP pour chaque élément de notre topologie tel que suit, ainsi qu'un masque de sous-réseau de 255.255.255.0.

Texte de remplacement généré par une machine :
S ensol 
e-te-ctO( 
882 c 
able 

La topologie est en place, nous allons pouvoir configurer notre serveur via le service IoT disponible sur Packet Tracer.

Texte de remplacement généré par une machine :
pc_pT Texte de remplacement généré par une machine :
S vices 
Ser 
1.1.1.1 
101 Light 
Serv810T 
Ccrii9 
SERVICES 
DHCP 
OHCPv6 
ONS 
SYSLOG 
EMAIL 
VM Management 
Radius EAP 
Attrb.*es 
S 8va 
O on 
@ Off 
D elete 

En activant ce service notre serveur deviendra un Registration Server pour regrouper tous nos objets.

Pour accéder à nos informations il nous faut ensuite ajouter un poste utilisateur que nous configurerons sur une IP statique comme suit :

À partir de là nous pourrons accéder via l'adresse IP de notre serveur à une interface web qui nous listera nos objets connectés.

Texte de remplacement généré par une machine :
pco 
Physical 
Config 
eh Browser 
LIRL 
Deskto 
Programming 
Attributes 
Registration Server Login 
pc_pT 
C semame: 
Password : 
Sign ln 
Don't have an 10E account? Sigu_up now 
Trip Sensor 
IoTTripDetector Vous noterez que l'on nous demande de nous connecter mais qu'aucun utilisateur n'existe, nous allons donc nous inscrire avec les informations suivantes :

Login : admin

Mdp : admin

Texte de remplacement généré par une machine :
pco 
Physical 
Config 
Deskto 
Programming 
Attributes 
eh Browser 
LIRL "home html 
[OT Server - Devices 
Home I Conditions I Editor I Lop Out 

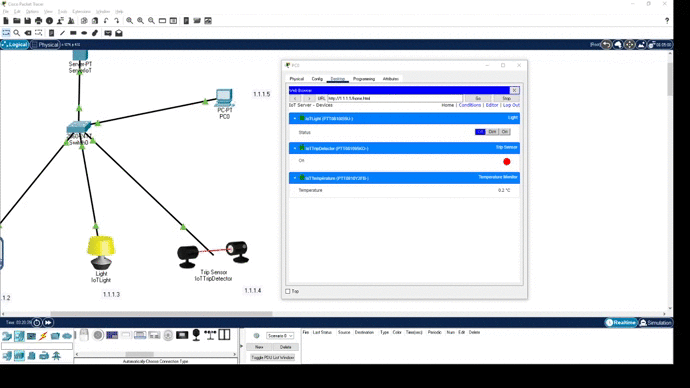
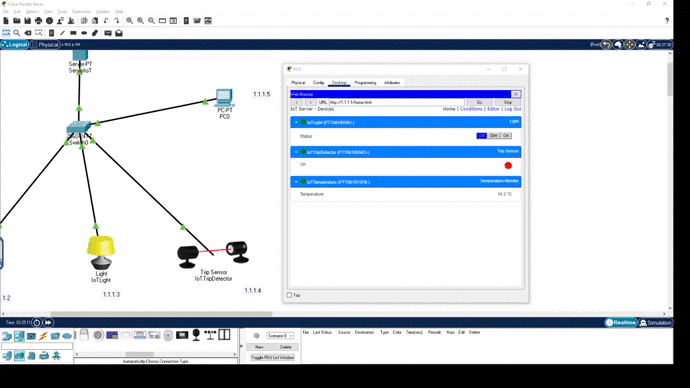
IcITempérature 
Specifications 
I/O Config 
0110 c 
T emperature Monitor 
IOTT empérature 
Gate„ay/DNS IPv4 
o 
Static 
Default Gateway 
DNS server 
Gate„ay/DNS IPv8 
o 
Automatic 
Static 
Default Gateway 
DNS server 
IOT Server 
o 
None 
o 
Home Gateway 
Remote Server 
Server Address 
user Name 
Password 
Thing Editor 
admin 
admin 
Programming 
Attributes 
GLOBAL 
Setti ngs 
Algorithm Settings 
Files 
INTERFACE 
Fast Etherneto 
aluetooth 
Refresh 
Advanced 

Nous pouvons enfin accéder à notre service qui ne liste encore aucun objet, nous allons pouvoir les y ajouter depuis les interfaces de chaque objet. Nous préciserons donc pour chaque objet un serveur distant où l'adresse IP est 1.1.1.1 et les identifiants admin / admin.

Une fois fait nous pouvons revenir à notre poste utilisateur et constater que les objets ont été ajoutés.

pco 
Physical 
Config 
eb Browser 
Deskto 
Programming 
Attributes 
URL /homahtml 
10T Server - Devices 
IoTTempérature (PTT0810Y2FB-) 
IOTLight (PTT08101sgU-) 
IoTTripoetector (PTT081095KO-) 
pc_pT 
rip Sensor 
T ripDetector 
Home I Conditions I Editor I Log Out 
Temperature Monitor 
Light 
Trip Sensor 

Vous pourrez de ce fait voir les informations des objets (état actuel) et agir dessus, en allumant ou éteignant la lampe par exemple (avec ALT + clic), ou en passant votre curseur (en maintenant ALT) sur le faisceau du capteur.

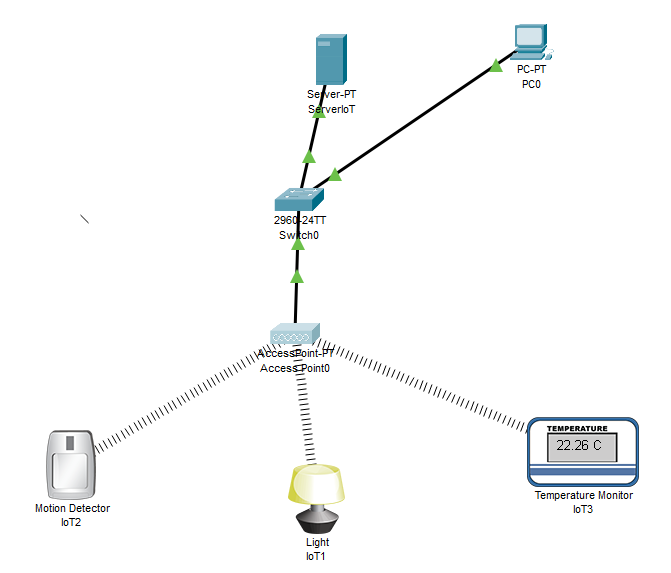


##### MQTT dans Packet Tracer

Changeons maintenant un peu notre topologie pour coller un peu plus à des conditions réelles.

1. Quels sont les protocoles les plus utilisés en loT ?
2. Qu'est ce qui les différencie ?

Voici une topologie que nous souhaitons reproduire, pour cela changez les IoT pour qu'ils soient connectés sans fil à un point d'accès :

* Ajouter un point d’accès « AccessPoint-PT » dans la topologie
* Dans sa config : sélectionner le port 1, changer la valeur de SSID à «  IOT-AP »
* Dans chaque IoT :
  + passer le network adapter à PT-IOT-NM-1W (dans les paramètres avancés)
  + dans la config : sélectionner l’interface Wireless0, puis mettre la valeur de SSID à « IOT-AP »
* Les IoT devraient être connectés au point d’accès :
* 

Sur le serveur que vous utiliserez comme broker, vous allez dans Desktop et lancez le gestionnaire d'applications utilisateur. Attendez un peu, jusqu'à ce qu'il soit entièrement chargé. Vous devriez maintenant pouvoir voir une application MQTT Broker ainsi qu'une application MQTT Client, comme indiqué ci-dessous.

Serverl 
Physical 
Config 
user Apps Manager 
Services 
Deskto 
Programming 
Attributes 
Installed 
Available 
MQTT Broker 
MQTT broker GUI and CLI app. 
Install 
ID: com.cisco.mqttbroker 
Version: 1.1 
Author: Tuan Hoang 
Contact: tuhoang@cisco.com 
Desktop GUI: Available 
Commands: 
MQTT Client 
MQTT client GUI and CLI app. 
Install 
ID: com.cisco.mqttclient 
Version: 1.1 
Author: Tuan Hoang 
Contact: tuhoang@cisco.com 
Desktop GUI: Available 
Commands: 

Choisissez le broker, cliquez sur installer, fermez le gestionnaire d'applications utilisateur et vous devriez voir l'icône de l'application MQTT Broker sur le Desktop. Vous pouvez maintenant démarrer le broker et lui fournir toutes les informations dont il a besoin.

Ensuite, vous devrez configurer les clients. Cette opération est similaire à celle du broker, à l'exception de certaines étapes que vous devez suivre avant de pouvoir lancer le User Apps Manager.

Lorsque vous ouvrez la configuration des appareils, vous ne pouvez pas voir de Desktop sur les appareils IoT. Pour changer cela, cliquez sur le bouton Advanced en bas à droite. Après être passé en mode avancé, votre fenêtre devrait ressembler à cet exemple, et vous pourrez lui ajouter le MQTT Client.

(Si Desktop ne s’affiche pas, il faut forcer l’affichage dans l’onglet I/O Config : Dekstop > Show).

Texte de remplacement généré par une machine :
loTLight 
Specifications 
Command 
1/0 Config 
Physical 
http : 
Web Browser 
Bluetooth 
Config 
Telnet / SSH Client 
Thing Editor 
Traffic Generator 
user Apps Manager 
Programming 
MUTT Client 
Attributes 
T ext Editor 
Advanced 

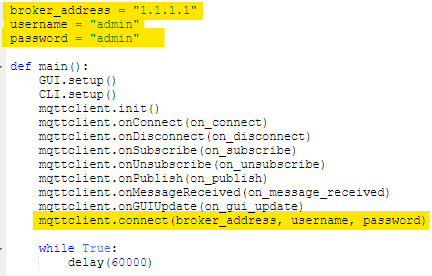
Maintenant que cette application est installée vous devez ajouter le serveur auquel le client va être connecté vous voyez que vous pouvez utiliser une interface MQTTClient pour cela (en utilisant l’identifiant / mot de passe suivant : admin/admin).

Texte de remplacement généré par une machine :
IcTLight 
Specifications 
MUTT Client 
1/0 Config 
Physical 
Config 
Deskto 
Thing Editor 
Programming 
Attributes 
MQTT Client 
Connection 
1.1.1.1 
IJsername (optional) 
Password (optional) 
Disconnect 
Subscriptions 
Topic 
Advanced 

Pour automatiser la chose, nous allons programmer cela dans le code python du service. Pour cela rendez-vous dans le mode "Advanced" dans l'onglet "Programming". Vous y trouverez le projet com.cisco.mqttclient, et le fichier main.py dans celui-ci.

IoTMotionDetectorI 
Specifications 
I/O Config 
No Project Opened 
Physical 
Config 
Desktop 
Attributes 
Open New Delete Rename I 
Motion Detector (Jav A 
Motion Detector (Pyt 
com.c.sco_m c ler 
Serial Outputs 
Advanced 10TLight2 
Specifications 
1/0 Config 
Physical 
Config 
com.cisco.mqttclient (Python) - main.py 
New Delete Rename Import 
Desktop 
Thing Editor 
Att[ibutes 
Help 
Reload Copy Paste Undo Redo Find Replace Zoom: + 
[gui]bootstrap nin .cs 
[gui]bo otstrap nin js 
[gui]iconpng 
[gui]index.html 
[gui]jqu er'/ min js 
[gui]scriptjs 
[gui]style. css 
app_manifestxml 
mqttclient.py 
Light 
25 
CII 
gul 
mat tellen: 
def gulEvent : 
İsen. (args) 
. ini t t) 
İf Zype 
"sZ2t2": 
Gül . ("3tBte", 
2LIf " cennect": 
2LIf "ölscznnect": 
2LIf 
2LIf 
2LIf 
(data ["tcplc"], , 
def cLIEvent : 
İf Zype 
"Invcked" 
Advanced 

Nous allons modifier ce code pour lui apporter toutes les fonctionnalités que nous voulons automatiser.

Commençons par la connexion à notre Broker, en ajoutant des variables et un appel à la méthode mqttclient.connect() dans le fichier main.py

Attention à bien mettre les bonnes valeurs dans username et password !

En relançant votre script avec le bouton "run" en haut de votre interface, vous pourrez le constater sur votre broker (un client doit être ajouté)



Texte de remplacement généré par une machine :
IcTLight 
Specifications 
MUTT Client 
1/0 Config 
Physical 
Config 
Deskto 
Thing Editor 
Programming 
Attributes 
ServerO 
Physical 
Config 
MUTT Broker 
Clients 
Services 
Deskto 
Programming 
Attributes 
Connection 
1.1.1.1 
IJsername (optional) 
Password (optional) 
Disconnect 
Subscriptions 
Topic 
Name (ID) 
loTMotionDetector (397a871 c4ab8aOL) 
PCO (397a871f4470aOL) 
loTLight (397a872286a350L) 
loTMotionDetector1 (397a87e2e7e7dOL) 
loTLight1 (397a880f388290L) 
Subscriptions 

On répètera la manipulation sur les autres IoT.

Nous récupèrerons ensuite le code permettant d'agir sur l'objet provenant du fichier main.py contenu dans le projet Light(Python). Nous y récupèrerons tous les imports, variables globales, et fonctions (petite précision, nous renommerons la fonction setup() en objsetup() pour préciser sa provenance)

Attention à :

* bien fusionner les deux fonctions main() : les boucles while doivent devenir une seule et même boucle.
* rajouter un délai avant la boucle while : Sleep(3)

def main() 
# call object 
objsetup() 
GUI. setup() 
CLI.setup() 
"e": "Off 
"1": "Dim", 
"name": "Status" , 
last TimelnSeconds - 
properties 
mqttclient. init() 
mqttcl ient. onconnect(on_connect) 
mqttcl ient. sconnect) 
mqttcl ient. onSubscri be (on_s ubscri be) 
mqttclient. 
mqttcl ient. onPub1 s h) 
mqttclient. connect(broker address, username, 
# delay (18eea) 
while True: 
# call looping function 
loop() 
def objsetup(): 
IoEC1ient. setup({ 
"type" : 
"Light", 
" states": 
"type" : 
"options" , 
"options": 
"e": "Off" , 
"1": "Dim", 
password) 
14 
27 
44 
def 
def 
def 
def 
main ( ) 
setup() 
ile True: 
loop() 
setup() : 
IoEC1ient. setup({ 
"type": " 
" states" : 
"name": "Status" , 
"type" : 
"options" , 
"options" : 
" controllable" : 
True 
" controllable" : 
True 
IoEC1ient. onlnputReceive(onInputReceive 
global state 
add event detect (e, detect) 
state = restoreproperty(" state", e) 
setstate(state) 
detect ( ) : 
processData(customRead (e), False) 
onlnputReceiveDone(ana10gInput) : 
processData(ana10gInput, True) 

Encore une fois on répètera l'opération sur les différents objets connectés en prenant soin d'adapter chaque code à son objet.

MQTT est un système d'émetteur / récepteur, qui permet de spécifier les messages que notre client souhaite recevoir ou envoyer.

Pour cela nous utilisons des "topic" qui sont les canaux de discussions auxquels nos objets pourront souscrire et ainsi recevoir tous les messages postés sur ce sujet.

C’est notre serveur qui jouera le rôle de médiateur entre les deux appareils et redistribuera les informations aux bons objets. Il veille à ce que les appareils restent connectés. Si un appareil edge-of-network perd sa connectivité, tous les clients qui y sont liés seront notifiés grâce à la fonctionnalité « Last Will and Testament »

Nous définirons les topics suivant une structure commune : *Usine/Pièce/TypeObjet/Donnée*

Par exemple :

* RSS/Bureau/Motion/status (pour envoyer les statuts des capteurs de mouvement dans le bureau)
* RSS/Bureau/Light/status (pour envoyer les statuts des lumières dans le bureau)
* RSS/Stocks/Motion/status (pour envoyer les statuts des capteurs de mouvement dans les stocks)
* RSS/Stocks/Light/status (pour envoyer les statuts des lumières dans les stocks)

Seules les lumières souscriront aux topics du Motion Sensor de leur pièce.

Nos MQTT clients n'auront donc plus qu'à souscrire automatiquement à ces canaux de discussion, reprenons donc le code pour y ajouter ces éléments (dans le fichier main.py)

23 
# Publication and subscriptions topics 
pub= " RSS/ Bureau/ Light/ status " 
status " 
issub=e 

#Publication and subscriptions topics

pub = "RSS/Burean/Light/status"

sub = "RSS/Bureau/Motion/status"

issub=0

Attention à bien modifier ces valeurs pour chaque composant pour qu’il soit inscrit (et publie) dans le bon canal !

Comme nous pourrions nous retrouver déconnectés et que la souscription ne fonctionne que si notre client est connecté, nous ferons notre requête de souscription dans notre boucle (en fusionnant avec la méthode loop déjà existante ou en en créant une s’il n’y a pas de méthode loop() au départ)  
Attention, pour le détecteur de température et de mouvement, les boucles loop() ne seront du coup pas les mêmes !

271 
272 
273 
274 
275 
276 
277 
278 
def 
loop() 
updateEnvironment() 
if issub 
e and 
sub " 
mqttclient. subscribe(sub) 
mqttclient. onmessageReceived (on_message received) 
mqttcl ient. onGUIUpdate (on_gui_update) 
sleep(l) 

Le code de loop() final sera donc le suivant pour la lampe (à adapter pour les capteurs de mouvement) :

def loop():

global issub

updateEnvironment()

if issub==0 and sub!="":

mqttclient.subscribe(sub)

issub=1

mqttclient.onMessageReceived(on\_message\_received)

mqttclient.onGUIUpdate(on\_gui\_update)

sleep(1)

Lorsque nous recevrons un message il faudra appeler la fonction permettant de changer l’état du composant, on modifie pour cela la méthode on\_message\_received(). Exemple pour la lampe :

174 
175 
176 
177 
178 
179 
18e 
181 
182 
183 
def 
on_message received (status, msg, packet): 
if status = 
"Success" or 
status 
" Error" : 
print status + " 
elif status 
print msg 
if packet["topic"]==sub: 
print 
at ' 
1) 
CLI 
split _ string = packet[" payload"] . split(" 
CO], True) 
. exit() 

La publication, elle, fera l'objet d'une fonction particulière dans laquelle nous utiliserons la méthode mqttclient.publish() en y précisant : le topic, la donnée (ou payload) et le niveau de QoS.

lge 
191 
192 
193 
194 
195 
196 
197 
198 
199 
2ee 
2e1 
2e2 
# Define publish function 
def mypublish(topic ,val): 
*topic change topic value if other room/object 
# Optional # 
at=" at 
t = localtime() 
actual _ time = strftime("XH:%M:%S", 
# Optional # 
payload=str(val)+at+actual_time 
qos="l" 
mqttclient. init() 
t) 
mqttcl ient. publish(topic , payload , qos ) 
print topic 

Cette fonction sera bien sûr appelée lors du changement d'état de notre objet en rajoutant un appel à la méthode mypublish() dans la méthode setState() déjà existante. Exemple pour la lampe :

287 
288 
289 
2ge 
291 
292 
293 
294 
295 
296 
297 
298 
def 
setState(newState) : 
global state 
if newState 3 . 
newState = e 
state = newState 
analogWrite(A1, state) 
state) 
# publish MQTT message 
mypublish( pub, state) 
IoEC1ient. reportstates(state) 
setDeviceProperty(getName(), "state" , 
state) 

def on\_message\_received(status,msg, packet):

if(status=="Success" or status=="Error"):

print status + ": "+ msg

elif status =="":

print msg

if packet["topic"]==sub:

print packet["payload"]

split\_string = packet["payload"].split(" at ",1)

processData(split\_string[0],True)

CLI.exit()

def mypublish(topic,val):

at=" at "

t = localtime()

actual\_time = strftime("%H:%M:%S",t)

payload=str(val)+at+actual\_time

qos="1"

mqttclient.init()

mqttclient.publish(topic, payload,qos)

print topic

La fonction main doit être similaire à celle-ci :

def main():

objsetup()

GUI.setup()

CLI.setup()

mqttclient.init()

mqttclient.onConnect(on\_connect)

mqttclient.onDisconnect(on\_disconnect)

mqttclient.onSubscribe(on\_subscribe)

mqttclient.onUnsubscribe(on\_unsubscribe)

mqttclient.onPublish(on\_publish)

mqttclient.onMessageReceived(on\_message\_received)

mqttclient.onGUIUpdate(on\_gui\_update)

mqttclient.connect(broker\_address,username,password)

sleep(3)

while True:

loop()

Après avoir relancé le programme et appliqué ces modifications à tous nos objets nous devrions observer toutes les souscriptions.

RSS/ Bureau/ Light/ 
cannected Cc Z:aker. 
Success: Connected at L. 
Success: Subscribed RSS/ Burea 
status . 

ServerO 
Physical 
Config 
MOTT Broker 
Services 
Deskto 
Programming 
10TLight1 (397b7086a8a420L) 
Subscriptions 
Topic 
RSS/Bureau/Motion/status 
RSS/Stock/Motion/status 
Event Log 
Attributes 
Clients 
10TLight (397b6ed5a5a890L) 
10TLight1 (397b7086a8a420L) 
{ " cmd " 
{ " cmd " 
" : ID" , "ID" : "397b6cfb4bf15ØL"} 
" client" : "10TLight (397b6cfb4bf15øL)% " protocol 
CONNECT , " 

Vous pouvez maintenant tester le système en mode realtime et passer devant votre capteur (touche ALT maintenue) pour constater que la lumière s'allume bien.

ccm. . rr.zZZc2ž_enZ 
ccm. 
RES/ 
E z zcr: ccnnecZeă Ezoker. 
Succesz: Connecr-eă L. 
Succesz: SubscržEeă . 
Succesz: Recežvecl mess age 
RES/ 
Succesz: Recežvecl mess age 
RES/ 
Succesz: Recežvecl 
RES/ 
Succesz: 
Succesz: Recežvecl 
RES/ 
: me ssage 
from 
from 
from 
from 
'RSS/Bureau/Lžghz/sr-azus' QcS 
' RSS/Bureau/Lžghz/sr-azus' QcS 
'RSS/Bureau/Lžghz/sr-azus' QcS 
'RSS/Bureau/Lžghz/sr-azus' with QcS 
Level 
Level 
Level 
Level 

Vous pourrez même constater en mode "Simulation" le déplacement des paquets et le fonctionnement de MQTT.

Quel est le chiffrement des données apporté par MQTT ?

Texte de remplacement généré par une machine :
Quality of Service (QoS) 
at most once 
(fire and forget) 
PUBLISH 
Message deleted; 
does not survtve 
no dupltates 
QoS1 
at most once 
PUBLISH 
PUBACK 
Message stored•, 
survive ccnnectltn lost; 
duNcate possible 
MQTT 
Broker 
QoS2 
exactly once 
PUBLISH 
PUBREC 
PUBREL 
PUBCOMP 
NEssaoe stored; 
survives conrpctlon 
bSS; no dupucates 

Le dernier paramètre que nous avons laissé de côté est le paramètre de QoS (Quality of Service).

Pour chaque topic, MQTT permet de gérer la QoS désirée, les messages seront délivrés :

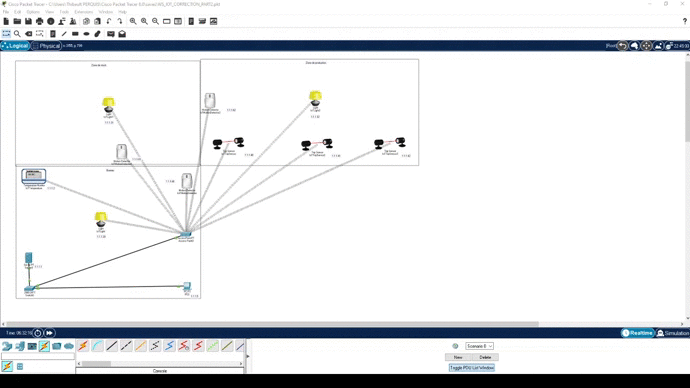
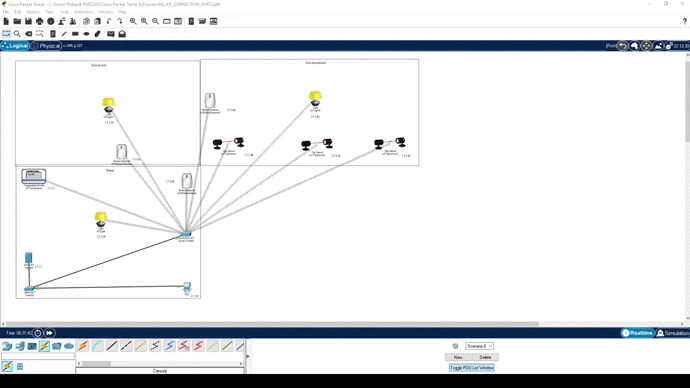
* QoS 0 : Une fois au plus
* QoS 1 : Une fois au moins
* QoS 2 : Exactement une seule fois

TCP assure la fiabilité des transmissions sur le réseau. Mais MQTT permet en plus de gérer les pertes de connexion, lors d’une reconnexion, en QoS 1 et QoS 2, les informations non transmises sont retransmises.

Par exemple en QoS 1 les messages seront validés par le client receveur, s’il ne confirme pas, le message lui sera retransmis jusqu'à validation.

En QoS 2 le message s'auto-valide pour être sûr de n'être envoyé qu'une seule fois.

Par contre en QoS0 les informations sont uniquement transmises et validées par TCP.



# Pour aller plus loin

Voilà, vous savez désormais comment développer et installer une infrastructure IoT ! Pour aller plus loin n'hésitez pas à imaginer et mettre en application d'autres scénarios d'interactions entre les différents clients, en associant différents objets, en mettant en place plusieurs usines distantes, en faisant interagir plusieurs objets…