HTTPServer (hérité de TCPServer)

from tcpcom import *

server = HTTPServer (requestHandler, serverName = "PYSERVER", port = 80, isVerbose = False)	Crée un serveur HTTPS (serveur Web, hérité de TCPServer) qui écoute un client se connectant sur un port donné (valeur par défaut = 80). Démarre un thread qui gère et retourne les demandes HTTP GET. L'en-tête de réponse HTTP inclut le nom du serveur donné (par défaut: PYSERVER). Seules les réponses textuelles sont prises en charge. requestHandler () est une fonction de rappel appelée lorsqu'une demande GET est reçue. Signature: msg, stateHandler = requestHandler (clientIP, nom de fichier, paramètres) Paramètres: clientIP: adresse IP du client au format de point marqué nom de fichier: le nom de fichier demandé avec les paramètres précédents '/' : un tuple de format : ((param_key1, param_value1), (param_key2, param_value2), ...) (tous les éléments sont des chaînes) Valeurs de retour: msg: réponse textuelle HTTP (l'en-tête est automatiquement créé) stateHandler: fonction de rappel invoquée immédiatement après l'envoi de la réponse. Si stateHandler = None, rien n'est fait. La fonction peut inclure des actions plus longues du serveur ou un temps d'attente, si les capteurs ne sont pas immédiatement prêts pour de nouvelles mesures. Appelez terminate () pour arrêter le serveur. La connexion est fermée par le serveur à la fin de chaque réponse. Si le client se connecte mais n'envoie pas de demande dans les 5 secondes, la connexion est fermée par le serveur.
getClientIP ()	Renvoie l'IP pointée d'un client connecté. Si aucun client n'est connecté, retourne une chaîne vide
getServerIP ()	Renvoie l'adresse IP du serveur (méthode statique)

MQTT Client

from mqttclient import *

MQTTClient

client = MQTTClient (messageReceived = None, nom d'utilisateur = "", mot de passe = "")	crée un MQTTClient qui publie et / ou souscrit des sujets MQTT. Le client ne se connecte pas encore à un courtier MQTT. messageReceived (topic, message) est une fonction de rappel déclenchée par les messages entrants et exécutée dans un thread distinct. Si aucun, aucune notification de message n'est déclenchée, par exemple pour un client qui ne publie que des sujets. nom d'utilisateur est utilisé pour se connecter au courtier MQTT (vide, si aucune authentification d'utilisateur n'est nécessaire). password est le mot de passe utilisé pour se connecter au courtier MQTT (vide, si aucune authentification utilisateur n'est nécessaire)
client.connect (hôte, port = 1883, keepalive = 60)	démarre un essai de connexion au courtier MQTT donné sur le port donné. hôte est l'adresse IP du courtier et le port sur lequel il écoute (par défaut: 1883). keepalive est la période maximale en secondes entre les communications avec le courtier. Si aucun autre message n'est échangé, il s'agit de la période de temps pour l'envoi de messages ping au courtier (valeur par défaut: 60 s). Renvoie True si la connexion est réussie. sinon Faux

client.disconnect()	déconnecte le client du courtier
client.publish (sujet, charge utile, qos = 0, conserver = faux)	envoie un message avec le sujet et la charge donnés au courtier. payload est une chaîne (si un int ou float est donné, il est converti en chaîne). qos est le niveau de qualité de service (0, 1, 2, 0 par défaut). retenue détermine si le message est le «dernier message de confirmation / valide» pour le sujet (par défaut: False)
client.subscribe (topic, qos = 0)	abonne le client à un ou plusieurs sujets. topic est une chaîne ou une liste de n-uplets de format (topic, qos). qos est le niveau de qualité de service (numéro 0, 1, 2. par défaut: 0); non utilisé, si topic est une liste de tuples '
client.setVerbose(verbose)	active / désactive les informations de l'enregistreur imprimées sur stdout