Bluetooth Client/Server

from btcom import *

BTServer

server = BTServer (serviceName, stateChanged, isVerbose = False)	crée un serveur Bluetooth qui expose le service RFCOMM avec le nom serviceName. Les changements d'état sont notifiés par le callback stateChanged (). Pour isVerbose = True, les messages de débogage sont écrits dans la fenêtre de sortie.
stateChanged (state, msg)	Callback appelé lors d'événements de changement d'état. state: "LISTENING", msg: vide state: "CONNECTED", msg: informations distantes: nom Bluetooth (adresse MAC) state: "TERMINATED", msg: vide state: "MESSAGE", msg: message reçu
server.disconnect ()	ferme la connexion avec le client et passe à l'état LISTENING
server.isConnected ()	True, si un client est connecté au serveur
server.terminate ()	ferme la connexion et met fin à l'état LISTENING. Libère les ressources internes
server.isterminated ()	True, si le serveur a été arrêté
server.sendMessage (msg)	envoie les informations au client (String, le caractère \ 0 (ASCII 0) sert d'indicateur de fin de chaîne, il est ajouté et supprimé de manière transparente)
BTServer.getVersion ()	renvoie la version du module sous forme de chaîne

BTClient

client = BTClient(stateChanged, isVerbose = False)	crée un client Bluetooth préparé pour une connexion avec un serveur BTS. Les changements d'état sont notifiés par le callback stateChanged (). Pour isVerbose = True, les messages de débogage sont écrits dans la fenêtre de sortie.
client.findServer (serverName, timeout)	effectue une recherche de périphérique pour le nom Bluetooth du serveur donné. Renvoie le tuple serverInfo: ("nn: nn: nn: nn: nn: nn", canal), par exemple ("B8: 27: EB: 04: A6: 7E", 1). Si le serveur est introuvable, Aucun est renvoyé. La recherche est répétée jusqu'à ce que le délai (en s) soit atteint
client.findService (serviceName, timeout)	effectue une demande de service pour un serveur Bluetooth exposant le service RFCOMM donné. Renvoie le tuple serverInfo: ("nn: nn: nn: nn: nn: nn", canal), par exemple ("B8: 27: EB: 04: A6: 7E", 1). Si le service n'est pas trouvé, Aucun est renvoyé. La recherche est répétée jusqu'à ce que le délai (en s) soit atteint
stateChanged (état, msg)	Callback appelé lors d'événements de changement d'état. state: "CONNECTING", msg: informations sur le serveur (adresse MAC, canal Bluetooth) state: "CONNECTED", msg: informations sur le serveur state: "DISCONNECTED", msg: vide state: "CONNECTION_FAILED", msg: informations sur le serveur state: "MESSAGE ", msg: message reçu
client.connect (serverInfo, timeout)	Effectue un essai de connexion au serveur avec serverInfo donné. Si l'essai de la connexion échoue, il est répété jusqu'à atteindre le délai (en s). Renvoie True si la connexion est établie. sinon, False est renvoyé. serverInfo est un tuple avec l'adresse MAC et le numéro de canal ("nn: nn: nn: nn: nn: nn", canal), par exemple ("B8: 27: EB: 04: A6: 7E", 1)
client.isConnecting ()	renvoie True lors de l'essai de connexion
client.isConnected ()	renvoie True si le client est connecté

client.disconnect()	met fin à la connexion au serveur
client.sendMessage (msg,)	envoie les informations au serveur (string, le caractère \ 0 (ASCII 0) sert de fin de message, automatiquement ajouté et supprimé de manière transparente).
BTClient.getVersion ()	renvoie la version du module (string)