Documentation GPanel

makeGPanel()	Crée une fenêtre de graphiques GPanel avec un système de coordonnées où x et y varient entre 0 et 1. Le curseur graphique est placé sur l'origine (0, 0) au coin inférieur gauche de la fenêtre
makeGPanel(xmin, xmax, ymin, ymax)	Crée une fenêtre GPanel avec les coordonnées flottantes indiquées. Le curseur graphique est placé en (0,0)
makeGPanel(xmin, xmax, ymin, ymax, False)	Idem, mais en cachant la fenêtre (pour la rendre visible, appeler visible(True))
makeGPanel(Size(width, height))	Idem que makeGPanel(), mais en spécifiant la taille de la fenêtre (en pixels)
getScreenWidth()	Retourne la largeur de l'écran (en pixels)
getScreenHeight()	Retourne la hauteur de l'écran (en pixels)
window(xmin, xmax, ymin, ymax)	Change la plage de coordonnées utilisées pour la fenêtre
drawGrid(x, y)	Dessiner une grille pour le système de coordonnées avec un marqueur tous les x pour l'axe horizontal et tous les y pour l'axe vertical. Les étiquettes des marqueurs sont déterminées par le type des paramètres x et y : int ou float
drawGrid(x, y, color)	Idem, en spécifiant la couleur color de la grille
drawGrid(x1, x2, y1, y2)	Idem, en indiquant la plage de coordonnées x1..x2, y1..y2 sur laquelle s'étend la grille
drawGrid(x1, x2, y1, y2, color)	Idem, en spécifiant la couleur color de la grille
drawGrid(x1, x2, y1, y2, x3, y3)	Idem, en spécifiant les unités utilisées pour la grille avec x3, y3 sur l'axe horizontal, respectivement vertical
drawGrid(x1, x2, y1, y2, x3, y3, color)	Idem, en spécifiant la couleur color de la grille
drawGrid(p, ...)	Idem que drawGrid() en utilisant le GPanel référencé par la variable p. Utile lors de l'utilisation conjointe de plusieurs panels
visible(isVisible)	Affiche / Cache la fenêtre
resizeable(isResizeable)	Spécifie si la fenêtre est redimensionnable. Valeur par défaut : non
dispose()	Ferme la fenêtre et libère les ressources utilisées pour le dessin
isDisposed()	Retourne True si la fenêtre est fermée par le bouton « fermer » de la barre de titre ou suite à l'appel de la fonction dispose()
bgColor(color)	Règle la couleur d'arrière-plan. Le paramètre color est une chaîne de couleur X11 ou un objet couleur retourné par le constructeur makeColor()
title(text)	Affiche text dans la barre de titre de la fenêtre
makeColor(colorStr)	Retourne un objet de type couleur correspondant à la chaine de couleur X11 passée en paramètre
windowPosition(ulx, uly)	Règle la position de la fenêtre par rapport aux coordonnées de l'écran (en pixels)
windowCenter()	Centre la fenêtre au milieu de l'écran
storeGraphis()	Stocke le graphique courant dans une mémoire tampon graphique
recallGraphics()	Effectue le rendu du contenu de la mémoire tampon dans le canevas GPanel
clearStore(color)	Efface la mémoire tampon graphique en y peignant avec la couleur indiquée
delay(time)	Met le programme en pause pour l'intervalle de temps indiqué par time (en ms)
getDividingPoint(pt1, pt2, ratio)	Retourne les coordonnées du point qui sépare le segment reliant les points p1 et p2 avec le rapport ratio. Le rapport peut être négatif ou supérieur à 1
getDividingPoint(c1, c2, ratio)	Idem, en utilisant les nombres complexes c1 et c2
clear()	Réinitialise la fenêtre graphique en effaçant son contenu et en replaçant le curseur graphique à l'origine (0, 0)
erase()	Efface le contenu de la fenêtre graphique sans réinitialiser la position du curseur graphique
putSleep()	Met le programme en pause jusqu'à l'appel ultérieur de la fonction wakeUp()

wakeUp()	Remet le programme en route suite à une mise en pause avec putSleep()
linfit(X, Y)	Effectue une régression linéaire y = a*x + b sur la base des données contenues dans les listes X et Y et retourne les coefficients de la droite sous la forme du tuple (a, b)
addExitListener(onExit)	Enregistre une fonction de rappel onExit() qui est applée à un clique sur fermeture de fenêtre. Pour fermet la fenêtre, appeler dispose()

Dessin

lineWidth(width)	Règle la largeur des lignes (en pixels)
setColor(color)	Règle la couleur de dessin à color (chaîne de couleur X11 ou objet de type Color)
move(x, y)	Place le curseur graphique à la position (x, y) sans dessiner de ligne
move(liste)	Idem, en spécifiant les coordonnées dans la liste coord_list=[x, y]
move(c)	Idem, en utilisant le nombre complexe c pour spécifier les coordonnées
getPosX()	Retourne la coordonnée x de la position du curseur
getPosY()	Retourne la coordonnée y de la position du curseur
getPos()	Retourne les coordonnées de la position du curseur sous forme de liste
draw(x, y), lineTo(x, y), lineto(x, y)	Dessine une ligne depuis la position actuelle jusqu'au point (x, y) et met à jour la position du curseur avec (x, y)
draw(list), lineTo(coord_list), lineto(coord_list)	Idem, en spécifiant les coordonnées dans une liste de la forme [x, y]
draw(c), lineTo(c), lineto(c)	Idem, en spécifiant les coordonnées par le nombre complexe c
line(x1, y1, x2, y2)	Dessine une ligne de (x1, y1) vers (x2, y2) sans modifier la position actuelle du curseur
line(pt1, pt2)	Idem, en spécifiant les coordonnées des points de départ et d'arrivée de la ligne avec les listes pt1 = [x1, y1] et pt2 = [x2, y2]
line(c1, c2)	Idem, en spécifiant les coordonnées des points de départ et d'arrivée de la ligne par les nombres complexes c1 et c2
line(li[p0, p1, ..])	Dessine le trajet partant du point p0 de tous les points de la liste (ou tuple)
circle(radius)	Dessine un cercle de rayon radius centré à la position actuelle du curseur graphique
fillCircle(radius)	Dessine un disque plein de rayon radius centré à la position actuelle du curseur graphique. La couleur de remplissage utilisée est déterminée par la couleur actuelle du pinceau
ellipse(a, b)	Dessine une ellipse vide d'axes a et b centrée à la position actuelle du curseur graphique
fillEllipse(a, b)	Dessine une ellipse pleine d'axes a et b centrée à la position actuelle du curseur graphique. La couleur de remplissage utilisée est déterminée par la couleur actuelle du pinceau
rectangle(a, b)	Dessine un rectangle de côtés a et b centré à la position actuelle du curseur graphique
rectangle(x1, y1, x2, y2)	Idem, en spécifiant le point supérieur gauche (x1, y1) et le point inférieur droit (x2, y2)
rectangle(pt1, pt2)	Idem, en spécifiant les points de la diagonale par les listes à deux éléments p1 et p2
rectangle(c1, c2)	Idem, en utilisant les nombres complexes c1 et c2 pour spécifier les coordonnées
fillRectangle(a, b)	Dessine un rectangle plein de côtés a et b centré à la position actuelle

	du curseur graphique. La couleur de remplissage utilisée est déterminée par la couleur actuelle du pinceau
fillRrectangle(x1, y1, x2, y2)	Idem, en spécifiant le point supérieur gauche (x1, y1) et le point inférieur droit (x2, y2)
fillRectangle(pt1, pt2)	Idem, en spécifiant les points de la diagonale par les listes à deux éléments p1 et p2
fillRrectangle(c1, c2)	Idem, en utilisant les nombres complexes c1 et c2 pour spécifier les coordonnées
arc(radius, startAngle, extendAngle)	Dessine un arc de cercle de rayon radius centré à la position du curseur et d'angle au centre extendAngle. startAngle indique l'angle du point de départ par rapport à l'horizontale. Le sens positif est le sens contraire des aiguilles de la montre
fillArc(radius, startAngle, extendAngle)	Idem, mais en dessinant un secteur circulaire plein. La couleur de remplissage utilisée est déterminée par la couleur actuelle du pinceau
polygon(x-list, y-list)	Dessine le polygone dont les coordonnées des sommets sont spécifiées par les listes x_list, respectivement y_list
polygon((li[pt1, pt2,..)	Idem en spécifiant les sommets dans une liste de points pti représentés par des listes de deux éléments [x,y]
polygon(li[c1, c2, c3,...])	Idem, en utilisant une liste de nombres complexes c1, c2, c3, … pour spécifier les coordonnées des sommets.
fillPolygon(x-list, y-list)	Dessine le polygone dont les coordonnées des sommets sont spécifiées par les listes x_list, respectivement y_list. La couleur de remplissage utilisée est déterminée par la couleur actuelle du pinceau
fillPolygon((li[pt1, pt2,..)	Idem en spécifiant les sommets dans une liste de points pti représentés par des listes de deux éléments [x,y]
fillPolygon(li[c1, c2, c3,...])	Idem, en utilisant une liste de nombres complexes c1, c2, c3, … pour spécifier les coordonnées des sommets
lowerPath, upperPath, hull = getHull(li[pt1, pt2,...])	Retourne tuples lowerPath, upperPath et hull de la coque convexe des points pt1, pt2,...
quadraticBezier(x1, y1, xc, yc, x1, y2)	Dessiner la courbe de Bézier quadratique définie par les points extrémaux (x1, y1) et (x2, y2) ainsi que le point de contrôle (xc, yc)
quadraticBezier(pt1, pc, pt2)	Idem en spécifiant les points avec des listes de deux éléments
quadraticBezier(c1, cc, c2)	Idem en spécifiant les points avec des nombres complexes
cubicBezier(x1, y1, xc1, yc1, xc2, yc2, x2, y2)	Dessiner la courbe de Bézier cubique définie par les points extrémaux (x1, y1) et (x2, y2) ainsi que les deux points de contrôle (xc1, yc1) et (yc2, yc2)
cubicBezier(pt1, ptc1, ptc2, pt2)	Idem en spécifiant les points avec des listes de deux éléments
cubicBezier(c1, cc1, cc2, c2)	Idem en spécifiant les points avec des nombres complexes
triangle(x1, y1, x2, y2, x3, y3)	Dessine un triangle de sommets (x1, y1), (x2, y2), (x3, y3)
triangle(pt1, pt2, pt3)	Idem en spécifiant les sommets avec des listes de deux éléments
triangle(c1, c2, c3)	Idem en spécifiant les sommets avec des nombres complexes
fillTriangle(x1, y1, x2, y2, x3, y3)	Dessine un triangle plein dont les sommets sont spécifiés par les points (x1, y1), (x2, y2), (x3, y3). La couleur de remplissage utilisée est déterminée par la couleur actuelle du pinceau
fillTriangle(pt1, pt2, pt3)	Idem en spécifiant les sommets avec des listes de deux éléments
fillTriangle(c1, c2, c3)	Idem en spécifiant les sommets avec des nombres complexes
point(x, y)	Dessine un point isolé (pixel) à la position (x, y)
point(pt)	Idem en spécifiant les coordonnées par une liste de deux éléments
point(complex)	Idem en spécifiant les coordonnées par un nombre complexe
fill(x, y, color, replacementColor)	Remplit la surface fermée contenant le point (x, y) en remplaçant tous les

	pixels actuellement de couleur color par un pixel de couleur replacementColor
fill(pt, color, replacementColor)	Idem, en spécifiant les coordonnées du point par la liste de deux éléments pt
fill(complex, color,replacementColor)	Idem, en spécifiant les coordonnées du point par le nombre complexe complex
image(path, x, y)	Insère l'image au format GIF, PNG ou JPEG stockée dans le fichier de chemin path. Place son coin inférieur gauche en (x,y). Le chemin path peut être relatif au dossier TigerJython, être contenu dans le dossier sprites de l'archive JAR de la distribution TigerJython ou être une URL débutant par http://
image(path, pt)	Idem, en spécifiant les coordonnées du coin inférieur gauche par la liste de deux éléments pt
image(path, complex)	Idem, en spécifiant les coordonnées du coin inférieur gauche par le nombre complexe complex
imageHeighpath)	Retourne la hauteur de l'image présente dans le fichier path (en pixels)
imageWidth(path)	Retourne la largeur de l'image présente dans le fichier path (en pixels)
enableRepaint(boolean)	Active/désactive le rendu automatique du tampon graphique. Valeur par défaut : True = activé
repaint()	Effectue le rendu du tampon graphique à l'écran. Nécessaire si le rendu automatique est désactivé
setPaintMode()	Active le mode de dessin standard qui dessine par-dessus l'arrière-plan
setXORMode(color)	Active le mode de dessin XOR qui effectue une combinaison XOR entre les pixels de l'arrière-plan et la couleur de dessin color. Deux dessins successifs identiques en mode XOR s'annulent pour redonner la couleur d'arrière-plan initiale
getPixelColor(x, y)	Retourne la couleur du pixel situé aux coordonnées (x, y) comme un objet de type Color
getPixelColor(pt)	Idem, pour le point dont les coordonnées sont indiquées dans la liste à deux éléments pt
getPixelColor(complex)	Idem, pour le point dont les coordonnées sont indiquées par le nombre complexe complex
getPixelColorStr(x, y)	Retourne la couleur du pixel situé aux coordonnées (x, y) comme une chaîne de couleur X11
getPixelColorStr(pt)	Idem, pour le point dont les coordonnées sont indiquées dans la liste à deux éléments pt
getPixelColorStr(complex)	Idem, pour le point dont les coordonnées sont indiquées par le nombre complexe complex

Texte

text(string)	Insère le texte contenu dans la chaîne string à partir de la position actuelle du curseur graphique
text(x, y, string)	Idem, en écrivant à partir du point de coordonnées (x, y)
text(pt, string)	Idem en spécifiant les coordonnées du point de départ par une liste de deux éléments
text(complex, string)	Idem en spécifiant les coordonnées du point de départ par un nombre complexe
text(x, y, string, font, textColor, bgColor)	Affiche le texte contenu dans string à partir de la position (x, y) en utilisant la police font, la couleur de texte textColor et la couleur d'arrière-fond bgColor
text(pt, string, font, textColor, bgColor)	Idem en spécifiant les coordonnées du point de départ par une liste de deux éléments

text(complex,string, font, textColor, bgColor)	Idem en spécifiant les coordonnées du point de départ par un nombre complexe
font(font)	Sélectionne une autre police pour les appels subséquents à text()

Fonctions de rappel

makeGPanel(mouseNNN = onMouseNNN)	Enregistre une fonction de rappel onMouseNNN(x,y) qui est appelée à chaque fois qu'un événement souris survient. Les valeurs possibles pour NNN sont: Pressed, Released, Clicked, Dragged, Moved, Entered, Exited, SingleClicked, DoubleClicked
isLeftMouseButton() isRightMouseButton()	Retourne True si le dernier événement souris a été généré par un clique gauche, respectivement droit
makeGPanel(keyPressed = onKeyPressed)	Enregistre la fonction de rappel onKeyPressed(keyCode) qui est appelée
getKeyModifiers()	Retourne un nombre entier lorsque des touches spéciales du clavier sont enfoncées (Ctrl, Majuscule, Alt). Il est également possible d'obtenir ainsi les combinaisons de touches spéciales, par exemple Ctrl + Maj
makeGPanel(closeClicked = onCloseClicked)	Enregistre la fonction de rappel onCloseClicked() qui est appelée lorsque le bouton « fermer » de la barre des tâches est cliqué. On peut fermer la fenêtre à l'aide de l'appel dispose()
showSimulationBar(NNN = onNNN)	Affiche barre de contrôle avec les boutons 'Step', 'Run'/'Pause', 'Reset' et un ascenceur pour régler la durée de chaque période de simulation.
showSimulationBar(ulx, uly, initPeriod, NNN = onNNN)	Idem mais en indiquant la position de la barre (coin haut gauche) et avec une durée de simulation de 100 par défaut.
hideSimulationBar()	Ferme la barre de contrôle et libère les ressources.

KeyBoard

getKey()	Retourne, sous forme de chaîne de caractères, le caractère correspondant à la dernière touche du clavier enfoncée
getKeyCode()	Retourne le code (nombre entier) de la dernière touche du clavier enfoncée
getKeyWait()	Met le programme en pause jusqu'à ce qu'une touche du clavier soit actionnée et retourne le caractère en question sous forme de chaine de caractères
getKeyCodeWait()	Idem, en retournant le code de la dernière touche enfoncée sous forme de nombre entier
kbhit()	Retourne True si une touche du clavier a été pressée depuis le dernier appel à getKey() ou getKeyCode()

Composants d'interface graphique

add(component)	Insère un composant GUI vers le bord supérieur de la fenêtre
validate()	Redessine la fenêtre (et son contenu) après qu'un composant graphique a été rajouté avec add()

addStatusBar(height)	Ajoute au bas de la fenêtre une barre d'état dont la hauteur est donnée par height (en pixels)
setStatusText(text)	Affiche le texte text dans la barre d'état. Le texte qui y figurait est supprimé
setStatusText(text, font, color)	Idem en indiquant la police et la couleur de texte à utiliser

Police

font(name, style, size)	Crée un nouvel objet de type police de caractères utilisant la police nommée name, dans le style style et la taille de caractères size. Les différents paramètres ainsi que leur type sont décrits dans les trois lignes ci-dessous
name	Chaîne de caractères décrivant une police de caractères installée sur le système, comme par exemple "Times New Roman", "Arial" ou "Courier"
style	Nombre entier parmi les constantes suivantes: Font.PLAIN, Font.BOLD, Font.ITALIC. Ces constantes peuvent être combinées par addition, comme par exemple: Font.BOLD + Font.ITALIC
size	Nombre entier correspondant à une taille disponible pour la police de caractères choisie, par exemple 12, 16, 72

Boîtes de dialogue

msgDlg(message)	Ouvre une boîte de dialogue modale avec le message message et un bouton OK
msgDlg(message, title = text)	Idem, en spécifiant le titre de la fenêtre
inputInt(prompt)	Ouvre une boîte de dialogue modale avec les boutons OK/Annuler. OK retourne le nombre entier entré. Un clic sur Annuler ou Fermer termine le programme.
inputInt(prompt, False)	Idem, sauf qu'un clic sur Annuler/Fermer n'interrompt pas le programme mais retourne la valeur None
inputFloat(prompt)	Ouvre une boîte de dialogue modale avec les boutons OK/Annuler. OK retourne le nombre à virgule flottante entré. Un clic sur Annuler ou Fermer termine le programme.
inputFloat(prompt, False)	Idem, sauf qu'un clic sur Annuler/Fermer n'interrompt pas le programme mais retourne la valeur None
inputString(prompt)	Ouvre une boîte de dialogue modale avec les boutons OK/Annuler. OK retourne la chaine entrée. Un clic sur Annuler ou Fermer termine le programme.
inputString(prompt, False)	Idem, sauf qu'un clic sur Annuler/Fermer n'interrompt pas le programme mais retourne la valeur None
input(prompt)	Ouvre une boîte de dialogue modale avec les boutons OK/Annuler. OK retourne le nombre entier, le nombre flottant ou la chaine de caractères saisie par l'utilisateur. Les boutons Annuler ou Fermer terminent le programme.
input(prompt, False)	Idem, sauf qu'un clic sur Annuler/Fermer n'interrompt pas le programme mais retourne la valeur None
askYesNo(prompt)	Ouvre une boîte de dialogue modale avec les boutons Oui/Non. Oui retourne True et Non retourne False. Les boutons Annuler ou Fermer terminent le programme.
askYesNo(prompt, False)	Idem, sauf qu'un clic sur Annuler/Fermer n'interrompt pas le programme mais retourne la valeur None

from fitter import *

polynomfit(xdata, ydata, n)	fits a polynom of order n and returns the fitted values in ydata. Return value: list with n + 1 polynom coefficients
splinefit(xdata, ydata, nbKnots)	fits a spline function that passes through nbKnots aequidistant data points. Returns the fitted data in ydata
functionfit(func, derivatives, initialGuess, xdata, ydata)	fits the function func(x, param) with n parameters in list param. derivatives(x, param) returns a list with the values of the partial derivatives to the n parameters. initGuess is a list with n guessed values for the n parameters
functionfit(func, derivatives, initialGuess, xdata, ydata, weights)	same but with a list weights that determines the relative weights of the data points

toAequidistant(xrawdata, yrawdata, deltax)	returns two lists xdata, ydata with aequidistant values separated by deltax (linear interpolation)

TCP Client/Server

from tcpcom import *

TCPServer

server = TCPServer (port, port, stateChanged, isVerbose = False)	crée une requête TCP pour se connecter avec un client avec le port port (entier). Les changements d'état appellent stateChanged (). Pour isVerbose = True, les messages de débogage sont écrits dans la fenêtre de sortie.
stateChanged (state, msg)	Rappel, qui est appelé lors d'un changement d'état. state: TCPServer.PORT_IN_USE, msg: port state: TCPServer.CONNECTED, msg: adresse IP du client state: TCPServer.LISTENING, msg: vide state: TCPServer.TERMINATED, msg: vide state: TCPServer.MESSAGE, msg: Message reçu du client (String)
server.isConnected ()	True si le client est connecté au serveur
server.terminate()	Clôture de la connexion, fin de l'état LISTENING. Le port IP est libéré.
sever.isTerminated ()	renvoie True si le serveur est dans l'état TERMINATED
server.isConnected ()	renvoie True si le client est connecté à un serveur
server.disconnect ()	met fin à la connexion au serveur. state=LISTENING
server.sendMessage (msg)	envoie les informations au client (string, le caractère \ 0 (ASCII 0) sert indique la fin du msg, automatiquement ajouté et supprimé de manière transparente).
TCPServer.getVersion ()	renvoie la version du module (String)

TCPClient

client = TCPClient (ipAddress, port, stateChanged, isVerbose = False)	crée une requête TCP pour se connecter avec un serveur TCP avec l'adresse IP ipAddress (String) et le port (entier). Les changements d'état appellent stateChanged (). Pour isVerbose = True, les messages de débogage sont écrits dans la fenêtre de sortie.
stateChanged (state, msg)	Rappel, qui est appelé lors d'un changement d'état. state: TCPClient.CONNECTING, msg: adresse IP du serveur state: TCPClient.CONNECTION_FAILED, msg: adresse IP du serveur state: TCPClient.CONNECTED, msg: adresse IP du serveur state: TCPClient.SERVER_OCCUPIED, msg: adresse IP du serveur state: TCPClient.DISCONNECTED, msg: vide state: TCPClient.MESSAGE, msg: message reçu du serveur (String)
client.connect ()	crée une connexion au serveur (bloquant jusqu'à expiration du délai). Renvoie True si la connexion a été établie, sinon False sera renvoyé.
client.connect (timeout)	idem, mais en spécifiant le délai d'attente (en s) pour la tentative de connexion
client.isConnecting ()	renvoie True lorsque le client tente de se connecter
client.isConnected ()	renvoie True si le client est connecté à un serveur
client.disconnect ()	met fin à la connexion au serveur
client.sendMessage (msg, responseTime)	envoie les informations au serveur (string, le caractère \ 0 (ASCII 0) sert indique la fin du msg, automatiquement ajouté et supprimé de manière

	transparente). Pour responseTime> 0, la méthode bloque et attend une réponse dans responseTime (en s). La réponse est livrée en tant que valeur de retour. Si la réponse n'arrive pas, aucun n'est retourné
TCPClient.getVersion ()	renvoie la version du module (String)

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,	Effectue un essai de connexion au serveur avec serverInfo donné. Si

timeout)	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)

HTTPServer (hérité de TCPServer)

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