**DronLink: Funciones de la clase Dron**

Se describen en la tabla siguiente las funciones de la clase Dron en la versión actual de la librería DronLink. En muchas de ellas aparecen los tres parámetros requeridos para implementar la modalidad no bloqueante (blocking, callback y params).

|  |  |
| --- | --- |
| def connect(self,  connection\_string, baud,  id=None, freq = 4,  blocking=True, callback=None, params = None) | |
|  | Conecta con el dron. Los parámetros connection\_string, baud indican si hay que conectar con el simulador o con el dron real, y la velocidad de comunicación. Los strings de conexón habituales son:  Simulador: ‘tcp:127.0.0.1:5763’ y velocidad de 115200.  Directamente al dron por radio de telemetría: ‘com??’ y velocidad 57600  Con el dron a través de mavproxy: 'udp:127.0.0.1:14551' y velocidad de 115200.  La conexión admite una identificador para el dron, que la librería añadirá como primer parámetro en todas las funcione callback. El parámetro freq indica la frecuencia con la que se van a enviar datos de telemetría. Por defecto se envían 4 paquetes de datos por segundo. Con frecuencias mayores puede conseguirse, por ejemplo, mayor fluidez en el movimiento del icono del dron sobre un mapa, aunque puede saturarse un poco el computador en el que se ejecuta la aplicación.  La conexión suele ser rápida. Por eso es habitual usar el modo bloqueante al usar esta función. |
| def disconnect (self) | |
|  | Desconecta el dron. Además, detiene el envío de datos de telemetría. |
| def arm(self, blocking=True, callback=None, params = None) | |
|  | Arma el dron. El armado es rápido. Por eso es habitual usar el modo bloqueante al usar esta función. |
| def takeOff(self,  aTargetAltitude,  blocking=True, callback=None, params = None) | |
|  | Despega el dron hasta alcanzar la altura indicada en el parámetro. |
| def changeNavSpeed (self, speed) | |
|  | Cambia la velocidad de navegación. |
| def go(self, direction) | |
|  | Hace que el dron navegue en la dirección indicada. Las opciones son: ‘North’, ‘South’, ‘West’, ‘East’, ‘NorthWest’, ‘NorthEast’, ‘SouthWest’, ‘SouthEast’, ‘Stop’, ‘Forward’, ‘Back’, ‘Left’, ‘Right’, ‘Up’, ‘Down’. |
| def getParams(self,  parameters,  blocking=True, callback=None) | |
|  | Pide al dron el valor de los parámetros indicados. En el caso de que la llamada sea o bloqueante, ejecutará la función del callback pasándole como parámetro la lista de valores recibida (y el identificador del dron como primer parámetro en el caso de que el dron haya sido identificando en el momento de la conexión). Este ejemplo muestra el formato en el que debe pasarse la lista de parámetros y el formato de la respuesta.  parameters = [  "RTL\_ALT",  "PILOT\_SPEED\_UP",  "FENCE\_ACTION" ])  result = dron.getParams(parameters)  print('Valores:',result)  El resultado podría ser:  Valores: [{'RTL\_ALT': -1.0}, {'PILOT\_SPEED\_UP': 100.0}, {'FENCE\_ACTION': 4.0}] |
| def setParams(self,  parameters,  blocking=True, callback=None, params = None) | |
|  | Establece el valor de los parámetros que se le pasan en una lista. Un ejemplo de uso es este:  parameters =[  {'ID': "FENCE\_ENABLE", 'Value': 1},  {'ID': "FENCE\_ACTION", 'Value': 4} ]) dron.setParams(parameters) |
| def RTL (self, blocking=True, callback=None, params = None) | |
|  | Ordena al dron que retorne a casa (Return to launch). |
| def Land (self, blocking=True, callback=None, params = None) | |
|  | Ordena al dron que aterrice en el punto que está sobrevolando. |
| def setGEOFence(self,  fence\_waypoints,  blocking=True, callback=None, params = None) | |
|  | Envía al dron la configuración del geofence, que consiste en la lista de las coordenadas geográficas que delimitan el polígono (al menos 3 puntos). Un ejemplo puede ser este:  fence\_waypoints =[  {'lat': 41.123, 'lon': 1.988},  {'lat': 41.234, 'lon': 1.999},  {'lat': 41.150, 'lon': 1.870}  ])  dron.setGEOFence(fence\_waypoints) |
| def getGEOFence(self,  blocking=True, callback=None, params = None) | |
|  | Retorna la lista de puntos del geofence, en el formato que se ha indicado en la función setGEOFence. Si no hay geofence retorna None. |
| def send\_telemetry\_info(self, process\_telemetry\_info) | |
|  | Pide al dron que envíe los datos de telemetría. Cuando el dron tiene un nuevo paquete de telemetría llama al callback y le pasa ese paquete (y el identificador del dron como primer parámetro en el caso de que el dron haya sido identificando en el momento de la conexión). El dron va a enviar tantos paquetes por segundo como indique el parámetro freq en el momento de la conexión. El ejemplo siguiente muestra el formato en que se reciben los datos de telemetría:  def process\_telemetry\_info(self, telemetry\_info):  print ('info: ', telemetry\_info)  El resultado podría ser:  info:{'lat':41.276,'lon':1.988, 'alt': -0.016, 'groundSpeed': 0.120, 'heading': 355.88, 'state': 'connected'}  Los posibles estados en los que puede estar el dron son: ‘connected’, ‘disconnected’, ‘arming’, ‘armed’, ‘takingOff’, ‘flying’, ‘returning’, ‘landing’. |
| def stop\_sending\_telemetry\_info(self) | |
|  | Detiene el envío de datos de telemetría. |
| def goto(self,  lat, lon, alt,  blocking=True, callback=None, params=None) | |
|  | Dirige al dron al punto geográfico indicado. |
| def change\_altitude(self,  alt,  blocking=True, callback=None, params=None) | |
|  | Cambia la altura a la que se encuentra el dron. |
| def send\_local\_telemetry\_info(self, process\_local\_telemetry\_info) | |
|  | Pide al dron que envíe los datos de telemetría local. Cuando el dron tiene un nuevo paquete de telemetría llama al callback y le pasa ese paquete (y el identificador del dron como primer parámetro en el caso de que el dron haya sido identificando en el momento de la conexión). El dron va a enviar tantos paquetes por segundo como indique el parámetro freq en el momento de la conexión. El ejemplo siguiente muestra el formato en que se reciben los datos de telemetría:  def process\_local\_telemetry\_info(self, telemetry\_info):  print ('info: ', telemetry\_info)  El resultado podría ser:  info:{‘posX’:3 ,'posY':2.5, 'posZ': -3}  El valor posX es el desplazamiento en metros en el que se encuentra el dron en la dirección del Norte, respeto al home (o en la dirección de Sur si el valor es negativo). De manera similar, posY indica el desplazamiento en la dirección Este (u Oeste para valores negativos) y posZ el desplazamiento hacia abajo (o hacia arriba para valores negativos). |
| def stop\_sending\_local\_telemetry\_info(self) | |
|  | Detiene el envío de datos de telemetría local. |
| def uploadMission(self,  mission,  blocking=True, callback=None, params = None) | |
|  | Carga en el autopiloto la misión indicada, que debe especificarse en ese formato:  {  "takeOffAlt": 5,  "waypoints":  [  {  'lat': 41.2763410,  'lon': 1.9888285,  'alt': 12  },  {  'lat': 41.27623,  'lon': 1.987,  'alt': 14  }  ]  }  El dron despegará en la posición en la que esté y al llegar al último waypoint harán un RTL. |
| def executeMission(self,  blocking=True, callback=None, params = None) | |
|  | El dron ejecuta la última misión que se haya cargado. |