csDronLink: Funciones de la clase Dron

Se describen en la tabla siguiente las funciones de la clase Dron en la versión actual de la librería csDronLink. En muchas de ellas aparecen los tres parámetros requeridos para implementar la modalidad no bloqueante (`bloquear`, `f` y `param`).

Comandos básicos

Lo básico para el ciclo más elemental: conectar, armar, despegar, aterrizar y desconectar.

`public void Conectar(string modo, string conector = null)`
Conecta con el dron. El parámetro `modo` puede ser `"simulación"` o `"producción"`. En el primer caso no se necesita `conector` y se conecta a `"tcp:127.0.0.1:5763"` con velocidad de 115200. En el caso de `"producción"` se conecta al puerto que indique `conector` y la velocidad será 57600.
`public void Despegar(int altitud, Boolean bloquear = true, Action<object> f = null, object param = null)`
Despega el dron hasta alcanzar la altura indicada en el parámetro.
`public void Aterrizar (Boolean bloquear = true, Action<object> f = null, object param = null)`
Ordena al dron que aterrice en el punto que está sobrevolando.
`public void RTL(Boolean bloquear = true, Action<object> f = null, object param = null)`
Ordena al dron que retorne a casa (Return to launch).

Navegación

Comandos para indicarle al dron hacia dónde tiene que navegar o a que punto geográfico debe ir.

`public void CambiaVelocidad(int velocidad)`
Cambia la velocidad de navegación.
`public void Navegar(string direccion)`
Hace que el dron navegue en la dirección indicada. Las opciones son: `'North'`, `'South'`, `'West'`, `'East'`, `'NorthWest'`, `'NorthEast'`, `'SouthWest'`, `'SouthEast'`.
`public void IrAlPunto(float lat, float lon, float alt, Boolean bloquear = true, Action<object> f = null, object param = null)`
Dirige al dron al punto geográfico indicado.
`public void Mover(string direccion, int distancia, Boolean bloquear = true, Action<object> f = null, object param = null)`
Mueve el dron los metros indicados por `distancia` y en la dirección indicada. Los valores de `direccion` pueden ser: `'Forward'`, `'Back'`, `'Left'`, `'Right'`, `'Up'`, `'Down'`, `'Stop'`, `'ForwardLeft'`, `'ForwardRight'`, `'BackLeft'`, `'BackRight'`.

Misiones

Comandos para cargar y ejecutar misiones. Una misión es una secuencia de waypoints que el autopiloto va a ejecutar de manera autónoma.

```
public void CargarMision(List<(float lat, float lon)> mision)
```

Carga la misión definida por la lista de waypoins que recibe. La altura de los waypoints está preestablecida en 20 metros.

```
public void EjecutarMision(Boolean bloquear = true, Action<object> EnWaypoint = null,
Action<object> f = null, object param = null)
```

El dron ejecuta la misión que se ha cargado. El dron debe estar en el aire antes de dar la orden de ejecutar la misión. Si especificamos una función en el parámetro `EnWaypoint`, la librería llamará a esa función cada vez que llegue a un waypoint pasándole como parámetro el índice del waypoint alcanzado.

Escenarios

Los escenarios incluyen un geofence de inclusión y varios geofences de exclusión que representan obstáculos en la zona de vuelo (puede no haber obstáculos).

```
public void EstableceEscenario(List<List<(float lat, float lon)>> scenario)
```

El escenario se recibe en forma de lista. En cada posición hay un fence que representan áreas.
El primer elemento de la lista es un fence de inclusión, que representa el área (polígono) de la que el dron no va a salir. El resto de elementos de la lista son fences de exclusión que representan obstáculos dentro del fence de inclusión, que el dron no puede sobrevolar. El escenario debe tener un fence de inclusión (solo uno y es el primer elemento de la lista, que debe tener al menos 3 waypoints) y un número variable de fences de exclusión, que puede ser 0.
Un fence de exclusión puede ser un polígono o un círculo. En el primer caso el fence debe tener al menos 3 waypoints. Si el fence es un círculo entonces tiene 2 waypoints. El primero identifica el centro del círculo y en la latitud del segundo waypoint está el radio en metros.

Gestión de parámetros

```
public List<float> LeerParametros(List<string> parametros)
```

Pide al dron el valor de los parámetros cuyos nombres está en el parámetro. Devuelve una lista con los valores de los parámetros en el mismo orden en el que aparecen en la lista de nombres.

```
public void EscribirParametros(List<(string parametro, float valor)>
parametros)
```

Establece el valor de los parámetros identificados en la lista de tuplas (parametro, valor).

Telemetría

En la versión actual de la librería solo se capturan los datos siguientes: latitud, longitud, altitud y heading.

`public void EnviarDatosTelemetria(Action<List<(string nombre, float valor)>> f)`
Pide al dron que envíe los datos de telemetría. Cuando el dron tiene un nuevo paquete de telemetría llama al callback `f` y le pasa ese paquete. El paquete es una lista de tuplas con (nombre, valor) de los 4 datos. Los nombres que usa son: `"Lat"`, `"Lon"`, `"Alt"` y `"Heading"`.
`public void DetenerDatosTelemetria()`
Detiene el envío de datos de telemetría.

Miscelánea

`public void PonModoGuiado()`
Pone el dron en modo "Guided".
`public void CambiarHeading(float nuevoHeading, Boolean bloquear = true, Action<object> f = null, object param = null)`
Rota el dron en sentido horario hasta orientarse según `nuevoHeading`.