UAL Fixed Wing Backend

La principal diferencia en el control del ala fija es no poder usar el modo OFFBOARD sino AUTO.MISSION. Los comandos en el mismo se camponen de waypoints de distinta naturaleza más ciertos parámetros dependiendo de cada tipo. Esto hace que el comportamiento del autopiloto sea mucho más impredecible comparado con un ala rotatoria, ya que la trayectoria trazada, pese a ajustarse a los parámetros definidos, depende mucho de la posición y actitud de la aeronave en el momento de generar la misión.

Por ello, se ha implementado un nuevo control para UAL en modo misión. Desde éste, todos los parámetros que definen cada waypoint están accesibles y las posiciones en el espacio se defininen desde PoseStamped. Además, los waypoints se han agrupado en WaypointSets, para poder definir a un grupo de wp con los mismos parámetros sin necesidad de repetirlos. Otra utilidad de los WaypointSets es capacitar a UAL de, usando parámetros auxiliares, definir la localización de algunos wp en función de la posición actual de la aeronave. Ésto se explicará más adelante.

Los medios originales de comunicación con UAL, land, take off y gotowaypoint siguen estando accesibles. No son más que una traducción de los parámetros originales que los definen a un control de misión como el definido en el apartado anterior.

Misiones en Mavros

Las misiones pueden cargarse en cualquier momento, siendo ejecutadas cuando se pase a modo AUTO.MISSION. Éstas quedan definidas por el siguiente mensaje:

|  |  |  |
| --- | --- | --- |
| **mavros\_msgs/WaypointPush.srv** | | |
| **Request** | | **Response** |
| uint16 start\_index | | bool success |
| mavros\_msgs/Waypoint[] waypoints | uint8 FRAME\_GLOBAL=0  uint8 FRAME\_LOCAL\_NED=1  uint8 FRAME\_MISSION=2  uint8 FRAME\_GLOBAL\_REL\_ALT=3  uint8 FRAME\_LOCAL\_ENU=4  uint8 frame | uint32 wp\_transfered |
| uint16 command  bool is\_current  bool autocontinue |
| float32 param1  float32 param2  float32 param3  float32 param4 |
| float64 x\_lat  float64 y\_long  float64 z\_alt |

La petición está formada por una lista de mavros wps y el campo start\_index, que toma siempre valor 0 en este backend. Por ahora, frame toma el valor 0 (global) ya que no se ha conseguido, al menos en simulación, poder volar con el resto. El campo command define el tipo de wp, ya sea land, takeoff, pass, loiter... y sus variaciones. A estos últimos va asociado el significado de cada uno de los parámetros. Los utilizados actualmente se encuentran en la siguiente tabla. Para más información sobre mapeo de parámetros, y definición de los mismos, visitar <https://mavlink.io/en/messages/common.html> buscando por número o nombre.

|  |  |  |
| --- | --- | --- |
| **Fase** | **Nombre** | **Número** |
| Take off | MAV\_CMD\_NAV\_TAKEOFF | 22 |
| Pass | MAV\_CMD\_NAV\_WAYPOINT | 16 |
| Loiter | MAV\_CMD\_NAV\_LOITER\_UNLIM | 17 |
| MAV\_CMD\_NAV\_LOITER\_TURNS | 18 |
| MAV\_CMD\_NAV\_LOITER\_TIME | 19 |
| MAV\_CMD\_NAV\_LOITER\_TO\_ALT | 31 |
| Land | MAV\_CMD\_DO\_LAND\_START | 189 |
| MAV\_CMD\_NAV\_LAND | 21 |

Se ha detectado un grupo de parámetros de Mavros que deben ser ajustados al principio de la misión. Su información se encuentra en <https://dev.px4.io/v1.9.0/en/advanced/parameter_reference.html>. Se entiende que cuando se prueben más aeronaves, nuevos escenarios y en experimentos reales, surgirá la necesidad de añadir nuevos parámetros a la lista y reajustar el valor de los ya empleados.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parámetro** | **Descripción** | **Unidad** | **Comentario** |
| NAV\_DLL\_ACT | Set data link loss failsafe mode | Int32 | Poner a 0 al inicio para poder cambiar a AUTO.MISSION. Después reajustar al valor que se decida. |
| MIS\_DIST\_1WP | Maximal horizontal distance from home to first waypoint | Float | Por ahora se ha desabilitado esta constrain ya que en simulación no hay problema de distancias. Reajustar en una prueba real. |
| MIS\_DIST\_WP | Maximal horizontal distance from home to first waypoint | Float | Idem. |

Mavros pone a disposición dos topics, /mission/reached y /mission/waypoints. El primero es la forma más rápida de conocer el últmo wp por el que ha pasado la aeronave, atendiendo al orden de wp en el que se pasó la misión. Por su parte, /mission/waypoints informa de toda la misión cargada, informando de cuál es el siguiente punto, dentro del campo is\_current del waypoint correspondiente.

Estructura de datos para el backend Mavros FW

La composición de las misiones se hace a través de una nueva estructura de datos llamada WaypointSet. Ésta está compuesta por el tipo, una lista de PoseStamped y una lista de parámetros.

|  |  |
| --- | --- |
| **uav\_abstraction\_layer/WaypointSet.msg** | |
| **Tipo** | **Campo** |
| uint8 | TAKEOFF\_POSE = 0  TAKEOFF\_AUX = 1  PASS = 2  LOITER\_UNLIMITED = 3  LOITER\_TURNS = 4  LOITER\_TIME = 5  LOITER\_HEIGHT = 6  LAND\_POSE = 7  LAND\_AUX = 8  type |
| geometry\_msgs/PoseStamped[] | posestamped\_list |
| ParamFloat[] | params |

|  |  |
| --- | --- |
| **uav\_abstraction\_layer/ParamFloat.msg** | |
| **Tipo** | **Campo** |
| string | name |
| float32 | value |

Este type no coincide exactamente con los establecidos por Mavros. Se debe a que en el backend se implementan una serie de funcionalidades que facilitan la tediosa escritura del WaypointPush. El campo posestaped\_list contiene información local de los wp con respecto a un marco de referencia cuya posición global se facilita por medio del parámetro de ROS map\_origin\_geo. Como se observa, todos los wp de un WaypointSet comparten tipo y lista de parámetros. Cada elemento de la lista de parámetros está compuesta por su nombre y un valor float.

En su mayoría, la traducción de un WaypointSet es directa a los waypoints de ROS. Para esos casos, la única lógica implementada es la comprobación de que los parámetros coinciden con los asociados a su tipo, que deben tener exactamente el mismo nombre que el utilizado por Mavros. Con los parámetros, el tipo y las coordenadas, se forma cada waypoint que formará parte de la misión más extensa.

En cuanto a los tipos auxiliares, estos utilizan la posición y orientación actuales, así como nuevos parámetros auxiliares, para generar los waypoints resultantes. Tanto TAKEOFF\_AUX como LAND\_AUX generan un punto auxiliar de forma geométrica, definido por coordenadas cilíndricas: “aux\_angle”, “aux\_distance” y “aux\_hight”. Así se define cuál sera el wp al que orientar el despegue y el punto en el que aterrizar, respectivamente.

Los waypoints resultantes de un WaypointSet son concatenados y, a su vez, los resultantes de cada WaypointSet son también concatenados para formar la misión total.

Cambios en la interfáz de UAL

La necesidad de generar misiones, aunque simplificada con los WaypointSet, necesita un nuevo servicio disponible desde el servidor de UAL. Éste se ha llamado SetMission y tiene la siguiente estructura:

|  |  |
| --- | --- |
| **uav\_abstraction\_layer/SetMission.srv** | |
| **Request** | **Response** |
| WaypointSet[] waypoint\_sets  bool blocking | bool success |

Desde este servicio, todo lo explicado en el apartado anterior está accesible, definido en el campo waypoint\_sets. Además, los servicios, ya existentes, land, take off y gotowaypoint siguen estando disponibles. Estos han sido modificados atendiendo a las necesidades del control por misión. Por ello, los datos provistos en el servicio y la posición actual son empleados para generar una serie de WaypointSet de forma interna. Es importnate destacar aquí que estas funcionalidades aumentan el riesgo de colisiones y deben ser readaptadas en futuros experimentos.

A modo de ejemplo, al solicitar el servicio de takeoff, se empleará la orientación actual, se trazará un punto auxiliar a cierta distancia y altura definidas internamente (con valores estimados que, por ahora, no responden a criterios seguros ni específicos del modelo), para posteriormente dejar la aeronave haciendo loitering sobre el punto de despegue a la altura solicitada.

Pruebas

roslaunch ual\_backend\_mavros\_fw simulation.launch

rosrun ual\_backend\_mavros\_fw fw\_trials.py