Extensión de la implementación JAUS++ del estándar de comunicaciones JAUS para la interoperabilidad entre sistemas no tripulados

Cinthia Jazmin toala Zamora (1281680)

Resumen—La realización de este trabajo tuvo como objetivo desarrollar una extensión (JAUS Manipulator Service Set) de la implementación JAUS++ del estándar JAUS, que se utiliza para dotar de interoperabilidad a cualquier tipo de sistemas no tripulados, independientemente de la tecnología empleada en ellos, permitiendo así, la heterogeneidad en la comunicación entre diversos frameworks. El motivo por el que se creó esta extensión fue para ofrecer interoperabilidad a los manipuladores de uno de los sistemas no tripulados (vehículo terrestre) del proyecto europeo fp7 ICARUS, dicho proyecto tiene como objetivo dar soporte a los equipos de búsqueda y rescate de supervivientes en desastres naturales, utilizando una tropa heterogénea de vehículos no tripulados que operan bajo los frameworks ROS, MOOS y FINROC. Para dar una explicación detallada del trabajo, este artículo incluye una descripción de la arquitectura del estándar JAUS, ya que consiste en extender la librería de la implementación de dicho estándar, a continuación se explicará el objetivo y las tareas que se han desarrollado para conseguirlo. Se enseñará la metodología que se ha utilizado para cumplir con la planificación establecida, como así también, una conclusión del trabajo realizado. Por ultimo mostraremos las fuentes que se utilizó a lo largo del proyecto.

Paraules clau— Jaus++, Active-ist, FP7-Icarus, Jaus, sitemas no tripulados, interoperabilidad.

Abstract—The realization of this work had as goal to develop an extension (JAUS Manipulator Service Set) of JAUS ++ implementation of the JAUS standard, which it is used to provide interoperability between any types of unmanned systems, independently of the technology employed in them, allowing thus, heterogeneity in communication between various frameworks. The reason for this extension was created was to provide interoperability to the manipulator of one of unmanned systems (land vehicle) of the European project fp7 ICARUS, this project aims to support the teams of search and rescue of survivors natural disasters, using a heterogeneous troop of unmanned vehicles, operating under the frameworks ROS, MOOS y FINROC. To give a detailed explanation of the work, this article includes a description of the architecture of JAUS standard, as this implementation extends, then will be explained the goal and tasks that have been developed to achieve it. The methodology used to comply with the established plan, as well as a conclusion of the work performed will be explained. Finally we show the fonts that are used throughout the project.

Index Terms— Jaus++, Active-ist, FP7-Icarus, Jaus, unmanned systems, interoperability.

1 Introducción

ESTE trabajo final de grado fue propuesto por el centro tecnológico ASCAMM [5], una organización privada sin fines de lucro, especializada en I+D (investigación y desarrollo) de sistemas inteligentes para tecnologías industriales. ASCAMM tiene en marcha proyectos tanto a nivel nacional como a nivel internacional y con este trabajo se dio soporte a unos de sus proyecto a nivel internacional, concretamente al proyecto europeo fp7 ICARUS [10], en el cual trabaja su departamento de sistemas no tripulados.

El proyecto ICARUS nació después de los terremotos en l'Aquila, Haití y Japón, la Comisión Europea confirmó que existe una gran disconformidad en la tecnología (robótica) que se utilizó en el terreno de búsqueda y salvamento. De este modo, la Dirección General de la Comisión Europea de Empresa e Industria decidió financiar ICARUS, un proyecto de investigación (con presupuesto mundial de 17,5 millones €), que tiene como objetivo desarrollar herramientas robóticas que puedan ayudar a

- *E-mail de contacto: cinthiajazmin.toala@e-campus.uab.cat*
- *Mención realitzada: Ingeniería de Computadores*
- *Trabajo tutorizado por: Tomàs Manuel Margalef Burrull (Arquitectura de Computadores y Sistemas Operativos).*
  Germán Moreno Martínez (Fundació privada ASCAMM).
- *Curso 2014/15*

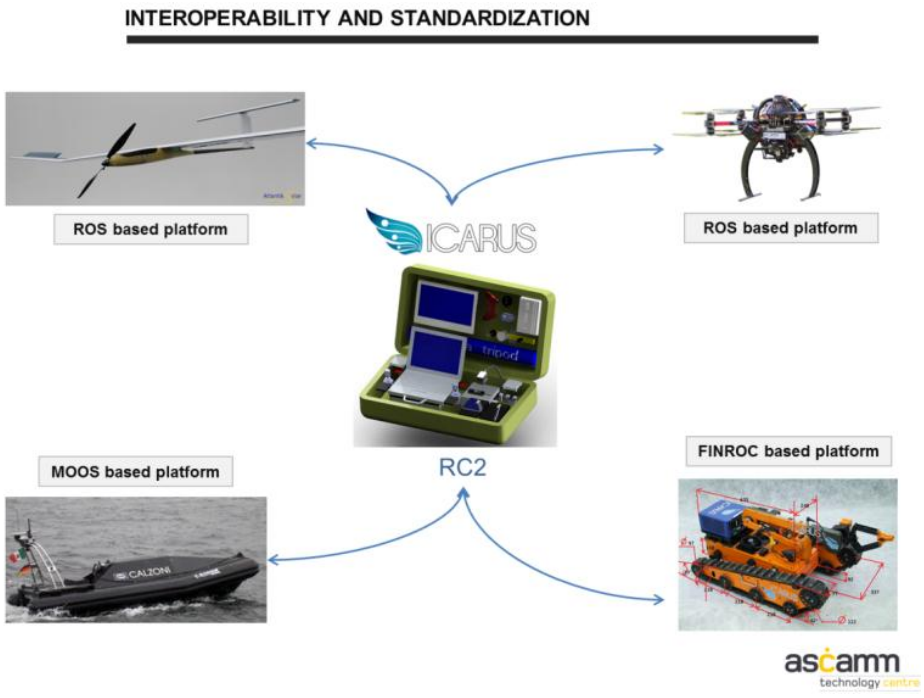

Figura_1. Ejemplo de interoperabilidad y estandarización entre distintos tipos de robots [8].

los equipos de intervención en estas crisis, para conseguir detectar, localizar y rescatar a seres humanos con dichas herramientas. Estos sistemas no tripulados minimizan los problemas que surgen en estos desastres y reducen el grado elevado de peligrosidad que hay y los costes de materiales de dicho trabajo.

Una de las cosas que necesitaba este proyecto era extender la implementación que utilizan para la interoperabilidad entre sus vehículos no tripulados. La extensión (JAUS Manipulator Service Set) que se añadirá en la implementación JAUS++ [7] será aplicada para dotar de interoperabilidad a los manipuladores del robot terrestre, dicho robot opera bajo la plataforma FINROC y podemos apreciarlo en la Figura_1.

2 JAUS++

Hoy en día aparecen muchas tecnologías nuevas constantemente, e inicialmente todas son independientes, por ejemplo en el caso de los sistemas no tripulados hay mucha diversidad de plataformas para poder manipular e interactuar dichos sistemas, el problema surge cuando queremos interactuar a la vez varios sistemas con diferentes frameworks, es aquí donde nace la necesidad de estandarizar y por eso se creó JAUS y la implementación JAUS++ de este estándar.

JAUS++ es una implementación de código abierto escrito en C++ [4] del estándar JAUS [6]. Sus siglas significan Joint Architecture for Unmanned Systems, como su nombre lo indica es una arquitectura para el dominio de los sistemas no tripulados. Esta arquitectura se utiliza para la comunicación con cualquier tipo de vehículos no tripulados reales y/o simulados, independiente de la tecnología empleada en ellos garantizando heterogeneidad.

2.1 Librería

Como podemos observar en la Figura_2 hoy en día JAUS++ que es de software libre, tiene implementado los siguientes conjuntos de servicios: El Core, el Mobility, el Transport y el Enviroment Sensing, pero carece del Manipulator y del Mission Spooling respecto al estándar JAUS que es de software privativo.

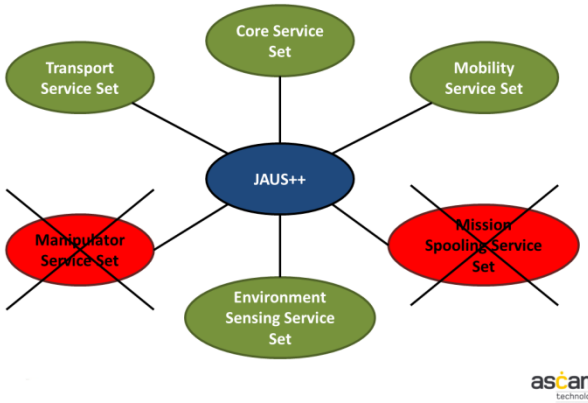

Figura_2. Comparación de la implementación JAUS++ con el estándar JAUS [9].

Estas librerías proporcionan los siguientes servicios donde se especifica una capa de comunicaciones de datos para el transporte de mensajes definidos JAUS++:

Core Service Set: Representan la infraestructura que se encuentra comúnmente en todos los dominios y tipos de sistemas no tripulados. En la actualidad, siete servicios se definen en el Core:

- Transport Service: Abstrae la funcionalidad de la capa de transporte de comunicación subyacente.
- Events Service: Establece una publicación/suscripción, mecanismo para la mensajería automática.
- Access Control: Gestiona el control exclusivo apropiable para las operaciones críticas de seguridad.
- Management: Define la gestión del ciclo de vida de los componentes.
- Time: Permite a los clientes consultar y establecer la hora del sistema para los componentes.
- Liveness: Proporciona un medio para mantener activa la conexión entre los componentes que se comunican.
- Discovery: Proporciona el descubrimiento automático de las entidades remotas (subsistemas tales como la unidad de control o los sistemas no tripulados) y sus capacidades.

Transport Service Set: Especificación de transporte define los formatos y protocolos utilizados para la comunicación entre las entidades que cumplen los protocolos de capa de enlace y medios de comunicación (JUDP).

Mobility Service Set: Proporciona los medios para comunicar y coordinar las actividades de un sistema no tripulado o sistema de sistemas no tripulados. Los servicios de movilidad representan las capacidades independientes de la plataforma que se encuentran comúnmente en todos los dominios y tipos de sistemas no tripulados. En la actualidad, 15 servicios se definen en Mobility, como por ejemplo:

- Pose sensors: Para determinar la posición instantánea y la orientación de una plataforma en coordenadas globales o locales.
- Velocity state sensor: Para determinar la velocidad instantánea de una plataforma.
- Acceleration state sensor: Para determinar la aceleración instantánea de una plataforma.
- Primitive driver: Para realizar la movilidad básica de una plataforma basada en la fuerza de torsión.
- Vector drivers: Para realizar la movilidad de circuito cerrado para el recorrido en línea recta.
- Velocity state driver: Similar a vector drivers, pero con grados de libertad adicionales.
- Waypoint drivers: Realizar la movilidad de circuito cerrado en una ubicación especificada.
- Path segment drivers: Realiza la movilidad de circuito cerrado a lo largo de una ruta especificada.

Environment Sensing Service Set: Los Servicios de detección de entorno representan capacidades de detección de ambientes que se encuentran comúnmente en todos los dominios y tipos de sistemas no tripulados de forma independiente de la plataforma. Son cinco los servicios que lo definen:

- Range sensor: Determinar la proximidad de los objetos en el entorno de la plataforma.
- Visual sensor: Proporciona la configuración común para los diferentes tipos de sistemas de imágenes.
- Digital video: Un tipo de sensor visual que logra vídeo digital.
- Analog video: Un tipo de sensor visual que logra vídeo analógico.
- Still image: Un tipo de sensor visual que gestiona y codifica las imágenes digitales individuales.

Mission Spooling Service Set: Cola de misión de una plataforma independiente. En la actualidad, 1 de servicio se define Spooling Services (Solo existe en JAUS y se planean más servicios en un futuro):

- Mission Spooler: Para planes de misión, coordina los planes de la misión.

Manipulator Service Set: Los manipuladores se utilizan a menudo en sistemas no tripulados para alterar el medio ambiente, para tomar el mando y el control de estos hay 19 servicios definidos en Manipulator service set.

Adelantándonos a los objetivos, lo que haremos es desarrollar servicios de este tipo para dotar de interoperabilidad a cualquier manipulador en sistemas no tripulados. Se realizaran los siguientes 4 servicios de los 19 que hay:

- EndEffectorPoseSensor: para reportar la posición y orientación del end effector del manipulador cuando se le pregunta.
- Joint position sensor: para reportar los valores de posición y orientación de las articulaciones del manipulador cuando se les pregunta.
- End effector pose driver: para realizar el control de la posición y la orientación del end effector del manbipulador.
- Joint position driver: es llevar a cabo el control de la posición y orientación conjunta del manipulador.

2.2 Arquitectura

Ahora pasaremos a explicar resumidamente el funcionamiento y la arquitectura de este sistema, para que se pueda entender el trabajo que se ha realizado.

Como vemos en la Figura_3 este sistema es un conjunto de subsistemas, dicho subsistema podría ser un vehículo no tripulado (UV) como por ejemplo un robot o una unidad de control de operación (OCU).

Cada subsistema está dotado de nodos, que es cualquier dispositivo informático, este nodo tiene una dirección física. Como por ejemplo en el caso de un robot, un nodo podría ser su microcontrolador o un manipulador.

Estos nodos están formados por componentes, cada componente tiene una colección de servicios básicos ya añadidos por defecto, que son los del Core Service Set que hemos mencionado antes en el apartado de característica, o añadidos como el Manipulator Service Set, el cual añadiríamos si nuestro robot utilizase un manipulador, (co-

mo en el caso de robot terrestre de la plataforma FINROC de la Figura_1). Estos componentes están identificados con una ID individual para poder acceder a toda la información añadida a ellos de forma directa, identificable y segura.

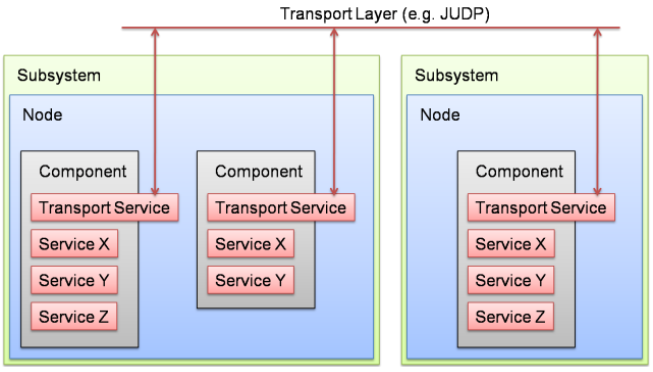

Figura_ 3. Arquitectura del sistema JAUS [1]

2.2 Características

JAUS++ es totalmente orientado a objetos por lo que permite crear componente JAUS con todos los servicios básicos que hay en el Core (Transport (JUDP), Control, Discovery, Events, Liveness, Time; Management) en pocas líneas de código.

Tiene librerías completas de mensajes tanto para el núcleo (core) donde están los servicios básicos, como para otros servicios, tales como el de movilidad (mobility).

JAUS ++ también incluye algunos servicios personalizados y experimentales que no forman parte de las normas JAUS. Sus creadores ampliaron el diseño para facilitar la transmisión de video, la utilización de joystick o interactuar con el microcontrolador.

3 OBJETIVO

Como hemos estado adelantando el objetivo de este trabajo es extender la implementación JAUS++ y hacer algunos servicios del Manipulator Service Set para poder dotar de interoperabilidad a los manipuladores del robot terrestre del proyecto FP7 Icarus. Para ello se desarrollará los siguientes servicios:

- Dos servicios de sensores:
  ManipulatorEndEffectorPoseSensor.
  ManipulatorJointPositionSensor.

- Dos servicios de posición de los drivers:
  ManipulatorEndEffectorPoseDriver.
  ManipulatorJointPositionDriver.

Lo que se proporciona con estos servicios de sensores y drivers es un sistema de coordenadas en la que especificamos la posición y orientación de cada parte del manipulador; Este sistema suele estar unido a la base del vehículo no tripulado. Podemos observamos un ejemplo del sistema en la Figura_4.

En cada servicio de JAUS++ se establecen 3 tipos de

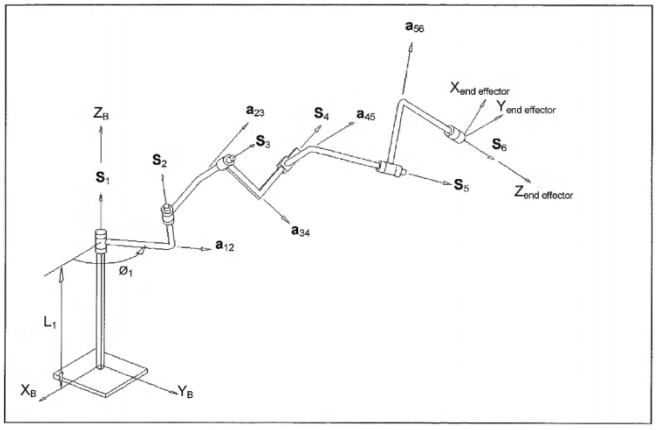

Figura_4. Ejemplo del sistema de coordenadas de un manipulador [2].

mensajes que son Queries, Reports y Sets para poder operar e interactuar entre ellos. Por ejemplo en la Tabla_1 podemos observar los mensajes que tendrá Manipulator Joint Position Driver.

Message Id (hex)	Name	isCommand?
Input Set		
0602h	*SetJointPosition*	True
0607h	*SetJointMotionProfile*	True
2600h	QueryManipulatorSpecifications	False
2607h	QueryJointMotionProfile	False
2608h	QueryCommandedJoint Position	False
Output Set		
4600h	ReportManipulatorSpecifications	False
4607h	ReportJointMotionProfile	False
4608h	ReportCommandedJoint Position	False

Tabla_1. Manipulator Join Position Driver Service [3]

4 METODOLOGÍA

Para poder organizar el trabajo que se tenia que realizar, primeramente utilizamos la metodología Work Breakdown Structure (WBS) [11] para dividir el trabajo en partes. WBS es una estructura de descomposición de trabajo (EDT) en paquetes entregables, su descomposición jerárquica orientada a entregar proyectos en componentes más pequeños, facilitó la estructuración y definición de la carga de trabajo.

Como cada servicio era independiente, se organizó el trabajo en 4 paquetes, uno por cada servicio, para conseguir el alcance total del proyecto. En la Figura_9, Figura_10 y Figura_11 del apéndice podemos observar la estructuración y la descomposición de cada uno de ellos.

Una vez hecha la estructuración del trabajo, pasamos a realizar la planificación del tiempo, para poder tener un orden y control de los días con los que se contaba para finalizar el proyecto, en dicho proyecto ponemos identificar 3 fases, la fase de investigación y aprendizaje, la fase de desarrollo y testeo y por último la fase de documentación.

La planificación que se hizo tenía 7 etapas que se realizaban en 18 semanas, como podemos observar en la Tabla_2. Cada etapa tenía varias actividades a realizar, dichas actividades realizadas se expondrán a continuación brevemente.

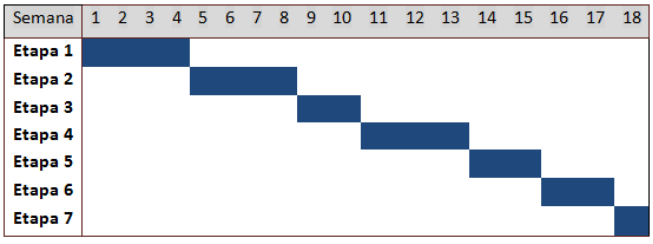

Tabla_2: Planificación semanal del trabajo.

Etapa 1: Esta etapa fue dedicada expresamente para la fase de investigación y aprendizaje, así que en las primeras 4 semanas que corresponde del 9 de febrero al 8 de marzo, se familiarizó con la implementación JAUS++, se instalado el programa y se leyó toda la documentación para entender dicha arquitectura de comunicación. Por último se procedió hacer los 8 tutoriales de la página web de JAUS++.

Etapa 2 y etapa 3: En estas etapas que corresponde del 9 de marzo al 19 de abril se inició la fase de desarrollo, primeramente se implementó Manipulator End Effector Pose Sensor y sus mensajes. Tal y como habíamos mencionado anteriormente, los servicios son independientes, así que también se hizo pruebas del servicio una vez acabado, con simulaciones para garantizar de su buen funcionamiento. Seguidamente se procedió hacer lo mismo con Manipulator Joint Position Sensor.

Etapa 4 y etapa5: Una vez hecho la parte de sensores, se desarrollaron los drivers (del 20 de abril al 24 de mayo), empezando por el Manipulator End Effector Pose Driver y después con Manipulator Joint Position Driver, al igual que los sensores también se hizo las pruebas correspondientes para verificar su buen funcionamiento.

Etapa 6: Una vez acabado todo el código se comprobó detalladamente, que la extensión realizada se integraba a la perfección a JAUS++.

Etapa 7: La última etapa se utilizó para hacer la última fase de todo proyecto, la fase de documentación y presentación del trabajo realizado.

5 Desarrollo y resultados

En esta sección se ofrecerá una visión clara y breve del trabajo hecho, junto con los resultados obtenidos.

Primeramente pasaremos a definir el entorno donde opera estos sistemas no tripulados, que se especifican con sistemas de coordenadas. Para entender mejor este apartado se aconseja ver la Figura_4 a medida que se va leyendo.

Sistema de coordenadas global: Este sistema de coordenadas se definen en unidades de longitud, latitud y altitud, para saber la posición global de la plataforma donde se interactúa.

Sistema de coordenadas del Vehículo: Este sistema de coordenadas se definen en unidades de longitud, latitud y altitud, para saber la posición global del subsistema (vehículo no tripulado), según la Figura_4 es X, Y y Z de la base.

Sistema de coordenadas del manipulador: En la mayoría de los casos, este sistema de coordenadas está unido a la base del vehículo y las coordenadas se definen igual que antes. El origen de estas coordenadas se encuentra en la intersección del primer eje de la articulación S1 y acaba en la última articulación (lista de posición de cada articulación).

Sistema de Coordenadas del efector final: Este sistema de coordenadas está unido al último eslabón del manipulador en serie. El punto de origen está situado a una distancia Sn (S6 para un manipulador de seis ejes por ejemplo).

Con estos sistemas de coordenadas se podrá tomar el mando, el control y la supervisión de los componentes de un manipulador, que está compuesto de indiferentes número de articulaciones.

Una vez explicado los sistemas de coordenadas que tienen estos vehículos no tripulados, para poder saber la posición exacta de cada componente pasamos a decir los tipos de mensajes que hay en el sistema de comunicación JAUS++. Hay 3 tipòs de mensajes que son:

Query: Este tipo de mensajes sirve para lanzar consulta del servicio que estemos interesados.
Report: Estos mensajes sirven para poder recibir información del servicio consultado.
Set: Este mensaje sirve para poder comandar cuando se necesita controlar el servicio.

En la siguiente Figura_5 se puede ver un ejemplo de esquema de secuencia de mensajes del servicio de discovery del core.

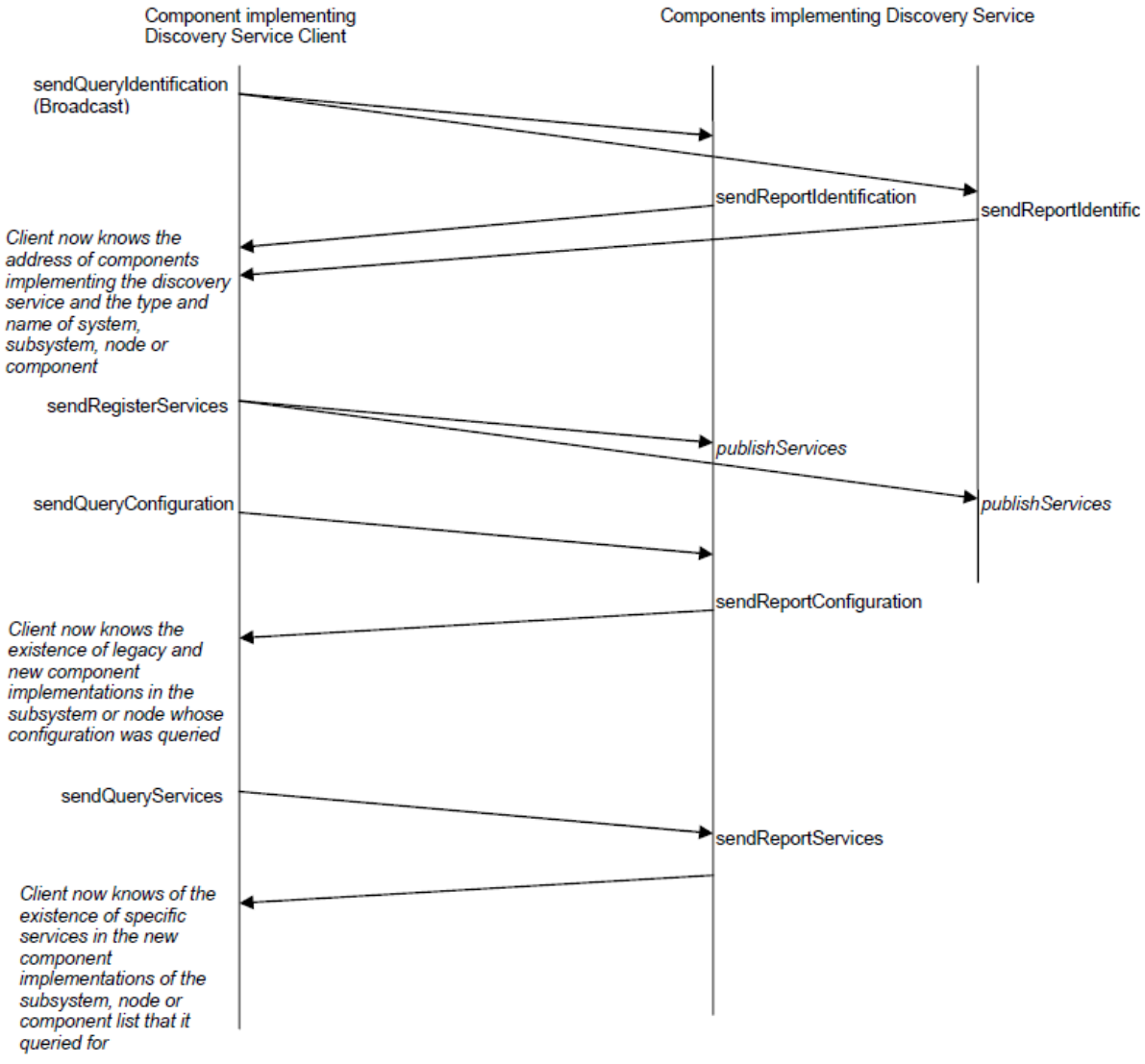

Figura_5: Diagrama de la secuencia de mensajes del servicio discovery

Cada mensaje se crea en una clase de C++, dichas clases tienen funciones implementadas tales como: Crear el mensaje, establecer valores según el estado del componente, Pintar por pantalla el report del servicio, mostrar y establecer los valores para cada variable del servicio, especificar si se puede generar eventos del servicio o si se puede controlar el servicio, etc. Estas funciones se crean dependiendo de lo que necesitemos de cada servicio y siempre se pueden ir ampliando.

Ahora que ya sabemos las definiciones necesarias para entender el trabajo que se ha desarrollado pasamos a exponer los 4 servicios implementados.

- **Manipulator End-Effector Pose Sensor**

  En este servicio se ha desarrollado 4 tipos de mensajes:
  - Query Manipulator Specifications
  - Query End Effector Pose
  - Report Manipulator Specifications
  - Report End Effector Pose

  Los mensajes de query se han desarrollado para generar consultas de la posición relativa y la orientación de la base del manipulador, con respecto al sistema de coordenadas del efector final.

  Este servicio se refiere únicamente a la operación remota de un manipulador. Los outputs que se generan son los report respectivos de cada query.

- **Manipulator Joint Position Sensor**

  Los mensajes asociados aquí son:
  - Query Manipulator Specifications
  - Query Joint Positions
  - Report Manipulator Specifications
  - Report Joint Positions.

El Report Joint Positions message proporciona las posiciones conjuntas del manipulador. Las posiciones se dan en grados para de revolución y en metros para prismáticos. El componente se representa en la Figura_6.

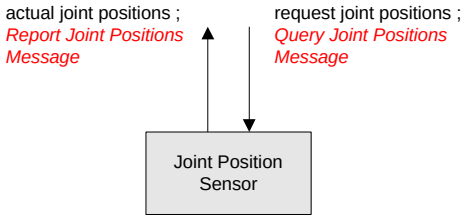

Figura_6: Componente Joint Position Sensor.

- **Manipulator End-Effector Driver**

  Algunos de los mensajes realizados en este servicio son (se puede observar la tabla completa en la Figura_11 del apéndice):
  - Set Tool Offset
  - Set Joint Motion Profile
  - Set End Effector Pose
  - Query Manipulator Specifications
  - Query Tool offset
  - Query Joint Motion Profile
  - Report Manipulator Specifications
  - Report Tool offset
  - Report Joint Motion Profile

Este componente realiza el control de la posición y orientación del efector final. La entrada es la posición deseada y la orientación del efector final, especificado en el sistema de coordenadas del vehículo, los ángulos de las articulaciones actuales, y los datos del Report Manipulator Specifications. El componente se representa en la Figura_8.

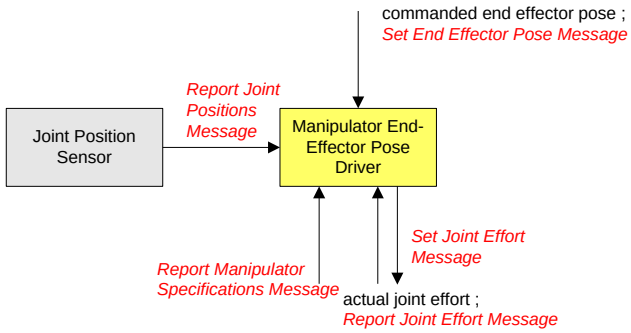

Figura_7: Manipulator End-Effector Pose Driver

- **Manipulator Joint Positions Driver**

  Algunos de los mensajes asociados aquí son (se puede observar la tabla completa en la Figura_11 del apéndice):
  - Set Joint Positions
  - Report Manipulator Specifications
  - Reports Joint Positions

La Figura_ representa el componente de joint position driver.

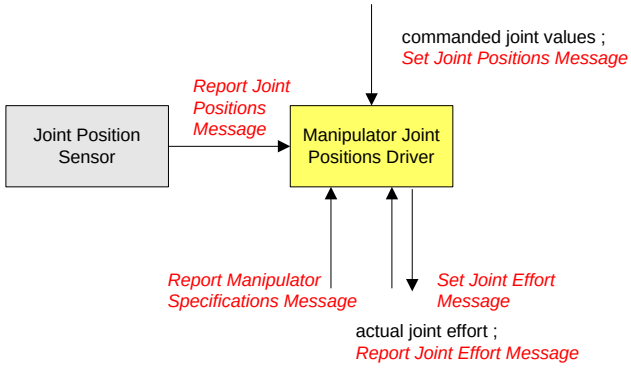

Figura_8: Componente Joint Position Driver.

6 Conclusión

Como habíamos dicho inicialmente el objetivo que nos habíamos propuesto era desarrollar una extensión de la implementación JAUS++ basada en el estándar JAUS. Dicha extensión es JAUS Manipulator Service Set, con la cual conseguimos dotar de interoperabilidad a los manipuladores del vehículo terrestre del proyecto fp7 ICARUS.

Los servicios que se ha implementó finalmente han sido los dos servicios de sensores: Manipulator End Effector Pose Sensor y Manipulator Joint Position Sensor. Y los dos servicios de drivers: Manipulator End Effector Pose Driver y Manipulator Joint Position Driver.

Finalmente se concluye el trabajo satisfactoriamente ya que se ha realizado los objetivos marcados y se ha cumplido con la planificación inicial propuesta.

Agradecimientos

En primer lugar quería dedicarles toda mi trayectoria de aprendizaje a mi madre Violeta y a mi abuela Violeta, que siempre han estado a mi lado y han respetado cada una de mis decisiones.

De forma general quisiera agradecer a todo el profesorado del grado de ingeniería informática, ya que a día de hoy puedo afirmar que los conocimientos adquiridos en el grado, me ayudarán a solventar cualquier problema en el futuro como hasta ahora.

A mi tutor de este trabajo en la UAB Tomàs Margalef, quiero agradecerle la dedicación y el apoyo que me ha dado durante estos meses. Sus consejos y experiencia me han guiado en la preparación de este.

Mis más sinceros agradecimientos a Ana Cortés, que me puso en contacto con la corporación Ascamm, para hacer las prácticas en empresa, sin ella no hubiese sido posible todo esto.

Le agradezco también a mi tutor en Ascamm, German Moreno, por el tiempo que me dedicó a lo largo de mi trayectoria en dicha empresa. No sólo me ha guiado en la elaboración del trabajo final de grado, sino que también me ha enseñado todo lo necesario por si en un futuro quiero realizar trabajos similares.

Agradezco a Ascamm por haberme dado la oportunidad de trabajar con ellos.

Por último y no menos importantes a mis compañeros Carles y Noely, que me han animado en todo el trayecto de la universidad.

Bibliografía

[1] ACTIVE-IST. (2013). Jaus++ [Figura_3]. Recuperado de http://active-ist.sourceforge.net/jaus++.php?menu=jaus

[2] ACTIVE-IST. (2013). Jaus++ [Figura_4]. Recuperado de http://active-ist.sourceforge.net/jaus++.php?menu=jaus

[3] ACTIVE-IST. (2013). Jaus++ [Tabla_1]. Recuperado de http://active-ist.sourceforge.net/jaus++.php?menu=jaus

[4] C++. (n.d.). Retrieved March, 2015, from http://www.cplusplus.com/

[5] Fundación Privada Ascamm. (n.d.). Retrieved March, 2015, from http://www.ascamm.com/

[6] Jaus. (n.d.). Retrieved March, 2015, from http://openjaus.com/doc41/namespaces.html

[7] Jaus ++. (n.d.). Retrieved March, 2015, from http://active-ist.sourceforge.net/jaus++.php?menu=jaus

[8] Moreno, G. (2014). Unmanned Systems. [Figura_1].

[9] Moreno, G. (2014). Unmanned Systems. [Figura_2].

[10] Proyecto Icarus. (n.d.). Retrieved March, 2015, from http://www.fp7-icarus.eu/Etc.

[11] Work Breakdown Structure (WBS). (n.d.). Retrieved March, 2015, from https://www.workbreakdownstructure.com/

Apéndice

A1. Definiciones y abreviaciones

Interoperabilidad: Es una habilidad de dos o más sistemas o componentes para intercambiar información o utilizar la información intercambiada.

Unmanned Systems: Son vehículos que se manipula sin tripulación.

Icarus: Es un proyecto Europeo cuyo objetivo es dar asistencia a los equipos de búsqueda y rescate utilizando una tropa heterogénea de vehículos no tripulados.

Jaus: Joint Architecture for Unmanned Systems es una arquitectura abierta para el dominio de los sistemas no tripulados que garantiza heterogeneidad.

Jaus++: Joint Architecture for Unmanned Systems ++ es una implementación de JAUS.

Ros: Robot Operating System es un framework para el desarrollo de software para robots de código abierto.

MOOS: Cross Platform Software for Robotics Research es un framework para el desarrollo de software para robots de código abierto.

FINROC: Framework for Intelligent Robot Control es un framework para el desarrollo de software para robots de código abierto.

WSB: Work Breakdown Structure es un tipo de metodología para llevar a cabo mediano o grandes proyecto, consiste en una estructura de descomposición de trabajo (EDT).

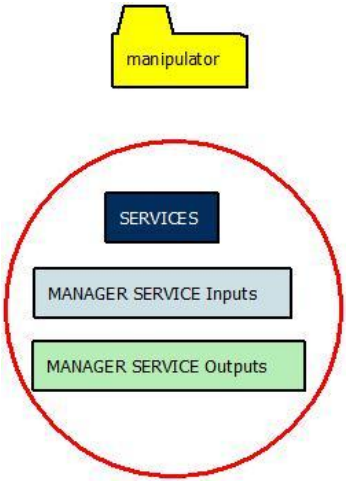

Figura_9: Clasificasción de clases de los servicios dentro de las carpetas.

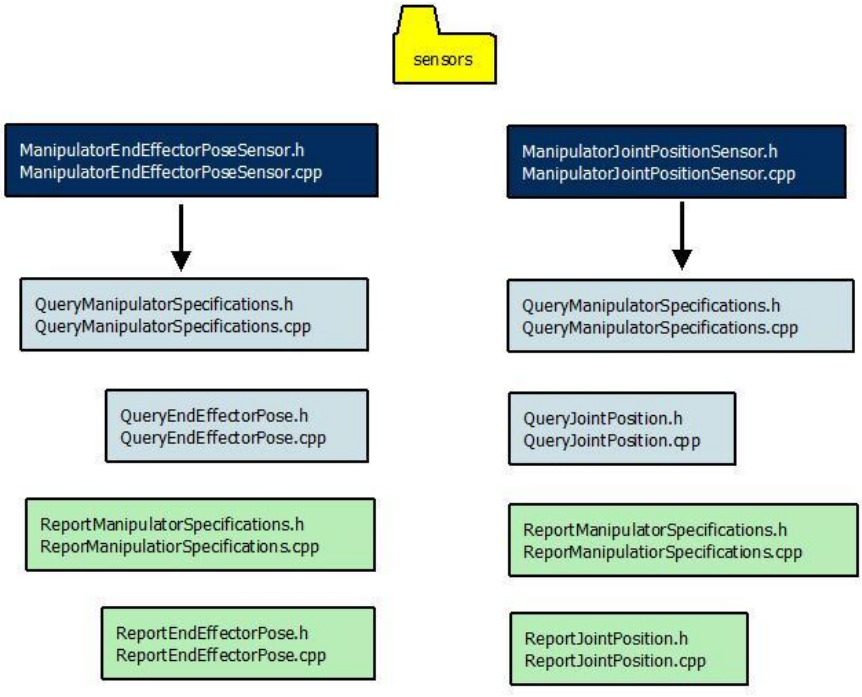

Figura_10: Estructura interna (clases) de los sensores.

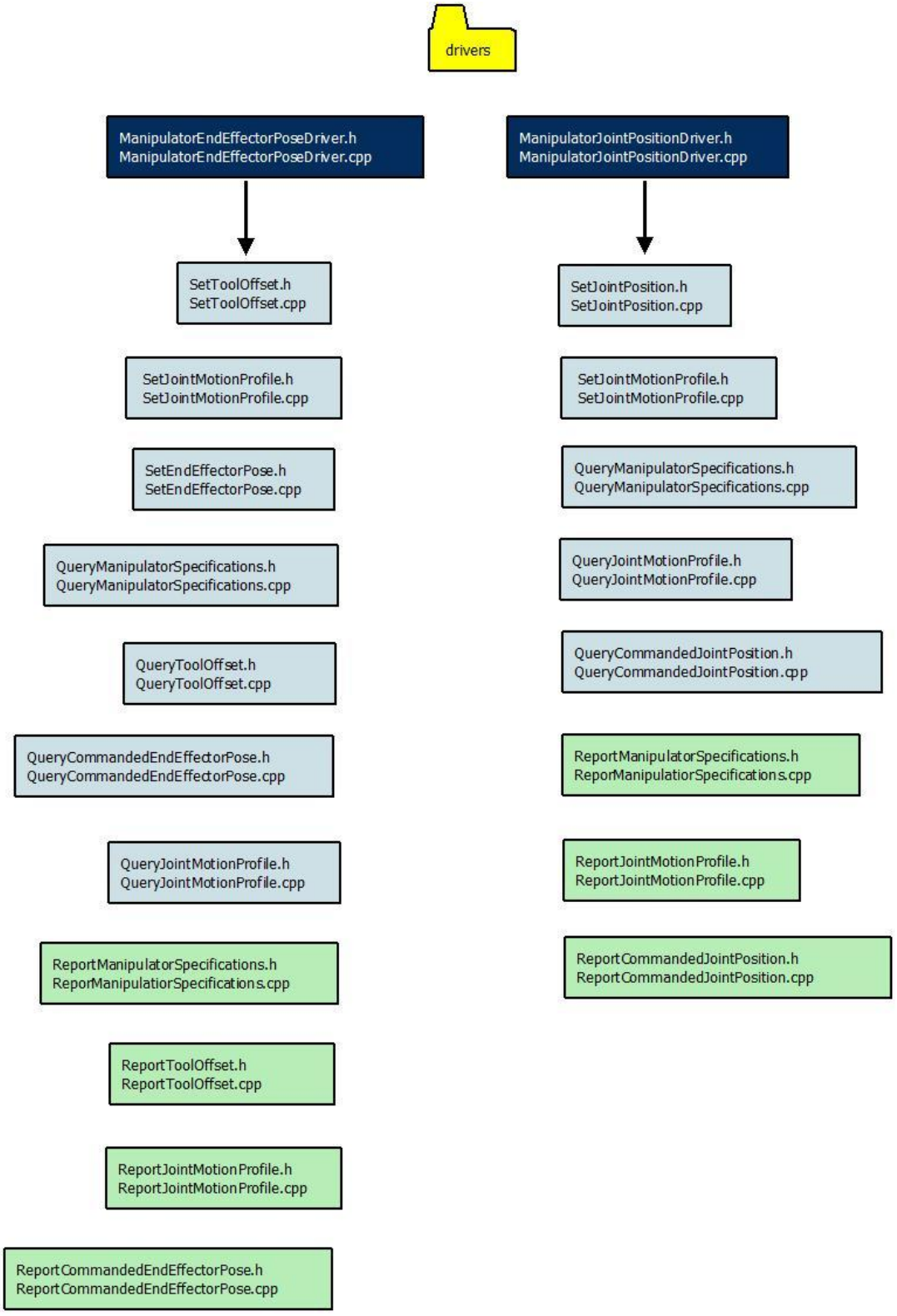

Figura_11: Estructura interna (clases) de los drivers.