Revista Iberoamericana de Ingeniería Mecánica. Vol. 27, N.º 2, pp. 49-60, 2023

MODELADO MULTIPLATAFORMA PARA EL CONTROL DE UN SISTEMA ARTICULADO

JAVIER IBARROLA CHAMIZO[1], MIKEL MERINO OLAGÜE[1], XABIER IRIARTE GOÑI[1,2], MIKEL HUALDE OTAMENDI[1]

[1]Dpto. de Ingeniería, Universidad Pública de Navarra, Campus Arrosadia, 31006, Pamplona, España
[2]Institute of Smart Cities, Universidad Pública de Navarra, Campus Arrosadia, 31006, Pamplona, España

(Recibido 10 de agosto de 2023, para publicación 28 de septiembre de 2023)

Resumen – En el presente artículo se ha trabajado con las herramientas de simulación de sistemas mecánicos Gazebo y Simulink, realizando una interconexión entre ambos con el fin de desglosar la parte dinámica y cinemática de la parte de control, obtención y tratamiento de datos. La vía de enlace entre ambos programas se realiza mediante otro software de comunicación, ROS. Con todo ello, se ha conseguido la cosimulación de un mecanismo articulado en la cual, Simulink comanda trayectorias a Gazebo a través de pares en articulaciones, Gazebo envía los valores de posición articular de la simulación a Simulink, y mediante el control multiarticular de pares realimentados diseñado en este último programa, se minimiza el error entre la trayectoria simulada y la ideal. La obtención de los parámetros del modelo dinámico requeridos en el control se ha realizado mediante una librería de Matlab (Lib_3D_MEC_Matlab) de análisis de sistemas multicuerpo desarrollada por la UPNA.

Palabras clave – Robótica, cosimulación, diseño de control, Gazebo, ROS, Simulink.

1. INTRODUCCIÓN

Hoy en día se está viviendo una nueva revolución industrial, un cambio que implica la integración de nuevas tecnologías para el desarrollo de herramientas cada vez más sofisticadas. Esta era digital, también llamada industria inteligente o Industria 4.0 aúna muchas ramas diversas de la ingeniería tales como la informática, electrónica, telecomunicaciones, robótica, mecánica, análisis y almacenamiento de datos, etc.

Una parte de esta Industria 4.0 promueve la integración de inteligencia artificial en robots para aplicaciones industriales, cuya finalidad es el desempeño de tareas de manera automatizada. Para ello es necesario el desarrollo de algoritmos sofisticados los cuales deben ser entrenados por una cantidad inmensa de datos. Esta información puede proceder de experimentos reales mediante equipos de adquisición o de simulaciones virtuales a través de herramientas computacionales.

Es aquí donde nace la necesidad de generar datos de manera masiva, en donde las herramientas computacionales (cada vez más potentes) toman ventaja sobre el campo experimental, pudiendo recrear en un entorno virtual espacios de trabajo parecidos a la realidad o situaciones que no se puedan dar en ella. Otra de las grandes ventajas es el ahorro de generar dichos datos, tanto en costes económicos como de tiempo.

Existen programas de simulación centrados en aplicaciones robóticas como: *CoppeliaSim, FreeCAD, Open Roberta Lab…* Con ellos se pueden realizar simulaciones para la obtención de datos, y con esta información desarrollar y entrenar algoritmos de IA e integrarlo en el control de un robot real.

Serena Ivaldi [1] estructuró estas herramientas de simulación en función de su alcance, dividiendo los llamados motores de físicas y los simuladores de sistemas. También realizó un estudio sobre los programas de simulación más conocidos, en el cual se clasificaban según criterios de: uso actual, uso en el pasado, uso como herramienta de prueba, uso como herramienta de investigación, principales aplicaciones, tipos de robots simulados o satisfacción de usuarios. Algunos de los programas que menciona son: *ODE, Bullet, V-Rep, ARGos, WeBots, OpenRave, Robotran, Blender, Nvidia* y *Gazebo*. Este último es el que mejor valoración obtuvo y ha sido el empleado en el presente trabajo.

Estos softwares integran las físicas, el control, la conexión con otros dispositivos y las herramientas de resolución numéricas para la simulación de cualquier sistema que se desea recrear.

El uso de estos programas para la simulación de robots está cada vez más extendido. Yun Niu [2] desarrolló una herramienta de simulación de robots móviles mediante los programas *Matlab/Simulink, Gazebo* y *ROS*. Este trabajo se centró más como material docente para el aprendizaje de nuevos algoritmos de guiado de robots.

Otro trabajo de cosimulación entre estos programas es el desarrollado por Wenbin Zha [3] donde realizó la simulación de un brazo robótico (*KUKA iiwa14*) basando el control en un modelo dinámico estimado, comprobando a su vez la estabilidad del sistema mediante el método de Lyapunov [4].

Existen otros tipos de trabajos en los que se comandan trayectorias a cuadricópteros [5,6]. También se ha trabajado con la autodetección de entornos mediante cámaras integradas en robots móviles para la generación de mapas en 2D [7,8].

En el presente artículo se aunará la parte de comanda de trayectorias de los trabajos descritos anteriormente con un sistema de control de pares realimentados basado en un modelo dinámico diseñado previamente. El objetivo del presente trabajo se centrará en realizar la comunicación entre los programas *Matlab/Simulink* y *Gazebo*, desglosando la parte de control para poder modificar y diseñar nuevas trayectorias obteniendo datos parecidos a los de un sistema real. En el apartado 2 se explica cada una de las partes que integra el proceso de interconexión y el diagrama de control diseñado en *Simulink* basado en la dinámica del sistema. Mientras que en el apartado 3 se muestran los resultados de la cosimulación de un brazo articulado de 2 grados de libertad en la cual se comanda una trayectoria establecida, obteniendo datos como: posiciones y velocidades articulares, momentos torsores en las juntas, errores del sistema de control, imágenes procedentes de cámaras, etc. Finalmente, en el apartado 4 se exponen las principales conclusiones obtenidas.

2. METODOLOGÍA

En este apartado se explicará más en detalle la interconexión entre los programas previamente mencionados y la estructura interna de la cosimulación, detallado en la Fig. 1.

El diseño y ensamblaje del sistema mecánico a simular se encuentra en *Gazebo* [9], toda la parte cinemática y dinámica la resuelve el motor de este software. Gracias a controladores internos del programa se pueden transmitir datos registrados en la simulación por sensores mediante mensajes compilados (*C++* o *Python*).

El programa *Robot Operating System (ROS)* [10] es capaz de trabajar con dichos mensajes y realizar la comunicación entre diferentes softwares o estaciones de trabajo, su comunicación multiplataforma se realiza de manera inalámbrica a través de la red local de internet.

En *Matlab/Simulink* [11] se tiene el sistema de control, que mediante mensajes de *ROS* se comunica con *Gazebo* mandando y recibiendo la información necesaria para cumplir unos requisitos deseados, en nuestro caso, la descripción de una trayectoria en el espacio articular del modelo.

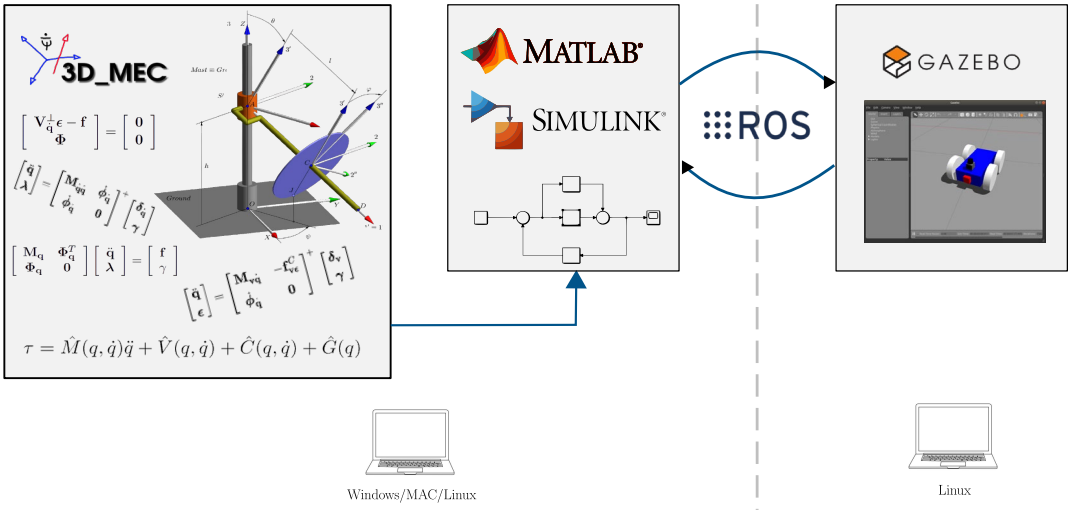

Fig. 1. Esquema conexión de programas. Fuente: Mathworks [11].

2.1. Gazebo

Gazebo es un entorno de simulación de sistemas mecánicos con el cual se es capaz de diseñar y construir infinidad de escenarios para la recreación de operaciones robóticas. Un software con el que se puede trabajar externamente desde su GUI o internamente desde su código fuente (SDFormat/XML). Todo para definir desde las físicas del entorno de simulación (<gravity> <magnetic_field> <wind> <ligth>) pasando por parámetros del solver de simulación (<bullet> <ode> <simbody> <dart> <max_step_size>) hasta llegar al diseño del mecanismo (<model> <link> <joints> <visual> <colision> <sensor> <plugins>).

Su potente interfaz gráfica permite hacer modelados y simulaciones dinámicas con un alto grado de realismo y resolución. Se pueden implementar sensores (cámaras, radares, micrófonos, IMUs) y actuadores. Gracias a la comunicación con *ROS* se puede lanzar simulaciones en servidores remotos o en la nube mediante *CloudSim*.

Como es un software centrado en aplicaciones robóticas, el modelo dinámico del sistema que se desea simular se definirá por la siguiente ecuación no lineal expresada en la forma inversa:

$$\tau = M(q)\ddot{q} + V(q,\dot{q}) + C(\dot{q}) + G(q) \tag{1}$$

donde τ es el vector de [n×1] de pares externos en cada una de las n articulaciones, M(q) es la matriz de inercias [n×n], $V(q,\dot{q})$ es el término de efectos centrífugos y de Coriolis [n×1], G(q) es el vector [n×1] de efectos gravitatorios y $C(\dot{q})$ engloba los esfuerzos de rozamientos viscosos y de Coulomb en las juntas; basado en el modelo de fricción clásico regido por la siguiente ecuación:

$$\tau_{fric} = \upsilon\,\dot{q} + \mu \cdot sign(\dot{q}) \tag{2}$$

donde υ y μ son los coeficientes de rozamiento viscoso y de Coulomb respectivamente.

Todas estas propiedades del sistema se definen en el código fuente de Gazebo con algunos de los siguientes comandos (<mass> <inertia> <pose> <gravity> <damping> <friction>).

Algunos de los sensores que integra el modelo registran posiciones articulares y cartesianas de las juntas del sistema, velocidades y aceleraciones articulares, cámaras, etc. Al sistema se ha dotado de actuadores de par en articulaciones replicando de esta manera los esfuerzos que realizan los motores, los cuales toman valores procedentes del sistema de control en *Simulink*. Se pueden asociar los datos registrados y solicitados por estos sensores y actuadores a mensajes de *ROS* para conseguir la interconexión.

2.2. Robot Operating System (ROS)

ROS es un entorno de trabajo que agrupa múltiples librerías, drivers y herramientas para el desarrollo de aplicaciones robóticas. Este software libre trabaja de manera modular mediante una serie de nodos que se comunican solicitando y enviando mensajes entre sí.

Con su herramienta *catkin* (basado en *Cmake*) se consigue compilar códigos en *.cpp/.cc/.py* donde se definen los nodos, mensajes, publicadores, subscriptores y servidores, generando archivos .so que *Gazebo* es capaz de leer. Con este paquete se puede registrar los datos de las simulaciones generadas en *Gazebo*, visualizar sensores y cámaras, comandar trayectorias en el espacio articular o cartesiano…

Con la herramienta de *ROS* llamada *rqt_graph* se puede visualizar el árbol de nodos y mensajes que se establece en la cosimulación. El sistema de control manda los pares a través del mensaje */M1_DP* y */M2_DP* y *Gazebo* envía el estado de las articulaciones y de las cámaras instaladas en el modelo.

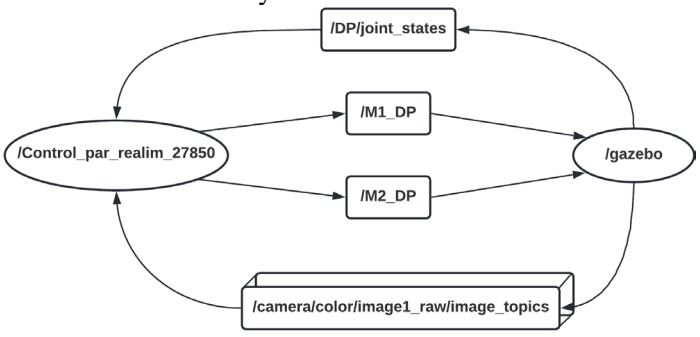

Fig. 2. Diagrama flujo ROS.

2.3. Matlab/Simulink

En 2015, *MathWorks* implementó un módulo de simulación de robots llamado *Robotics System Toolbox* con el cual se podía realizar cosimulaciones entre *Matlab* y *Gazebo*. No fue hasta la versión R2019 cuando se añadió el módulo *ROS Toolbox* con el cual se mejoró la comunicación entre ambos programas [12]. Con él se pueden generar códigos en *C++* a partir de scripts en *Matlab* o diagrama de bloques en *Simulink*, compilarlos con *ROS* y leerlos con *Gazebo*. De esta manera se consigue la comunicación entre los 3 programas. La parte de *Simulink* se utiliza para diseñar el control del sistema mecánico a simular. Se combinan bloques propios del programa, funciones de *Matlab* y la herramienta de *ROS Toolbox*.

Se ha optado por un sistema de control típico en el campo de la robótica [13], aplicados a manipuladores con más de un grado de libertad y regidos por ecuaciones no lineales, el control multiarticular de pares realimentados (Fig. 3). Este tipo de control exige la estimación de los parámetros del modelo dinámico del sistema descrito en la ecuación (1), y la minimización del error radica en la precisión de cálculo de dichos parámetros. De esta manera, se consigue un sistema de control sin parte integral, la cual suele dar problemas en sistemas de control con muchos grados de libertad al contar con integradores que aumentan el número de orden del sistema.

Se define una serie de trayectorias de referencia las cuales se desea que el robot desarrolle. En sistemas articulados, las especificaciones de trayectoria se realizan en el espacio articular en vez de en el cartesiano por diferentes razones: rapidez en los cálculos computacionales al no tener que resolver la cinemática inversa del sistema, mejor control de posiciones singulares, trayectorias que no salgan fuera del espacio de trabajo y obtención de una única configuración del sistema para un estado determinado.

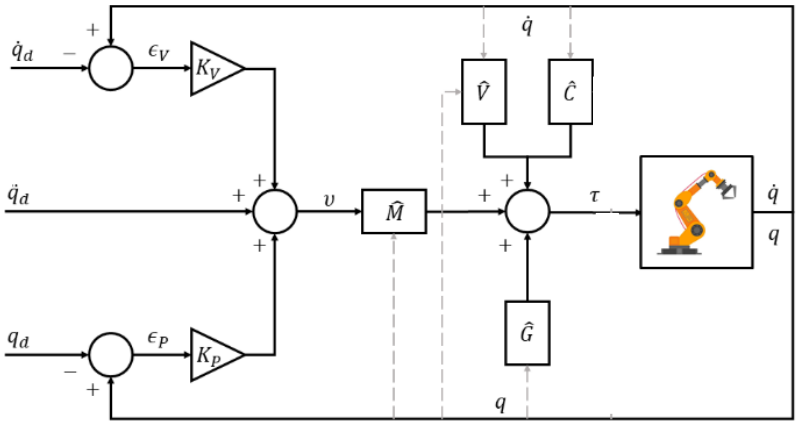

Fig. 3. Esquema de control de pares realimentados.

Con el control se calculan los momentos torsores para cada una de las articulaciones (τ_i). Estos pares se mandan a *Gazebo* vía *ROS* y *Gazebo* devuelve los valores de posición y velocidad (q_i , $\dot{q}_i$) para cada articulación. Estos valores llegan de nuevo a *Simulink* por mensaje de *ROS* y se comparan con los valores de las trayectorias de referencia (q_d , $\dot{q}_d$). Con ganancias proporcionales (K_P , K_V) se minimiza el error de posición (ϵ_P) y de velocidad (ϵ_V). Este proceso se realiza en cada paso de simulación para toda una trayectoria.

Si se analizan las señales del diagrama de control de la Fig. 3 se puede relacionar el modelo dinámico estimado del sistema con el simulado en *Gazebo*. La entrada de pares comandados se puede ver como:

$$\tau = \hat{M}(q)\,\upsilon + \hat{V}(q,\dot{q}) + \hat{C}(\dot{q}) + \hat{G}(q) \tag{3}$$

donde $\hat{M}$, $\hat{V}$, $\hat{C}$ y $\hat{G}$ son estimaciones del modelo dinámico del sistema simulado. Y υ se puede interpretar como:

$$\upsilon = \ddot{q}_d + K_V(\dot{q}_d - \dot{q}) + K_P(q_d - q) \tag{4}$$

K_V y K_P son matrices diagonales de ganancias que minimizan el error en velocidades y posiciones respectivamente. Igualando el par del control de la ecuación (3) con el par de entrada al modelo dinámico expresado en la ecuación (1) se obtiene:

$$\widehat{M}\left(\ddot{q}_d + K_V(\dot{q}_d - \dot{q}) + K_P(q_d - q)\right) + \hat{V} + \hat{C} + \hat{G} = M\ddot{q} + V + C + G \tag{5}$$

restando en ambos términos de la ecuación por $-\widehat{M}\ddot{q}$ y sustituyendo $e = (q_d - q)$ obtenemos que:

$$\ddot{e} + K_V\dot{e} + K_P e = M^{-1}\left(\left(M - \widehat{M}\right)\ddot{q} + \left(V - \hat{V}\right) + \left(C - \hat{C}\right) + \left(G - \hat{G}\right)\right) \tag{6}$$

Cuanto más preciso sea la estimación del modelo dinámico menor será la diferencia entre los valores reales y calculados en el segundo término de la ecuación (6) consiguiendo la siguiente expresión:

$$\ddot{e} + K_V\dot{e} + K_P e \approx 0 \tag{7}$$

De manera que se consigue un sistema de segundo orden lineal y desacoplado al ser K_V y K_P matrices diagonales. Comparando esta ecuación con la ecuación general de un sistema lineal de segundo orden:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \equiv s^2 + K_V s + K_P = 0 \tag{8}$$

donde ζ es el coeficiente de amortiguamiento y ω_n la frecuencia natural no amortiguada. De este modo se relacionan los requisitos de respuesta del sistema con las ganancias del control:

$$K_P = \omega_n^2 \tag{9}$$

$$\boldsymbol{K_V = 2\xi\omega_n} \tag{10}$$

Estos valores se ajustan mediante las especificaciones temporales que se definen para sistemas de control de segundo orden. Entre ellas se encuentra el tiempo de estabilización al 5% ($t_{s(\pm 5\%)}$) y el máximo sobreimpulso (M_P):

$$t_{s(\pm 5\%)} = \frac{\pi}{\zeta \cdot \omega_n} \tag{11}$$

$$\boldsymbol{M_P = e^{-\frac{\pi\cdot\zeta}{\sqrt{1-\zeta^2}}}} \tag{12}$$

También resulta útil variar la discretización del tiempo en ambos programas, haciendo que las frecuencias de simulación del sistema y de comandas del control sean diferentes.

El tiempo utilizado en el sistema de control se toma de *Gazebo* (mediante otro mensaje llamado */clock*), el cual se puede configurar para poder trabajar con valores de tiempo reales.

La frecuencia de simulación del motor en *Gazebo* es de 100 Hz, un orden superior a la del motor de *Simulink* de 10 Hz. De esta manera se establece un bucle de simulación el cual actualice los datos de manera rápida y un bucle de control que vaya comandando y registrando dichos datos a menor velocidad. Con ello, el coste computacional se ve reducido consiguiendo a su vez una mejor visualización de los resultados obtenidos.

2.4. Lib_3D_MEC_Matlab

Para la obtención de los términos dinámicos se hace uso de la librería desarrollada por la *Universidad Pública de Navarra* [14-16] con la cual se puede realizar análisis y simulaciones en el ámbito de la dinámica de sistemas multicuerpos. Este set de códigos y funciones de *Matlab* permiten diseñar, analizar y simular cualquier sistema multicuerpo que se desee, consiguiendo obtener las ecuaciones cinemáticas y dinámicas del mismo.

De esta manera, se ha diseñado un sistema en *Lib_3D_MEC_Matlab* que replique al de *Gazebo*, permitiendo obtener sus parámetros los cuales serán introducidos en *Simulink*. La precisión del control radicará en lo aproximado que sean ambos modelados.

Primero se resuelve el problema cinemático: parámetros y coordenadas, puntos, bases, sistemas de referencia, sólidos, etc.

Después se especifican los esfuerzos dinámicos involucrados: acciones constitutivas, gravitacionales e inerciales, esfuerzos viscosos y de fricción en juntas y pares externos. La obtención de las ecuaciones dinámicas se realiza mediante el principio de las potencias virtuales ya que con esta formulación se eliminan los términos de reacciones en los enlaces.

Igualando las ecuaciones dinámicas (en la forma implícita) con el modelo dinámico expresado en la ecuación (1):

$$\Phi_{Dyn}^{VP} = M(q)\ddot{q} + \delta(q,\dot{q}) \tag{13}$$

donde:

$$\delta(q,\dot{q}) = V(q,\dot{q}) + C(\dot{q}) + G(q) - \tau \tag{14}$$

$$\boldsymbol{M(\mathbf{q})} = \frac{\boldsymbol{\partial \Phi_{Dyn}^{VP}}}{\boldsymbol{\partial \ddot{q}}} \tag{15}$$

Con M y δ se puede definir el sistema de control de la Fig. 3, sin embargo, para poder analizar más en detalle la contribución de cada uno de los términos se continua con el desarrollo para extraer las demás partes. Evaluando δ cuando las velocidades son nulas podemos eliminar los esfuerzos centrífugos y de rozamientos viscosos para poder determinar la contribución gravitacional:

$$\delta(q,0) = G(q) - \tau \tag{16}$$

$$\boldsymbol{G(q)} = \boldsymbol{\delta(q,0)} + \boldsymbol{\tau} \tag{17}$$

Para determinar los esfuerzos producidos por rozamientos viscosos y de Coulomb, agrupamos los coeficientes de rozamientos de la ecuación (2) en un vector del tipo:

$$\bar{p} = [v,\mu] \tag{18}$$

y derivando δ con respecto a estos parámetros se puede obtener la contribución de C:

$$\boldsymbol{C(\dot{q})} = \frac{\partial \delta}{\delta p}\bar{p} \tag{19}$$

Por último, solo queda despejar el término de los efectos centrífugos y de Coriolis de la ecuación (14):

$$\boldsymbol{V(q,\dot{q})} = \delta - C(\dot{q}) - G(q) + \tau \tag{20}$$

Este proceso se realiza simbólicamente con la librería *lib_3D_MEC_Matlab* y, mediante la herramienta *matlabFunction*, se exportan M, G, C y V para integrarlo al diagrama de control en *Simulink*.

3. SIMULACIÓN Y RESULTADOS

Para la parte experimental del presente artículo se ha hecho uso de un equipo informático con las características descritas en la Tabla 1.

Tabla 1. Propiedades del equipo informático.

Sistema operativo	Ubuntu 20.04.4 LTS (64 bits)
Versión GNOME	3.36.8
Procesador	Intel Core i5-9500 CPU@3.00Ghz x6
Memoria RAM	32 GB
Memoria HDD	1.5 TB
Tarjeta gráfica	Mesa Intel UHD Graphics 630 (CFL GT2)

Los programas citados en el apartado anterior se encuentran en las siguientes versiones: *Gazebo 11.0, ROS Noetic Ninjemys* y *Matlab/Simulink R2021a*.

Con respecto a las simulaciones, el problema con el que se ha trabajado ha sido un sistema articulado plano de 2 grados de libertad. El brazo articulado se representa en la Fig. 4 donde se indican los parámetros del sistema y sus variables. En la Tabla 2 se especifican algunos de sus valores.

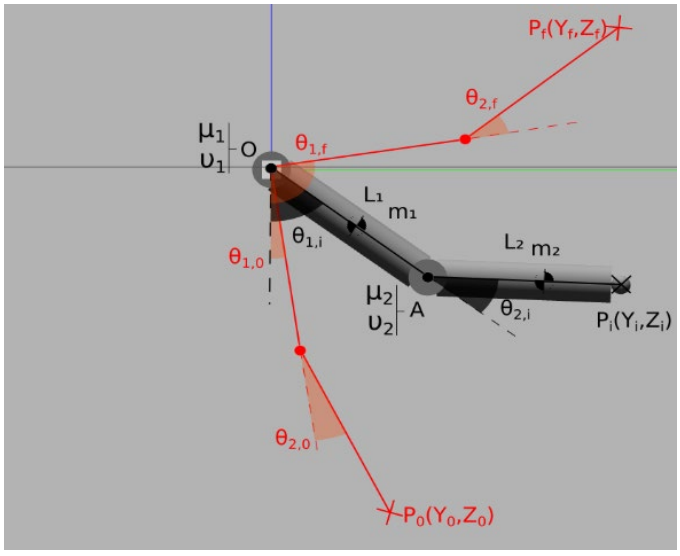

Fig. 4. Parametrización del brazo articulado.

De color negro se especifican los parámetros y algunas variables en un i-instante de tiempo cualquiera, mientras que de color rojo se indican las posiciones iniciales y finales del sistema con los subíndices correspondientes. Estas posiciones son meramente descriptivas sin tener relación alguna con las trayectorias posteriormente generadas.

Como el objetivo es validar el funcionamiento del proceso de conexión, los valores de los parámetros no son de gran relevancia, por lo que se han tomado unos valores dentro de unos rangos razonables.

Tabla 2. Parámetros del brazo articulado.

Parámetro	L_1 (m)	L_2 (m)	m_1 (kg)	m_2 (kg)	μ_1 (-)	μ_2 (-)	υ_1 (-)	υ_2 (-)
Valor	1	1	1	1	0.25	0.25	0.2	0.2

En *Matlab* se han diseñado una serie de trayectorias mediante curvas de Bézier de 5º orden llevando al sistema a diferentes posiciones articulares en distintos instantes de tiempo. Las especificaciones de las trayectorias pueden verse en la Tabla 2 y de manera gráfica en la Fig. 6 en color negro.

Tabla 3. Especificaciones de la trayectoria.

Tramo	**Tiempo (s)**	**Posición #1 (rad)** $[\theta_{1,0}, \theta_{1,f}]$	**Posición #2 (rad)** $[\theta_{2,0}, \theta_{2,f}]$
#1	10 - 17.5	[0 , 2]	[0 , 2]
#2	25 - 30	[2 , -0.5]	[2 , 1.57]
#3	35 - 40	[-0.5 , -1.57]	[1.57 , 0]
#4	50 - 55	[-1.57 , -3.14]	[0 , 0]

En *Simulink* se ha diseñado un sistema de control replicando al de la Fig. 3. La parte de guiado de trayectorias se ha realizado mediante unos diagramas de flujo que comparan los valores de posición y velocidad de las trayectorias definidas con las simuladas. Las ganancias de los controladores se calculan al final de este apartado, estos valores se han obtenido mediante compromiso entre la estabilidad de la cosimulación (ya que al aumentar la ganancia el proceso se descontrola) y especificaciones temporales razonables (tiempo de estabilización, sobreoscilación y error en el estacionario).

La suma de los pares aplicados a cada articulación se asigna a un mensaje en *ROS* llamados *M1_DP* y *M2_DP*. El estado de las articulaciones del mecanismo en *Gazebo* se obtienen del mensaje *DP/joint_state* y se desglosan en posiciones y velocidades articulares.

Este proceso se sincroniza de manera fluida observando la cosimulación en la Fig. 5. En el fondo de la imagen se muestra el sistema de control de *Simulink*, arriba a la izquierda lo registrado por la cámara de *Gazebo*, abajo a la izquierda los pares comandados por el control a cada articulación y en la parte inferior derecha el entorno gráfico de *Gazebo*.

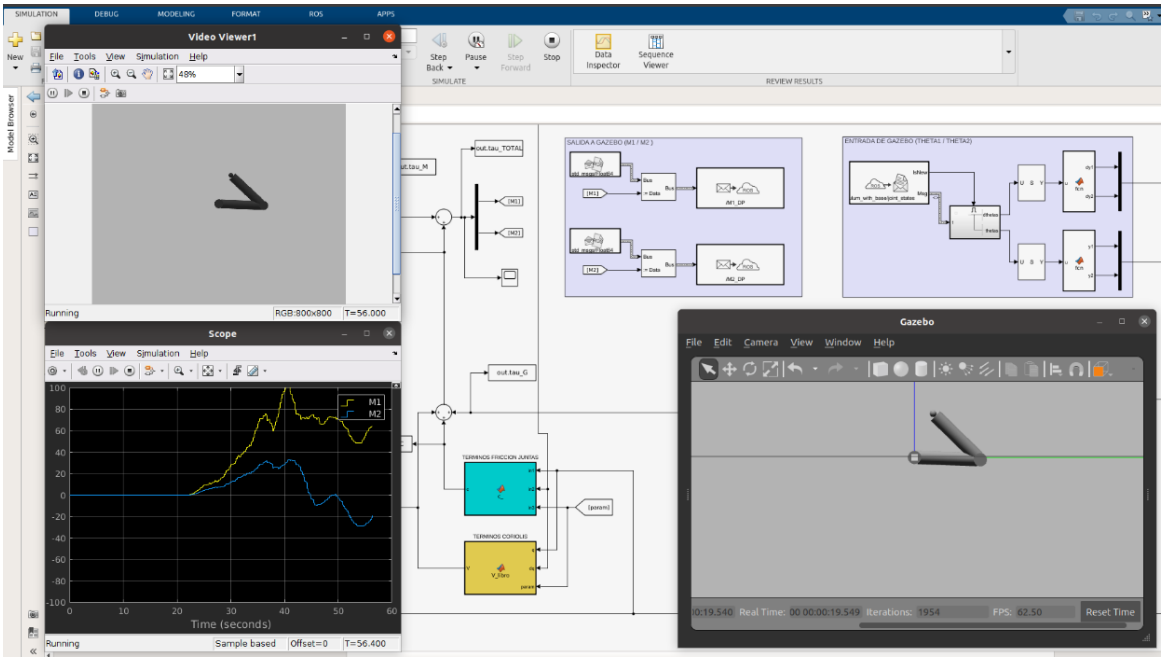

Fig. 5. Parametrización del brazo articulado.

El coste computacional se eleva si obtenemos estos datos de manera simultánea, empeorando la comunicación entre ambos programas e incrementando consigo el error de posición. En vez de lanzar la cosimulación con las interfaces gráficas de los programas, estas se deshabilitan y se almacenan todos los resultados en variables de *Matlab* para su posterior visualización. La trayectoria de consigna se muestra en la Fig. 6 donde se realiza también una comparativa entre los valores comandados (negro) y los procedentes de la simulación.

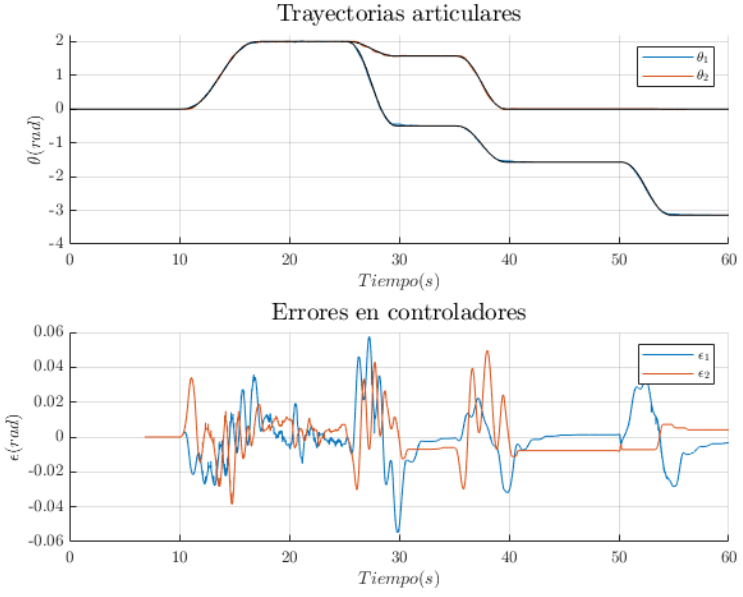

Fig. 6. Comparativa trayectoria de referencia y simulada.

Se observa como existen unas pequeñas oscilaciones y también se aprecia un pequeño error, llegando a obtener máximos de ±0.06 rad en zonas transitorias y oscilando entre valores de ±0.02 rad en estacionarios.

Este error viene determinado por la ganancia de los controladores (K_P, K_V) definidas en *Simulink*. Con el desarrollo descrito anteriormente se define las especificaciones temporales de respuesta del sistema y, mediante las ecuaciones (11) y (12), se calculan las ganancias de los controladores.

Tabla 4. Ajuste de los controladores.

Parámetro	$t_{s(5\%)}$ **(s)**	M_P **(%)**	ζ **(-)**	ω_n **(Hz)**	K_P **(s^{-2})**	K_V **(s^{-1})**
Valor	0.8	20	0.456	8.613	74.196	7.855

Como la cosimulación comienza lanzando *Gazebo* y posteriormente el control de *Simulink*, los datos comienzan a registrarse en instantes de tiempo comprendidos entre 5-10 segundos. El control interrumpe este registro cuando se ha llegado a los 60 segundos de simulación.

En la Fig. 7 se muestran la contribución de pares de cada uno de los términos del modelo dinámico. Entre ellos los que más destacan son los valores que contrarrestan los efectos gravitatorios (Fig. 7d) e inerciales (Fig. 7a), estos últimos dependientes del error de posición y velocidad. Los términos centrífugos (Fig. 7b) y de rozamientos (Fig. 7c) al depender de la velocidad son menores.

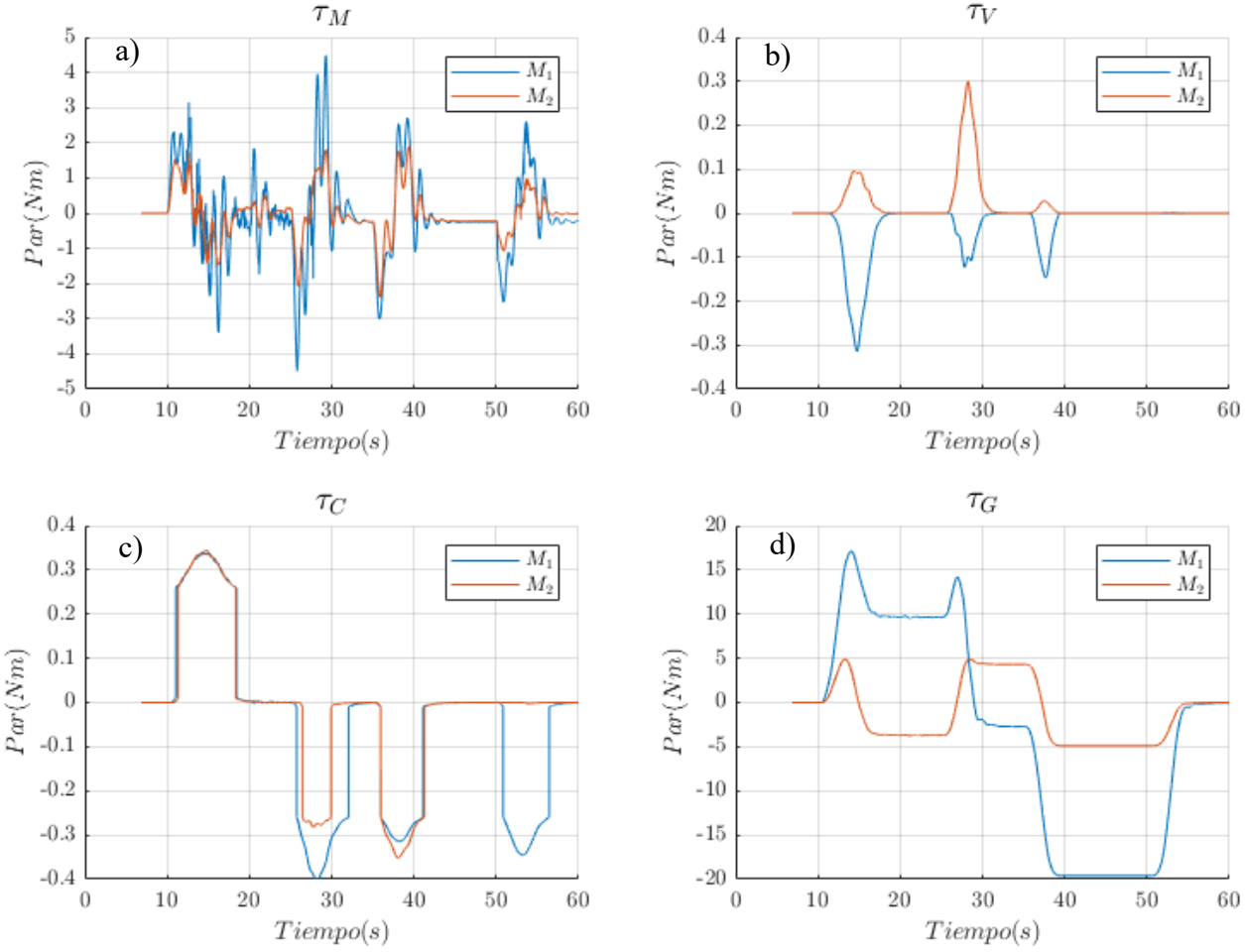

Fig. 7. Contribución de pares: a) Término inercial; b) Término centrifugo; c) Término de rozamiento; d) Término gravitacional.

Se observa también un extraño comportamiento en los pares por efectos de rozamiento. Esto se debe a causa del modelo clásico definido en la ecuación (2), al contar con el término de $sign(\dot{q})$ provoca

discontinuidades cuando las velocidades son nulas. Esto supone una ventaja al poder representar la fricción en situación estática, pero a su vez esta discontinuidad matemática provoca un comportamiento más inestable en el sistema de control, pero como estos esfuerzos son de menor orden no altera el comportamiento del sistema.

El mecanismo al partir de una posición de equilibrio (posiciones articulares nulas) los valores de par en el primer tramo de la trayectoria son nulos. Como la trayectoria diseñada comanda al sistema a otro punto de equilibrio en su parte final (Fig. 6), los pares también tenderán a anularse en dicho tramo. Las contribuciones inerciales y gravitatorias tienden a 0.

La suma de las 4 contribuciones se representa en la Fig. 8, donde se observa un comportamiento bastante típico en sistemas de control de segundo orden, sobreoscilaciones. También se aprecia unas pequeñas oscilaciones en los tramos estacionarios, esto se debe a la comunicación entre softwares que añade ruido y perturbaciones.

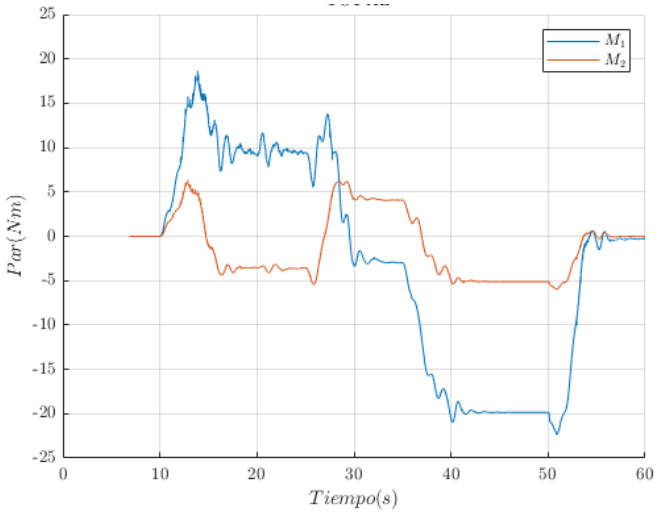

Fig. 8. Par articulaciones entrada a Gazebo.

Al realizarse de manera simultánea la cinemática directa del sistema, se pueden obtener las coordenadas cartesianas de las articulaciones y del elemento terminal P a partir de los valores articulares de posición.

En la Fig. 9 se representa la animación del brazo articulado en los 4 tramos de posiciones de la trayectoria, comparando los valores simulados con los de referencia.

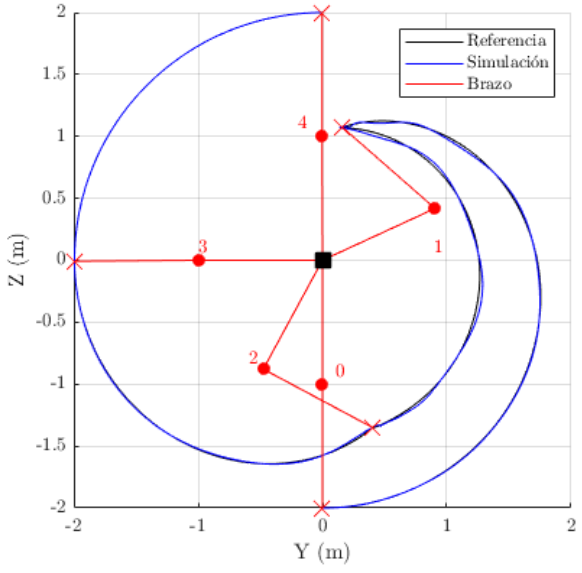

Fig. 9. Animación sistema a lo largo de la trayectoria.

Esta manera de representación resulta más clara e intuitiva para observar la eficacia del sistema de control. Aun así, se puede apreciar una pequeña diferencia de posiciones cartesianas del elemento terminal en algunas zonas de la trayectoria.

Otros posibles resultados de la cosimulación son valores de velocidad articular o imágenes procedentes de las cámaras de *Gazebo*, que puedan interpretarse como una matriz de valores RGB cuya dimensión viene dada por la resolución del sensor. En nuestro caso, [800×800×3×n] para n instantes de tiempo.

Este tipo de datos también resulta de gran interés fuera del ámbito de estudio de la dinámica de sistemas multicuerpo, como puede ser el desarrollo y entrenamiento de algoritmos de inteligencia artificial basado en redes neuronales, los cuales necesitan imágenes discretizadas en formato vectorial.

4. CONCLUSIONES

El principal objetivo del artículo es la interconexión y cosimulación de sistemas mecánicos mediante herramientas computacionales de distintos ámbitos. Se ha conseguido desglosar de la física del sistema la parte de control para poder generar una herramienta más intuitiva y flexible, aunque el coste computacional sea mayor al tener que ejecutar ambos programas simultáneamente.

Al separar ambas partes se pueden probar diferentes tipos de control aplicados al campo de la robótica, en nuestro caso se ha hecho uso del control de realimentación de pares computados basados en el modelo dinámico.

Este tipo de control se ha llevado a la practica en un sistema doble articulado y, a la vista de los resultados, se pueden entender los diferentes factores que afectan a la cosimulación: propiedades del motor del software de simulación, frecuencias en las comandas del sistema de control, exactitud en el cálculo de los parámetros del modelo dinámico, ajuste de controladores, etc.

La obtención de resultados con una herramienta externa también resulta interesante ya que el postproceso de dichos datos permite utilizarlos en campos ajenos a la robótica.

El sistema simulado resulta sencillo y básico, pero al introducir más grados de libertad el coste computacional del control se ve incrementado, volviendo al sistema inestable e incumpliendo las trayectorias comandadas.

Una posible línea futura es la modificación del sistema de control realizando uno adaptativo que minimice el error o una mejora en el cálculo de los parámetros de modelado para conseguir simular sistemas más complejos. También puede ser una solución interesante para no sobrecargar el procesador del equipo informático realizar simulaciones en remoto, donde una estación ejecute la simulación y otro lance el sistema de control.

AGRADECIMIENTOS

Este trabajo fue financiado por la *"Convocatoria de ayudas a proyectos de I+D del Gobierno de Navarra"* bajo los proyectos con Ref. 0011-1365-2021-000080 y Ref. 0011-1411-2021-000023.

REFERENCIAS

[1] S. Ivaldi, J. Peters, V. Padois, F. Nori, "Tools for simulating humanoid robot dynamics: A survey based on user feedback in Humanoid Robots (Humanoids)", de *14th IEEE-RAS International Conference*, Madrid (2014)

[2] Y. Niu, H. Qazi, Y. Liang, "Building a Flexible Mobile Robotics Teaching Toolkit by Extending MATLAB/Simulink with ROS and Gazebo", de *7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, Budapest (2021)

[3] W. Zha, X. Xu, Z. Chen, A. Rodic, P.B. Petrovic, "Manipulator Tracking Algorithm Based on Estimated Dynamics and Time-Varying Output Constraint State", de *6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, Chongqing (2021)

[4] J.P. LaSalle, S. Lefschetz, P. Howlett, *Stability by Liapunov's Direct Method with Applications,* Nueva York: Academic Press (1961)

[5] M. Nithya, M.R. Rashmi, "Gazebo - ROS - Simulink Framework for Hover Control and Trajectory Tracking of Crazyflie 2.0", de *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, Kochi (2019)

[6] K. Kumar, S.I. Azid, A. Fagiolini, M. Cirrincione, "Erle-copter Simulation using ROS and Gazebo", de *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*, Palermo (2020)

[7] YJ. Lim, M. Belge, "Automated ROS and ROS 2 Node Generation from Prototyping to Production (Webinar)", 7 Diciembre 2021. [En línea]. Available: https://es.mathworks.com/videos/automated-ros-and-ros-2-node-generation-from-prototyping-to-production-1639065373693.html?s_v1=40668&elqem=3551961_EM_NA_LWB_21-12_AUTOMATED-ROS-NODE-GENERATION-PROTOTYPING_POST_CERT&elqTrackId=8c3ff88b9b9d43b69b7287b309c83.

[8] Y.J. Lim, R. George, J. Antoniou, "Modeling and Simulation of Autonomous Mobile Robot Algorithms", 26 Mayo 2022. [En línea]. Available: https://es.mathworks.com/videos/modeling-and-simulation-of-autonomous-mobile-robot-algorithms-1654096329276.html?s_v1=42724&elqem=3695400_EM_NA_LWB_22-05_MODELING-SIMULATION-AUTONOMOUS-MOBILE-ROBOT_POST&elqTrackId=02d5b65cd11146dcab85c2efcd600e62&elq=9d92.

[9] "Gazebo" [En línea]. Available: https://gazebosim.org/home

[10] "Robot Operating System" [En línea]. Available: https://www.ros.org/

[11] "MathWorks" [En línea]. Available: https://es.mathworks.com/products/matlab.html

[12] "ROS Toolbox" [En línea]. Available: https://es.mathworks.com/products/ros.html

[13] J.J. Craig, *Introduction to robotics: mechanics and control*, 3ª ed., Pearson Educacion (2005)

[14] J. Aginaga, X. Iriarte, J. Ros, J.M. Aguinagalde, "Docencia de la mecánica del sólido rígido mediante el programa 3D Mec", *II Congreso Internacional de Docencia Universitaria (CIDU)*, Vigo (2011)

[15] J. Ros, L. Arrondo, J. Gil, X. Iriarte, "Lib3D Mec-GiNaC, a Library for Symbolic Multibody Dynamics", *ECCOMAS Thematic Conference of Multibody Dynamics*, Milano (2007)

[16] J. Ros, A. Plaza, X. Iriarte, J.M. Pintor, "Symbolic multibody methods for real-time simulation of railway vehicles", de *Multibody System Dynamics* (2018)

MULTIPLATFORM MODELLING FOR THE CONTROL OF AN ARTICULATED SYSTEM

Abstract – In this paper a couple of mechanical simulation softwares have been worked with. One has been used for representing the physics and dynamics of mechanisms (Gazebo) and the other one has the control of the system (Simulink), the way to interconnect them is through Robot Operating System software (ROS). Thus, a co-simulation of an articulated mechanism has been realised in which Simulink commands an ideal trajectory to Gazebo sending joints torques. Gazebo provides joint position values of the simulation to Simulink, and through its control system, the error between the real and the ideal trajectory gets smaller sending again the correct amount of torque to Gazebo, that works step by step during the simulation. In addition, the dynamic model parameters required by the control system have been calculated using a Matlab's library for the analysis of multibody system (Lib_3D_MEC_Matlab), which had been developed by the Public University of Navarra.

Keywords – Robotic, co-simulation, control design, Gazebo, ROS, Simulink.