Diseño de control y simulación de un brazo robot para asistencia en laboratorio

Control y programación de Robots
Curso 2020-2021

Pablo León Barriga
Pablo Manuel Guzmán Manzanares
Fátima María Fernández Vázquez

Índice

1. Descripción del proyecto **2**

2. Brazo robótico *Jaco*^{2} **2**

3. Gazebo **3**

4. Cinemática **3**

4.1. Cinemática directa 3

5. Dinámica **5**

6. Controladores **6**

6.1. Control por par computado en el espacio articular 6

6.2. Controlador PID de cada eslabón del sistema 6

7. Simulación en Gazebo **7**

8. Resultados **10**

8.1. Resultados del controlador por defecto de Gazebo 10

8.2. Resultados del controlador por Par Computado 12

8.3. Resultados del controlador PID 14

9. Plan de trabajo **15**

10. REFERENCIAS **17**

1. Descripción del proyecto

Este proyecto consiste en el desarrollo de un brazo robótico para asistencia en laboratorio, con el cual se intentaría facilitar el acceso a muestras, siendo desplazadas desde el lugar de trabajo a un sitio de almacenamiento específico, además de ayudar en la manipulación de sustancias peligrosas. De esta forma se agiliza el proceso de transporte y la seguridad de los científicos dentro del entorno de trabajo.

Se emplearán algoritmos de control, los cuales se evalúan en el robot, siempre atendiendo a las limitaciones que este pudiese tener, y se simulará en el entorno de Gazebo, plataforma ofrecida por Ros Kinetic.

2. Brazo robótico $Jaco^2$

Para la simulación de los controladores se ha utilizado el brazo robótico $Jaco^2$ de Kinova, que tiene 6 grados de libertad, con un efector final de tres dedos. Está compuesto por articulaciones de fibra de carbono, y actuadores de aluminio. La fibra de carbono resulta en un robot muy ligero, y fiable, lo que facilita su movilidad hacia otra posición de trabajo, y el aluminio proporciona adicionalmente, una mejor disipación del calor durante sus movimientos. Además, como hardware adicional, la empresa provee un joystick, un puerto usb para ser conectado a un ordenador personal, desde el cual se pueden enviar comandos gracias al kit de software desarrollado.

Figura 1: Componentes del robot $Jaco^2$ [1]

3. Gazebo

La simulación se realiza con el entorno Gazebo, para el cual Kinova nos ofrece una serie de paquetes que facilitan su interacción, proporciona una simulación del robot en 3D, esto ayuda a poder probar algoritmos de diseño rápidamente, y diferentes tipos de control atendiendo a su posición y velocidad [2]. Se proporciona también un paquete para facilitar el control de trayectorias. Todas estas interacciones se desarrollan a lo largo de la memoria.

4. Cinemática

4.1. Cinemática directa

Un robot manipulador consiste en una serie de componentes rígidos que se unen a través de pares que pueden ser de diferentes tipos, pero esencialmente, los pares a tratar en este robot serán de rotación, estos pares se conectan unos con otros formando una cadena que conforma al robot.

La estructura mecánica está caracterizada por 6 grados de libertad cada uno correspondiente a cada articulación del robot, con los que se determinará la posición en la que se encuentra el robot.

Se tomará un origen de coordenadas que estará ligado a la base del robot mediante un vector unitario. La función de cinemática directa se expresa mediante una matriz de transformación homogénea, que nos transforma el sistema de ejes principal al sistema de ejes final que representa el efector del robot.

Para calcular la cinemática directa para el manipulador de acuerdo con la transformación homogénea, se debe derivar la posición relativa junto a la orientación de los eslabones consecutivos. Una de las posibles soluciones a implementar es el proceso de Denavit-Hartenberg. Para su uso es necesario conocer una serie de parámetros geométricos básicos que se han obtenido del manual de Kinova-Robotics [3].

Las matrices de transformación asociadas a cada articulación, es la siguiente según el proceso de DH

$$T_n = \begin{bmatrix} cos(\Theta_i) & -cos(\alpha_i) \cdot sin(\Theta_i) & sin(\alpha_i) \cdot sin(\Theta_i) & a_i \cdot cos(\Theta_i) \\ sin(\Theta_i) & cos(\alpha_i) \cdot cos(\Theta_i) & -sin(\alpha_i) \cdot cos(\Theta_i) & a_i \cdot sin(\Theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

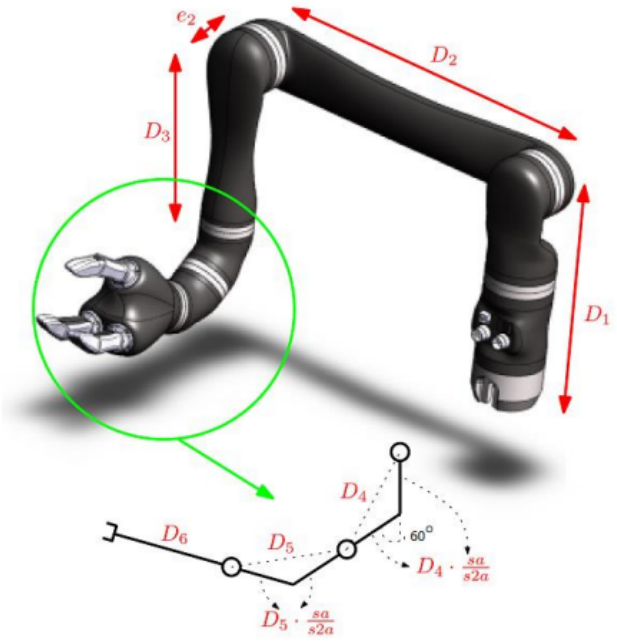


Figura 2: Sistema de ejes del $Jaco^2$ [3]

Robot length values (meters)		
D1	0.2755	Base to elbow
D2	0.4100	Arm length
D3	0.2073	Front arm length
D4	0.0741	First wrist length
D5	0.0741	Second wrist length
D6	0.1600	Wrist to center of the hand
e2	0.0098	Joint 3-4 lateral offset

Figura 3: Longitudes de los eslabones [3]

Classic DH parameters				
i	**alpha(i-1)**	**a(i-1)**	**di**	**theta1**
1	pi/2	0	D1	q1
2	pi	D2	0	q2
3	pi/2	0	-e2	q3
4	2*aa	0	-d4b	q4
5	2*aa	0	-d5b	q5
6	pi	0	-d6b	q6

Figura 4: Parámetros de DH [3]

Equations for transformation from DH algorithm to JACO² physical angles
Q1(Jaco²) = -Q1(DH Algo)
Q2(Jaco²) = Q2(DH Algo) + 90
Q3(Jaco²) = Q3(DH Algo) - 90
Q4(Jaco²) = Q4(DH Algo)
Q5(Jaco²) = Q5(DH Algo) + 180
Q6(Jaco²) = Q6(DH Algo) - 90

Figura 5: Ángulos de DH [3]

Los ángulos físicos del robot manipulador $Jaco^2$ deben convertirse a los ángulos del algoritmo de Denavit-Hartenberg.

5. Dinámica

El cálculo de la dinámica permite la realización de controladores más precisos que los creados exclusivamente teniendo en cuenta la cinemática del robot. Esto se debe a que con la dinámica se incluyen los efectos que tiene la masa de cada uno de los elementos del robot sobre el movimiento, algo que la cinemática no plantea. La dinámica permite también conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo. El modelo dinámico inverso permite expresar las fuerzas y los pares involucrados en el movimiento del brazo en función de la evolución de las coordenadas articulares y sus derivadas.

$$\tau(t) = f(q(t))$$

Existen dos formulaciones para el obtener el modelo dinámico de un robot: la formulación de Newton-Euler, basada en la segunda ley de Newton, y la formulación de Lagrange, que se apoya en el balance energético del sistema. En este proyecto se desarrolla esta segunda formulación.

$$L = E_c - E_p$$
$$\tau_i = \frac{d}{dt} \cdot \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i}$$

Para realizar estos cálculos se han ampliado varios códigos utilizados en la asignatura de Fundamentos de Robótica, de manera que, en lugar de ser para robots de 3 GDL, sirva para calcular el método de Lagrange para el $JACO^2$.

Para ello se necesitará conocer los parámetros dinámicos del robot, datos no disponibles abiertamente puesto el fabricante únicamente los aporta a clientes. Se opta por tanto a utilizar los parámetros dinámicos presentes en el documento "kinova_inertial.xacro" [4], incluido en el paquete de kinova-ros. Dentro de este archivo se incluyen los parámetros de masa, longitud y cálculos necesarios para obtener las matrices de inercia. Dicha información se porta a MATLAB, y se generan los scripts necesarios para calcular las matrices de inercia de cada uno de los eslabones del robot, junto con los actuadores.

Una vez obtenidos los parámetros, estos se introducen en el código principal que calculará las matrices de inercias (M), la de los términos de Coriolis y fuerzas centrípetas (C), y la de los términos gravitacionales (G). Una vez calculados, el modelo dinámico del robot queda de la siguiente manera

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

6. Controladores

6.1. Control por par computado en el espacio articular

Incluir el modelo dinámico en el diseño de control supondrá una mejora de la estabilidad, el rendimiento y la precisión siempre que las estimaciones se aproximen al modelo real. Se obtiene además una gran ventaja al linealizar el sistema, reduciendo los efectos de acoplamiento.

Este controlador, a pesar de haber sido calculado, no se ha podido implementar dentro de gazebo debido a la longitud de las ecuaciones resultantes de las matrices M, C y G (si se quiere ver la longitud de las mismas, el archivo **ma_va_ga_LAG.mat** contiene todas las matrices calculadas mediante el método de Lagrange. Por otra parte, se modificó el control de manera que se pudiera publicar directamente el valor de cada par que se quería dar a cada articulación, y mediante la herramienta de ROS rqt se comprobó que es posible, por lo que lo único que se necesitaría es poder implementar el cálculo arriba descrito.

6.2. Controlador PID de cada eslabón del sistema

Una vez se tienen las *matricesdd* dinámicas del sistema es posible obtener la función de transferencia de cada uno de los eslabones del brazo robótico, lo que permitiría calcular un controlador PID para cada uno de los links. Se comienza realizando varias simplificaciones sobre las matrices anteriores, y calculándolas en torno al punto

de equilibrio del robot, dado por un valor de π a todas las articulaciones. Una vez se simplifican, obtener las funciones de transferencia es trivial. Esto se realiza en el archivo **func_transf_JACO2.m**.

Una vez se tienen las funciones de transferencia, se pasa a calcular un PID mediante técnicas frecuenciales. Teniendo en cuenta que se publica a 500 Hz, o cada 2 ms, se escoge un tiempo de subida en bucle cerrado del sistema de 15 veces el tiempo de muestreo, resultando finalmente en una frecuencia de corte de aproximadamente 104 rad/s. La metodología para diseñar el control mediante técnicas frecuenciales está incluido en el archivo **controlPID.m**. En este caso se escoge un margen de fase deseado de 75, lo que evitaría las sobreoscilaciones. Una vez obtenidos los controladores, en el archivo **paramPID.m** se obtienen los parámetros finales que se introducen dentro del controlador incluido en ROS, dando los resultados que se explican más adelante.

7. Simulación en Gazebo

Para trabajar con el controlador en Gazebo, tendremos que estudiar el PROCESO de ROS para este cometido. ROS nos ofrece una serie de herramientas para facilitarnos el establecimiento de nuestros controladores calculados, contenidas en un modulo llamado **ros_control**.

Este es el esquema de control que la Wiki de Ros referencia para mostrar gráficamente el funcionamiento de las herramientas que proporciona.

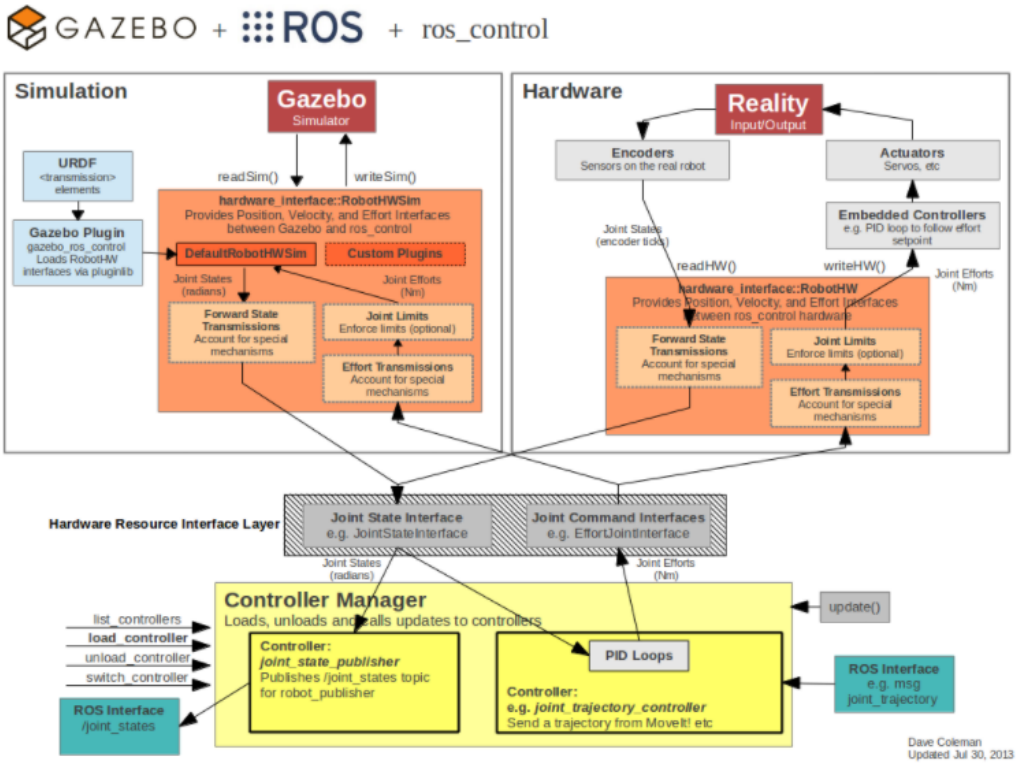


Figura 6: Esquema de control de ROS [5]

Se necesita crear un directorio que contendrá nuestros controladores, esto se hace en una carpeta llamada 'config' donde meteremos nuestro archivo de configuración .yaml y otra carpeta launch donde se encuentra el archivo .launch que mandará el controlador a gazebo y al sistema simulado.

Se configura el archivo .yaml, que es el que contiene las ganancias del PID y la configuración del controlador [5].

```
rrbot:
  # Publish all joint states ----------------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  # Position Controllers ---------------------------------------
  joint1_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint1
    pid: {p: 100.0, i: 0.01, d: 10.0}
  joint2_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint2
    pid: {p: 100.0, i: 0.01, d: 10.0}
```

Tras esto se crea el .launch, encargado de cargar el controlador del .yaml en nuestro modelo de la simulación. Igualmente podemos encontrar un ejemplo del formato en el tutorial citado:

```
<launch>
  <!-- Load joint controller configurations from YAML file to parameter
      server -->
  <rosparam file="$(find rrbot_control)/config/rrbot_control.yaml"
      command="load"/>
  <!-- load the controllers -->
  <node name="controller_spawner" pkg="controller_manager" type="spawner
      " respawn="false"
    output="screen" ns="/rrbot" args="joint1_position_controller
       joint2_position_controller joint_state_controller"/>

  <!-- convert joint states to TF transforms for rviz, etc -->
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="
      robot_state_publisher"
    respawn="false" output="screen">
    <remap from="/joint_states" to="/rrbot/joint_states" />
  </node>
</launch>
```

Tras la configuración, se ejecuta la simulación del robot en Gazebo junto con su controlador. Esto se conseguiría ejecutando dos comandos consecutivos (uno que abre gazebo con el modelo 3D y otro que carga el control dentro del modelo), pero modificando el archivo que lanza el modelo de gazebo, podemos juntar ambos procesos y solo será necesario usar el comando **roslaunch kinova_gazebo robot_launch.launch kinova_robotType:=j2n6s300** y tendremos una simulación del robot controlado.

Para la integración de los controladores, ROS proporciona varios módulos de control típicos en los que solo se tienen que introducir los parámetros. Estos son:

- EffortController
- Joint_state_controllers
- Position_controllers
- Velocity_controllers
- Joint_trajectroy_controllers

Cada uno de estos módulos admite a su vez diferentes tipos de funcionamiento. En el caso del EffortController, nos permite configurarlo como entrada de Torque, de posición articular o de velocidad articular. Estos módulos inicializan diferentes topics para su funcionamiento, interactuando con diferentes archivos del sistema. Entre ellos

está los URDF que contienen parámetros físicos del robot, como pesos e inercias. Tras tratar toda la información, ROS mandará a la simulación el dato ajustado que requiere el actuador simulado para que el robot simule un comportamiento real.

Por último, necesitamos tener una serie de posiciones programadas para que el robot recogiese la carga y la almacenase en otro punto. Es en este punto donde se aplica el archivo de python **mov_ cont_ trabajo.py**, que es publicador y suscriptor y se encarga de mover el brazo por diferentes puntos. Este archivo lee dónde está el robot y manda los puntos deseados (la referencia) al generador de trayectoria integrado de kinova Ros, traduciéndose en una elección de movimiento de las articulaciones o los dedos, aunque no da la posibilidad de controlar las últimas falanges de los dedos, por los que estos cuelgan, tambaleándose debido al peso e inercia. Tras programar el script, se incluye una función para informar al usuario del error cometido, además de usar ese cálculo para asegurarnos de que el robot pase por cada punto programado.

8. Resultados

En las siguientes gráficas se muestra la referencia de cada una de las articulaciones en comparación con el valor deseado esperado de cada una de las mismas, esto se ha realizado sacando en una tabla excel los datos de la simulación y analizándolas en Matlab.

8.1. Resultados del controlador por defecto de Gazebo

Este es el controlador que proporciona KINOVA, es un controlador proporcional con ganancias de [5000,5000,5000,500,200,500] para sus articulaciones, que se introducen por su generador de controlador al .yaml de configuración. Pese a que no se especifica, leyendo el código nos podemos dar cuenta de que se introducen algunas características como el antialiasing, máximos y mínimos. Tras hacer la prueba en los scripts de posiciones, tenemos este estudio de las diferentes articulaciones.

Como se puede esperar al ser el controlador de fábrica, el robot tiene un comportamiento muy similar al deseado. Las primeras 3 articulaciones presentan problemas de seguimiento, a los que el robot responde de manera satisfactoria, siguiendo la referencia dada de forma rápida y sin alguna oscilación apreciable. Con respecto a los 3 últimos, podemos decir que el problema al que se enfrentan es de perturbaciones ya que tienen que mantenerse en la misma posición mientras el robot se mueve. Los picos corresponden a paradas y arranques de las tres primeras articulaciones, y pese a que parece que presenta mucho problema para mantenerse por darse picos muy altos, se aprecia en la escala que realmente estas articulaciones prácticamente no se mueven de sus posiciones establecidas.

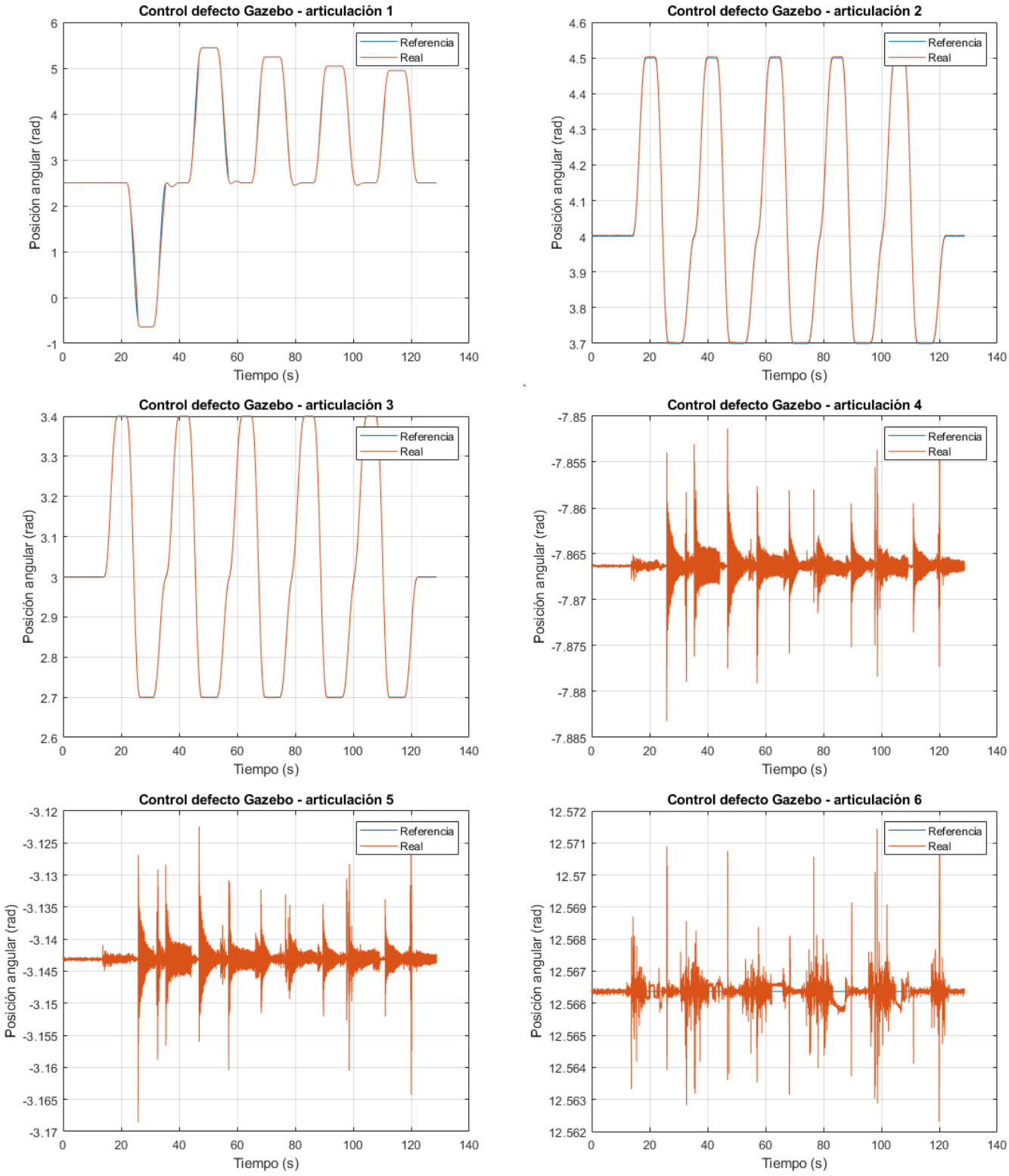


Figura 7: Gráficas del controlador por defecto de Gazebo

8.2. Resultados del controlador por Par Computado

Tras efectuar cálculos de cinemática y dinámica del robot, aplicamos procesos aprendidos en asignaturas anteriores para llegar a este controlador. El resultado será un controlador de par computado, que se procesa por ROS mediante sus ganancias Kp y Kd siendo estas de [2818 2818 2818 2818 2818 2818] y [101.38 101.38 101.38 101.38 101.38 101.38] respectivamente. Tras generar el fichero de configuración .yaml, cargamos la simulación y llegamos al estudio de las articulaciones.

Como se puede observar en las siguientes gráficas, este controlador es (en comparación con el anterior) poco fiable. Se produce una vibración constante y unos errores de seguimiento a la referencia fácilmente apreciables por el usuario.

Empezando de forma ascendente, las articulaciones 1 y 2 son las que mejor responden al proceso, seguramente por su geometría y estructura, y son capaces de responder correctamente ante el problema de seguimiento. Tras esto, empezamos a observar problemas de rendimiento, cometiendo la articulación 3 un error sistemático en el seguimiento de la referencia, al igual que las articulaciones 4 y 5, que además de cometer este error, tiene una vibración excesivamente alta debido seguramente a los problemas que ya tiene la articulación para mantenerse en la referencia junto con la vibración que hemos observado que se da en el controlador de fábrica. Y por último, la articulación 6 es la suma de todos estos fallos de seguimiento y vibraciones, que afectan al propio control de la articulación como perturbaciones, lo que resulta en está gráfica de picos y vibración.

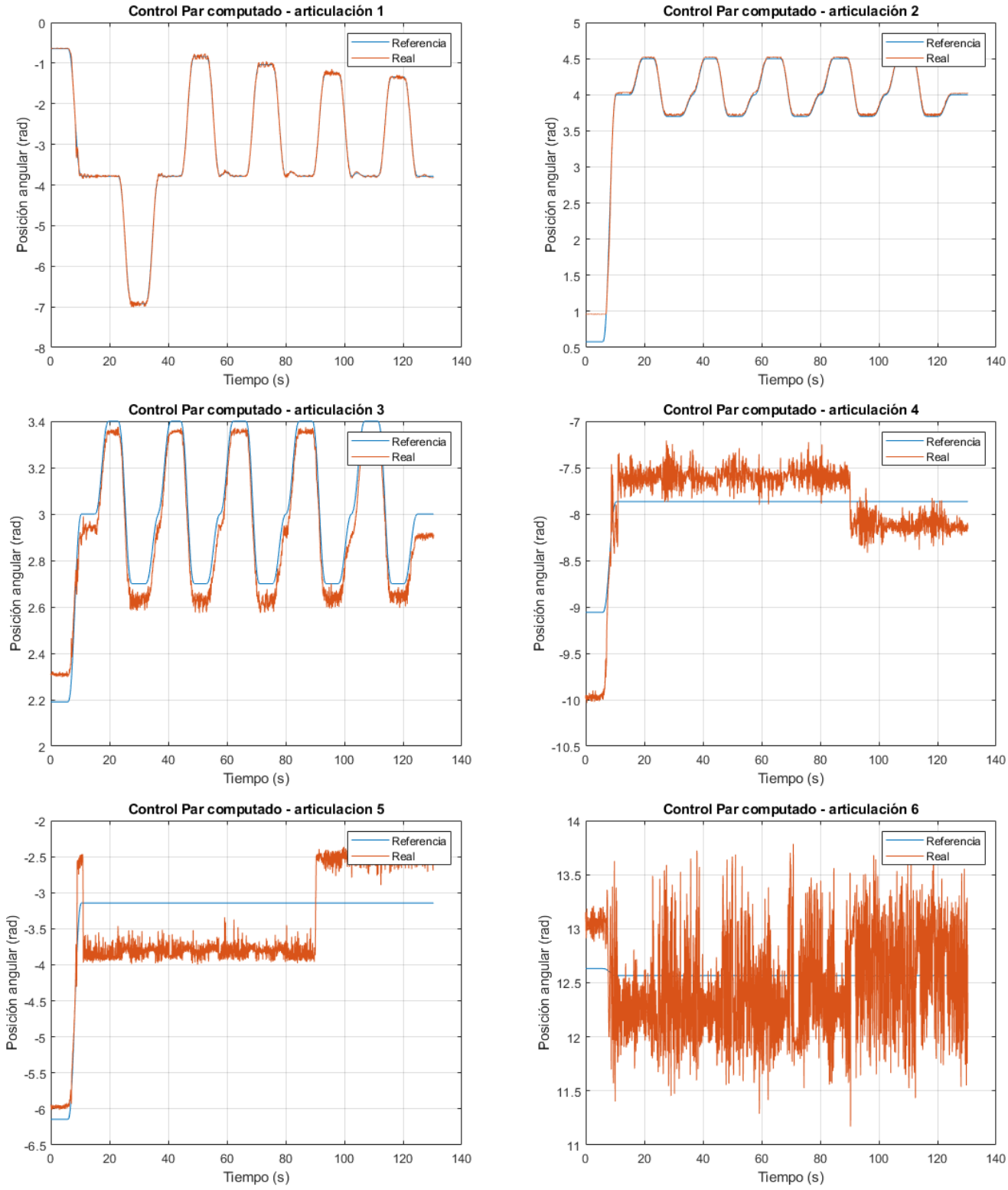


Figura 8: Gráficas del controlador por Par Computado

8.3. Resultados del controlador PID

Por último, se calcula un control PID por métodos dados en asignaturas pasadas, utilizando las funciones de transferencia propias de cada articulación y técnicas frecuenciales dando unos parámetros de Kp [146.6 2924.4 720.3 79.6 202.3 384.6], Kd [0.2198] y Ki [0.055]. Todo esto se implementa en el .yaml, que se ejecuta, y nos da el estudio articular. Teniendo en cuenta el resultado del primer controlador calculado, se observa que el robot apenas vibra y que no se están cometiendo errores sistemáticos en el seguimiento de las referencias visibles. Cuando vemos las gráficas observamos una pequeña sobreoscilación en la articulación 1, sobre la referencia al llegar al punto deseado, seguramente debido a que el robot no está compensado y la inercia que lleva debida al movimiento no se responde, pese a esto no es apreciable en los casos que nos interesan (los primeros cambios de referencia grabados son debidos a que el robot está dirigiéndose a home). Las primeras tres articulaciones tienen una respuesta ante seguimiento de referencia dentro de lo deseable. Con respecto a las 3 siguientes, vemos que su comportamiento también es bueno ante los problemas planteados y podemos ver como las pequeñas desviaciones son rápidamente rechazadas y de un orden de centésimas. Creemos que este segundo control cumple las expectativas de intentar igualar el comportamiento del robot dado por el controlador planteado de fábrica.

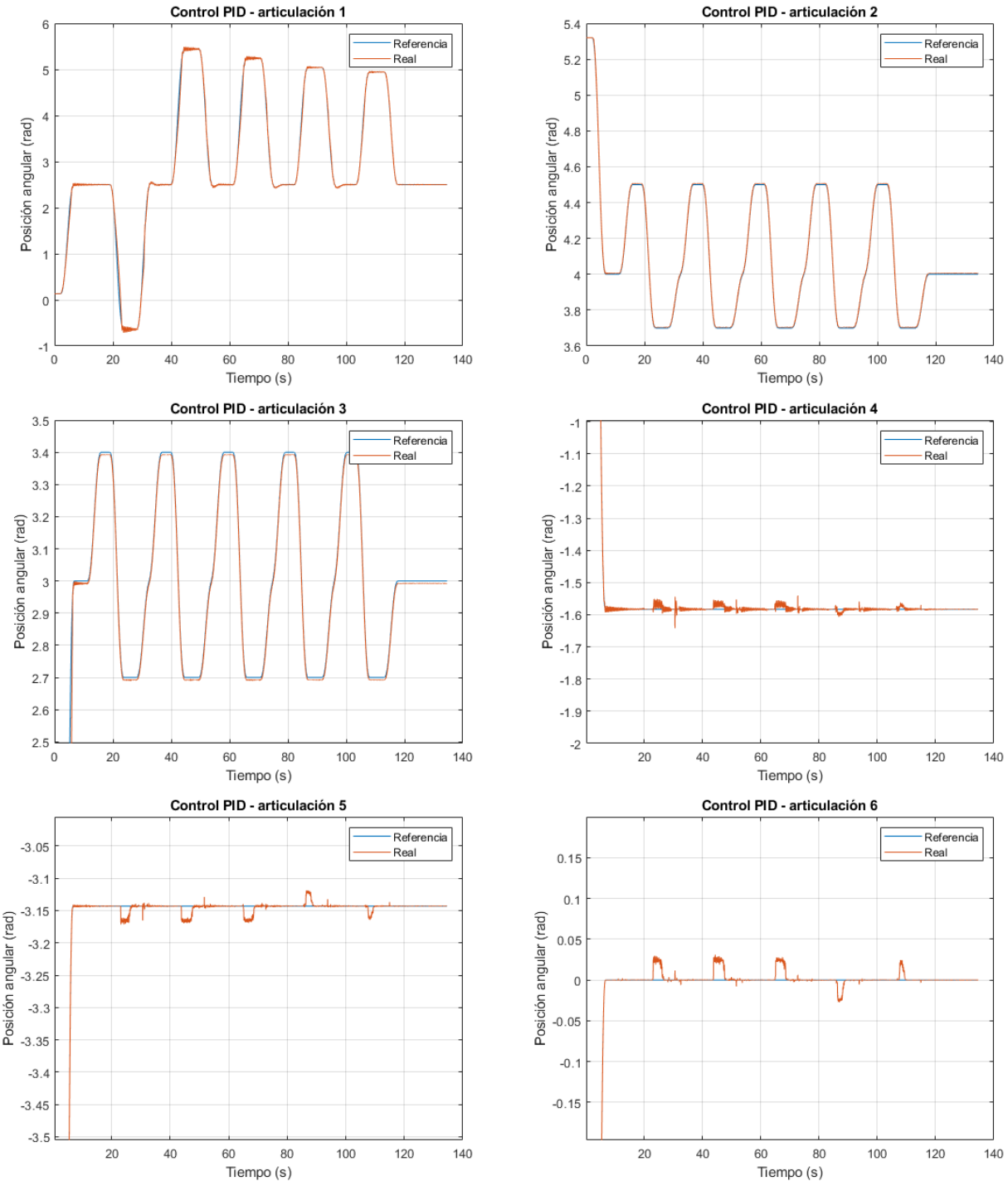


Figura 9: Gráficas del controlador PID

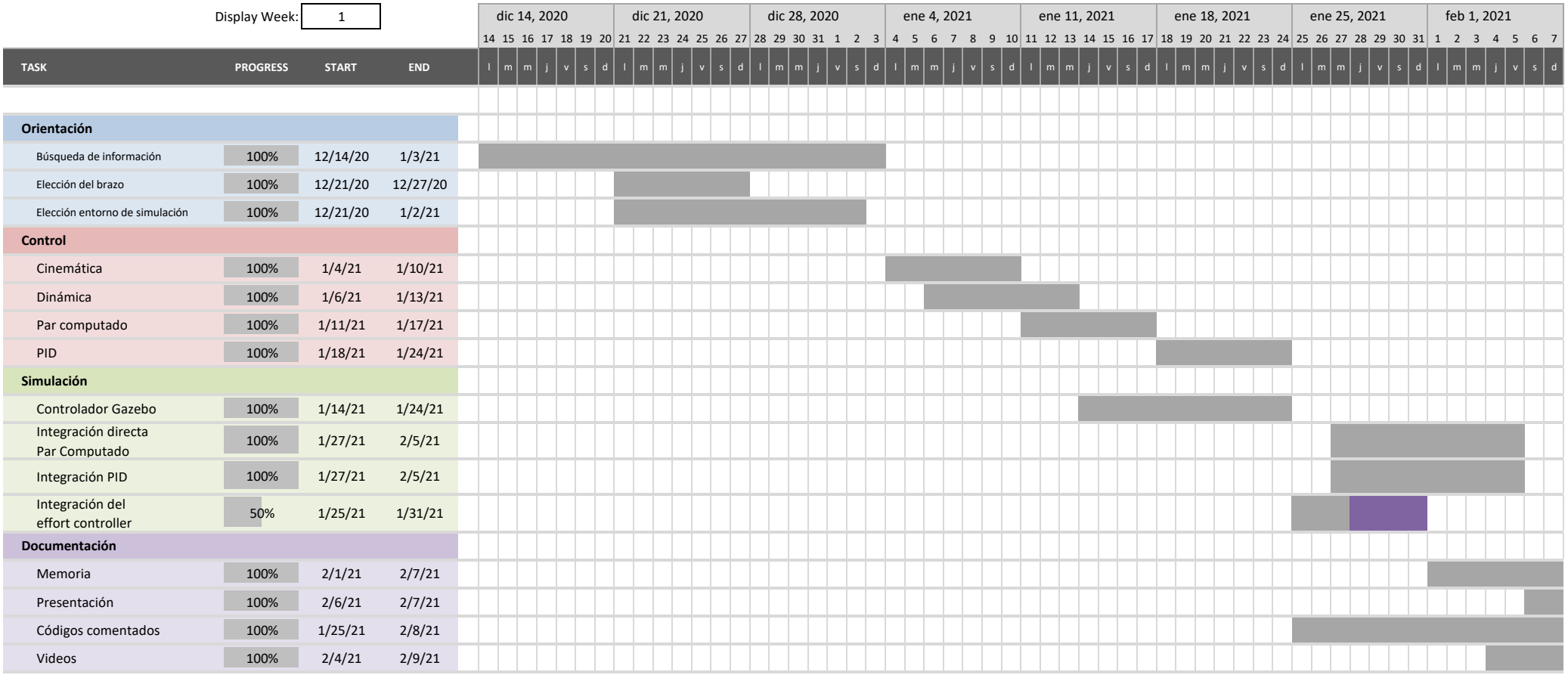
9. Plan de trabajo

Diseño de control y simulación de un brazo robot para asistencia en laboratorio

Comienzo: lu, 12/14/2020

Universidad de Sevilla

Display Week: 1



TASK	PROGRESS	START	END
Orientación			
Búsqueda de información	100%	12/14/20	1/3/21
Elección del brazo	100%	12/21/20	12/27/20
Elección entorno de simulación	100%	12/21/20	1/2/21
Control			
Cinemática	100%	1/4/21	1/10/21
Dinámica	100%	1/6/21	1/13/21
Par computado	100%	1/11/21	1/17/21
PID	100%	1/18/21	1/24/21
Simulación			
Controlador Gazebo	100%	1/14/21	1/24/21
Integración directa Par Computado	100%	1/27/21	2/5/21
Integración PID	100%	1/27/21	2/5/21
Integración del effort controller	50%	1/25/21	1/31/21
Documentación			
Memoria	100%	2/1/21	2/7/21
Presentación	100%	2/6/21	2/7/21
Códigos comentados	100%	1/25/21	2/8/21
Videos	100%	2/4/21	2/9/21

10. REFERENCIAS

Referencias

[1] Kinova Robotics. KINOVA Gen2 Ultra lightweight robot User Guide
Disponible en:https://www.kinovarobotics.com/sites/default/files/UG-009_KINOVA_Gen2_Ultra_lightweight_robot_User_guide_EN_R02.pdf

[2] Kinova Robotics. Gazebo for kinova robots.
Disponible en:https://github.com/Kinovarobotics/kinova-ros/wiki/Gazebo

[3] Kinova Robotics. $Jaco^2$ 6DOF Advanced Specification Guide

[4] Kinova Robotics. Paquete de ROS para brazos robóticos de Kinova.
Disponible en:https://github.com/Kinovarobotics/kinova-ros/blob/master/kinova_description/urdf/kinova_inertial.xacro

[5] Gazebo. Tutoriales ROS_ control
Disponible en:http://gazebosim.org/tutorials/?tut=ros_control