INSTRUCCIONES DE REALIZACIÓN DEL PROYECTO DE LA ASIGNATURA

CPR-4º GIERM - Curso 2024/25

- Cada proyecto será realizado por equipos integrados por 4 ó 5 alumnos. Aunque si está justificado se permitirán grupos con menor o mayor número de alumnos.
- Los proyectos tratarán sobre aspectos tratados en la asignatura pudiendo contener aspectos tales como:
    - construcción física de robots,
    - programación de robots,
    - control de robots,
    - diseño detallado de un sistema robótico de cierta complejidad,
    - revisión del estado del arte sobre los temas tratados en la asignatura

  En cualquier caso, el tema del proyecto deberá ser validado por el Responsable de la Asignatura.
- Una vez confeccionado un grupo y elegido un tema, se deberá solicitar al Responsable de la Asignatura mediante un informe de un mínimo de una página que se enviará mediante correo electrónico a jdedios@us.es con **asunto "Proyectos CPR"** indicando:
    - Integrantes del grupo,
    - Números de los trabajos elegidos por orden de prioridad,
- Se recomienda desarrollar el proyecto en ROS u otras herramientas de tipo profesional. Sólo se permitirá realizar el proyecto en Matlab de forma excepcional y bajo autorización expresa del Responsable de la Asignatura.
- Se dedicarán clases para solucionar dudas de los trabajos y supervisar su avance.
- El trabajo será entregado con fecha límite el día del examen de la asignatura. Se deberá entregar breve memoria con la descripción de los trabajos realizados, códigos, simulaciones y un Powerpoint del proyecto que permita su presentación en 15 minutos (máximo 13 transparencias). En todos los casos donde sea posible, se deben incluir vídeos que muestren el funcionamiento del desarrollo realizado en el trabajo.

Temas de trabajos

1. Simulación y control de robot PUMA

Se trata de encontrar o desarrollar el modelo de un robot PUMA, implementar el método de control de par computado y simular el método de control con diferentes referencias y errores de modelado para evaluar su comportamiento.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

2. Simulación y control de robot SCARA

Se trata de encontrar o desarrollar el modelo de un robot SCARA, implementar el método de control de par computado y simular el método de control con diferentes referencias y errores de modelado para evaluar su comportamiento.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

3. Simulación y control de robot ABB IRB120

Se trata de encontrar o desarrollar el modelo de un robot ABB IRM120, implementar el método de control de par computado y simular el método de control con diferentes referencias y errores de modelado para evaluar su comportamiento.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

4. Simulación y control de un robot con ruedas con configuración Ackerman

Se trata de encontrar o desarrollar el modelo de un robot con ruedas de configuración Ackerman, implementar el método de control de persecución pura (“pure pursuit”) y simular el método de control con diferentes trayectorias y errores de modelado para evaluar su comportamiento.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

5. Simulación y control de un robot con ruedas con configuración diferencial

Se trata de encontrar o desarrollar el modelo de un robot con ruedas de configuración diferencial, implementar el método de control de persecución pura y simular el método de control con diferentes trayectorias y errores de modelado para evaluar su comportamiento.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

6. Simulación y control de un robot con pistas

Se trata de encontrar o desarrollar el modelo de un robot con pistas de deslizamiento, implementar el método de control de persecución pura y simular el método de control con diferentes trayectorias y errores de modelado para evaluar su comportamiento.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

7. Simulación y control de quadrotor

Se trata de encontrar o desarrollar el modelo de un quadrotor, investigar en métodos de control de quadrotor, implementar el método elegido y simular el método de control con diferentes trayectorias deseadas que se le dan como referencia. Se empleará el simulador 3D Gazebo.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

8. Simulación y control de helicóptero

Se trata de encontrar o desarrollar el modelo de un helicóptero, investigar en métodos de control de helicópteros, implementar el método elegido y simular el método de control con diferentes trayectorias deseadas que se le dan como referencia. Se empleará el simulador 3D Gazebo.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

9. Simulación y control de robot submarino

Se trata de encontrar o desarrollar el modelo de un submarino, investigar en métodos de control de submarinos, implementar el método elegido y simular el método de control con diferentes trayectorias deseadas que se le dan como referencia. Se empleará el simulador 3D Gazebo.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

10. Simulación y control de robot marino de superficie

Se trata de encontrar o desarrollar el modelo de un robot marino de superficie (USV), investigar en métodos de control de USVs, implementar el método elegido y simular el método de control con diferentes trayectorias deseadas que se le dan como referencia. Se empleará el simulador 2D Gazebo.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

11. Simulación y control de robot ROSbot 2.0 mediante método "*pure pursuit*"

Se trata de encontrar o desarrollar el modelo del robot ROSbot 2.0, investigar en métodos de control adecuados para este robot, implementar el método elegido y simular el método de control "pure pursuit" con diferentes trayectorias deseadas que se le dan como referencia. Se empleará el simulador 2D Gazebo. Se valorará la realización de experimentos con los robots físicos.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

12. Simulación de evitación de obstáculos con robot ROSbot 2.0

Se trata de encontrar o desarrollar el modelo del robot ROSbot 2.0, investigar en métodos de detección y evitación de obstáculos adecuados para este robot, implementarlos y evaluarlos en simulación mediante Gazebo. Se valorará la realización de experimentos con los robots físicos.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

13. Simulación, planificación de trayectorias y control de robot manipulador

Se trata de implementar el esquema completo de planificación y control de un robot manipulador a elegir. El método de control será el método de par computado. Se evaluará con simulaciones.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

14. Simulación, planificación de trayectorias y control de robot móvil

Se trata de implementar el esquema completo de planificación y control de un robot móvil a elegir. El método de control será "pure pursuit". Se evaluará con simulaciones.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

15. Comparación de métodos de control de robots manipuladores

Se trata de implementar diversos métodos de control de robots manipuladores y compararlos mediante simulación.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

16. Modelado, simulación y control de robot manipulador con plataforma móvil

Se trata de encontrar o desarrollar el modelo de un robot móvil con un brazo manipulador, investigar en métodos de control adecuados, implementar el método elegido y simular el método de control con diferentes trayectorias deseadas que se le dan como referencia tanto a la plataforma móvil como al brazo manipulador. Se empleará el simulador 3D Gazebo

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

17. Modelado, simulación y control de robot manipulador paralelo

Se trata de encontrar o desarrollar el modelo de un robot manipulador de configuración paralela, implementar el método de control por par computado y simular el método de control con diferentes trayectorias deseadas que se le dan como referencia.

Plataforma de desarrollo: ROS
Lenguaje: ROS, C/C++ o Python

18. Modelado, simulación y control con MuJoCo de manipulador bi-brazo LiCAS A1

Se trata de implementar en C/C++ o Python un simulador del robot manipulador ligero bi-brazo LiCAS A1 usando el motor físico MuJoCo, partiendo del modelo 3D de los brazos y de los parámetros físicos de masa e inercia. Se implementará un controlador de posición articular y se visualizará la generación de trayectorias de los brazos.

Plataforma de desarrollo: Ubuntu/MuJoCo
Lenguaje: XML, C/C++ o Python