PRÁCTICA 3

En esta practica se llevará acabo el análisis de las posibles librerías que se usaran en nuestro brazo robótico cilíndrico, explicando de forma breve de como se implementa su uso en el robot.

Las primeras librerías son las visualizaciones para nuestro sistema del brazo, estas se encargan de realizar las simulaciones y desplazamientos que se generan en el brazo robótico, y también de los nodos que ejercerán dicho movimiento, nuestras posibles librerías son, open\_manipulator\_description: este visualiza y simula, open\_manipulator\_gazebo, que es el paquete de configuraciones de Gazebo para OpenManipulator, RVIZ: esta herramienta realiza la visualización en 3D para Ros y funcionaria como interface para la comunicación de otras librerías, Agni\_tf\_tools, funciona como un complemento para la librería Rviz que le permite configurar la transformación con un marcador interactivo. Junto con graph\_rviz\_plugin como complemento para los gráficos que serán dibujados a partir de los distintos valores de temas. Se agrega el cartographer\_rviz, que nos proporcionaría la localización y el mapeo simultáneos en tiempo real en 2D y 3D a través de múltiples plataformas y configuraciones. Se utilizará la librería graph\_msgs, para los mensajes ROS para la publicación de los gráficos de diferentes tipos de datos.

Ahora definimos las librerías que usaremos en los movimientos, desplazamientos, los nodos, todo movimiento ejercido en el robot. Primero especificamos las librerías de comunicación para las distintas plataformas que se usaran en el robot, ecl\_filesystem, que se implementará para la utilización del sistema de archivos multiplataforma, ros\_control\_boilerplate, este actuara como interfaz de simulación simple entre las simulaciones y plantillas para configurar la interfaz del hardware para el Ros, ecl\_threads, esta librería nos proporciona extensiones de c++ para la variedad de herramientas que se programaran de los subprocesos, ejerciendo la arquitectura para un marco multiplataforma, ecl\_utilities, este nos permitirá varias herramientas de soporte y utilidades para programar.

Una vez especificados, se implementan las librerías de los movimientos, dbw\_mkz\_twist\_controller, esta librería nos permitirá controlar el giro tanto la velocidad como la velocidad angular, también el freno, aceleración, dirección, Angles, es la librería que se designaría para la parte matemática sencilla que trabajará con los ángulos y la conversión entre grados y radianes, algunas como las distancias, move\_slow\_and\_clear, este nos permite realizar pruebas del funcionamiento a una velocidad bastante baja y limpiar los parámetros plasmados en las pruebas, global\_planner, este se encargara de la planificación de ruta y el movimiento de los nodos.

Dentro de estas librerías tenemos que especificar algunas que se encarguen de los errores del robot y los envíos de mensajes de los procesos que esta realizando el robot, para ello podemos empezar con la librería behaviortree\_cpp\_v3, que nos permite proporcionar la biblioteca principal de árboles de comportamiento, para después implementar la librería behaviortree\_cpp, que nos permite crear un núcleo de comportamiento, robot\_activity\_msgs, este se encargara de marcar el estado del robot, y los errores en los nodos. Controller\_manager, como lo indica el mismo nombre, se encargará de dirigir los controladores que se implementará en el robot, junto con controller\_interface, verificando la interfaz de los controladores, control\_toolbox, que tiene la herramienta de controlar los módulos que son útiles en todos los controladores, con bondcpp para verificar cuando ha finalizado otro proceso y Enlace, como un bono en el cual permite que dichos procesos “A” y “B” puedan conocer cuando a terminado uno del otro, ya sea de forma limpia o por algún fallo. También se utilizará la librería exotica\_collision\_scene\_fcl\_lastest, se utilizará para la verificación de los cálculos y las posibles colisiones en el sistema del robot.

Una de las formas en las que se pueden comunicar con el usuario, esto se puede realizar con fkie\_message\_filters, que son la forma mejorada en la que se comunican los mensajes de ROS que demuestra una gran eficiencia.

Con lo anteriormente mencionado podremos utilizar control\_msgs nos proporcionará los puntos de ajuste del control, las trayectorias conjuntas y cartesianas, con ello se utilizará la librería calibration\_estimation, ejecuta una optimización para estimar los parámetros cinemáticos del brazo robótico y con ello usar calibration\_launch ya que este contiene una colección de archivos de inicio que pueden ser útiles para configurar la pila de calibración que se ejecutaran en el robot.