Modelado y Simulación de Robots - GIRS

Práctica 3: Simulación de Robots usando middleware

Anexo: Configuración de los controladores

En este documento se muestran fragmentos de código para definir la configuración específica de los controladores. Este fichero será utilizado por el plugin de ROS 2 control.

Configuración principal de `controller_manager`

El `controller_manager` es un componente de ROS 2 que gestiona y coordina los diferentes controladores del robot. Se debe especificar el `update_rate`, es decir, cuántas veces por segundo se ejecuta el ciclo de control. En este componente se debe registrar cada controlador:

- Para la base móvil, `DiffDriveController` es el controlador estándar para robots con dos ruedas motrices (izquierda y derecha). Maneja comandos de velocidad lineal y angular.
- `JointStateBroadcaster` publica en el tópico `/joint_states` información de las articulaciones como posición, velocidad y esfuerzo (fuerza y par).
- Para el brazo y la pinza, `JointTrajectoryController` recibe trayectorias planeadas (por ejemplo de MoveIt) y mueve las articulaciones del grupo.

```yaml
controller_manager:
  ros__parameters:
    update_rate: 20 # Hz
    rover_base_control:
        type: diff_drive_controller/DiffDriveController
    joint_state_broadcaster:
        type: joint_state_broadcaster/JointStateBroadcaster
    scara_controller:
        type: joint_trajectory_controller/JointTrajectoryController
    gripper_controller:
        type: joint_trajectory_controller/JointTrajectoryController
```

Definir en detalle cada controlador

Para la base, se debe especificar de nuevo el tipo de controlador (`DiffDriveController`) y parámetros relativos al uso de marca de tiempo en la velocidad, la tasa de publicación (en este caso de la odometría) y si se desea usar el timpo de simulación.

En este ejemplo se utiliza `use_stamped_vel: false` ya que en el lauch se republica la velocidad añadiendo esta información. Para la práctica, se debe utilizar `use_sim_time: true` para asegurar una correcta sincronización con Gazebo.

Además, se deben especificar los parámetros de las ruedas como cuáles actúan como ruedas izquierdas y derechas, distancia entre las ruedas izquierda y derecha y tamaño de cada rueda. Finamente, se definen multiplicadores para ajustar el sentido o pequeñas correcciones de hardware. Para invertir el sentido de giro, se usa el multiplicador -1.0.

```yaml
/**/rover_base_control:
  ros__parameters:
      type: diff_drive_controller/DiffDriveController
      use_stamped_vel: false
      use_sim_time: true
      publish_rate: 50.0
      left_wheel_names: ["front_left_wheel_joint", "back_left_wheel_joint"]
      right_wheel_names: ["front_right_wheel_joint", "back_right_wheel_joint"]
      wheel_separation: 1.90
      wheel_radius: 0.30
      wheel_separation_multiplier: 1.0
      left_wheel_radius_multiplier: 1.0
      right_wheel_radius_multiplier: -1.0
```

Además es necesario añadir el marco de referencia para el robot y especificar si se desea publicar la transformación `odom -> base_footprint`. Finalmente, se purfr limitar la velocidad de las ruedas tanto lineal como angular.

```yaml
      # Base frame_id
      base_frame_id: base_footprint
      # odom tf will be published by direct laser odometry
      enable_odom_tf: true
      # Publish limited velocity
      publish_limited_velocity: true
      linear:
          x:
              has_velocity_limits: true
              max_velocity: 3.0
              has_acceleration_limits: true
              max_acceleration: 3.0
      angular:
          z:
            has_velocity_limits: true
            max_velocity: 3.0
            has_acceleration_limits: true
            max_acceleration: 3.0
```

En el caso de los parámetros del brazo y la pinza, se pueden copiar los generados por *Moveit Setup Assistant.* El uso de /**/ no es necesario, simplemente permite cambiar el namespace que resulta de gran utidad en sistemas multirobot.