1. Crear el espacio de trabajo

Primero, creamos el espacio de trabajo para ROS 2. Nos aseguramos de estar en nuestro directorio home:

```
C/C++
cd ~
mkdir -p prueba/src
cd prueba
colcon build
source install/setup.bash
```

---

2. Crear el paquete de interfaces (Servicio)

Este paquete define el **servicio** (`DesiredPosition.srv`) que usaremos para enviar y recibir la posición deseada.

Ubicación: **`~/prueba/src`**
Ejecutamos:

```
C/C++
cd ~/prueba/src
ros2 pkg create cinematica_inversa_pkg --build-type ament_cmake
cd cinematica_inversa_pkg
mkdir srv
cd srv
touch DesiredPosition.srv
```

📄 **Editar** **`DesiredPosition.srv`**
Editamos el archivo con:

```
C/C++
code .
```

Agregamos la siguiente definición:

```
C/C++
# Solicitud (Request): Posición deseada (x, y, z)
float64 x
float64 y
float64 z
---
# Respuesta (Response): Posición procesada (o valores de error si no es
posible)
float64 x
float64 y
float64 z
```

Ubicación: `~/prueba/src/cinematica_inversa_pkg`
Ahora editamos `CMakeLists.txt` y agregamos:

Editamos el archivo con:

```
C/C++
code .
```

Añadimos lo siguiente:

```
C/C++
find_package(rosidl_default_generators REQUIRED)
rosidl_generate_interfaces(${PROJECT_NAME}
  srv/DesiredPosition.srv
)
```

Ubicación: `~/prueba/src/cinematica_inversa_pkg`
Editamos `package.xml` para incluir dependencias:

Editamos el archivo con:

```
C/C++
code .
```

Agregamos:

```
C/C++
<build_depend>rosidl_default_generators</build_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<member_of_group>rosidl_interface_packages</member_of_group>
```

Ubicación: `~/prueba`
Compilamos y verificamos:

```
C/C++
cd ~/prueba
colcon build --packages-select cinematica_inversa_pkg
source install/setup.bash
ros2 interface show cinematica_inversa_pkg/srv/DesiredPosition
```

Debe mostrar:

```
C/C++
float64 x
float64 y
float64 z
---
float64 x
float64 y
float64 z
```

---

3. Crear el paquete de nodos

Ubicación: `~/prueba/src`
Ahora creamos el paquete que contendrá nuestros nodos:

```
C/C++
cd ~/prueba/src
ros2 pkg create cinematica_inversa --build-type ament_python --dependencies
rclpy cinematica_inversa_pkg
cd cinematica_inversa
```

---

4. Crear el Nodo de Servicio (`nodo_servicio.py`)

Ubicación: `~/prueba/src/cinematica_inversa/cinematica_inversa`
Creamos el archivo:

```
C/C++
touch nodo_servicio.py
chmod +x nodo_servicio.py
nano nodo_servicio.py
```

Código de `nodo_servicio.py`

Editamos el archivo con:

```
C/C++
code .
```

Agregamos:

```
Python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from cinematica_inversa_pkg.srv import DesiredPosition

class ServicioPosicion(Node):
    def __init__(self):
        super().__init__('nodo_servicio')
        self.cli = self.create_client(DesiredPosition,
'desired_position')

        while not self.cli.wait_for_service(timeout_sec=1.0):
            self.get_logger().info('Esperando el servicio de cinemática
inversa...')

    def send_request(self, x, y, z):
        request = DesiredPosition.Request()
        request.x = x
        request.y = y
        request.z = z

        future = self.cli.call_async(request)
        rclpy.spin_until_future_complete(self, future)
        return future.result()
```

```
def main():
    rclpy.init()
    node = ServicioPosicion()

    while True:
        try:
            x = float(input("Ingrese x: "))
            y = float(input("Ingrese y: "))
            z = float(input("Ingrese z: "))

            response = node.send_request(x, y, z)

            if response.x != -1 and response.y != -1 and response.z !=
-1:
                node.get_logger().info(f"Posición alcanzable:
({response.x}, {response.y}, {response.z})")
            else:
                node.get_logger().info("Posición NO alcanzable. Intente
nuevamente.")

        except ValueError:
            print("Ingrese valores numéricos.")

    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

---

5. Crear el Nodo de Cinemática Inversa (`nodo_cinematica.py`)

Ubicación: `~/prueba/src/cinematica_inversa/cinematica_inversa`
Creamos el archivo:

```
C/C++
touch nodo_cinematica.py
chmod +x nodo_cinematica.py
nano nodo_cinematica.py
```

Editamos el archivo con:

```
C/C++
code .
```

Código de `nodo_cinematica.py`

```python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from cinematica_inversa_pkg.srv import DesiredPosition
from std_msgs.msg import Float64MultiArray

class CinematicaInversa(Node):
    def __init__(self):
        super().__init__('nodo_cinematica')

        # Crear el servicio
        self.srv = self.create_service(DesiredPosition,
'desired_position', self.ik_callback)

        #Crear el publicador
        self.publisher = self.create_publisher(Float64MultiArray,
'desired_joint', 10)

        self.get_logger().info("Nodo de Cinemática Inversa Listo.")

    def ik_callback(self, request, response):
        x, y, z = request.x, request.y, request.z

        resultado = x**2 + y**2 + z**2  # Fórmula de validación

        if resultado < 20:
            response.x = x
            response.y = y
            response.z = z

            #Publicar
            msg = Float64MultiArray()
            msg.data = [x, y, z]
            self.publisher.publish(msg)
            self.get_logger().info(f"📢 Publicado en /desired_joint:
{msg.data}")
        else:
            response.x = -1
```

```
            response.y = -1
            response.z = -1
            self.get_logger().info(f"Posición ({x}, {y}, {z}) NO
alcanzable. No se publica en /desired_joint.")

        return response

def main():
    rclpy.init()
    node = CinematicaInversa()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

---

6. Configurar `setup.py`

Ubicación: **`~/prueba/src/cinematica_inversa`**
Editamos `setup.py`:

Editamos el archivo con:

```
C/C++
code .
```

Agregamos:

```
Python
entry_points={
    'console_scripts': [
        'cinematica_inversa_server =
cinematica_inversa.nodo_cinematica:main',
        'cinematica_inversa_client =
cinematica_inversa.nodo_servicio:main',
    ],
```

```
},
```

---

7. Compilación y Ejecución

Ubicación: `~/prueba`
Compilamos y cargamos la configuración:

```
C/C++
cd ~/prueba
colcon build --packages-select cinematica_inversa
source install/setup.bash
```

Ejecutamos el **servidor**:

```
C/C++
ros2 run cinematica_inversa cinematica_inversa_server
```

Ejecutamos el **cliente**:

```
C/C++
ros2 run cinematica_inversa cinematica_inversa_client
```