Taller 3, Diseño de un robot manipulador - IELE3338

Santiago Rodríguez Ávila - *s.rodrigueza@uniandes.edu.co*
Sergio Andrés Guerra Sánchez - *sa.guerra20@uniandes.edu.co*
Santiago Muñoz Correa- *s.munozc@uniandes.edu.co*
Isabela Buitrón Burbano - *i.buitron@uniandes.edu.co*

Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica

07 de mayo de 2022

Introducción

En este taller se presenta el diseño e implementación de un robot manipulador que puede tomar un objeto y comunicarse a través de ROS. Este manipulador se ubicará en la parte superior del robot diferencial previamente diseñado (Taller 2).

1. Diseño del Robot

Cantidad	**Componente**	**Referencia**	**Precio und. ($COP)**
4	Servomotor	MG995	$11.000
1	Chasis Brazo Robótico	SG90	$39.000
N/A	Tornillos	Varios	$4.000
1	Arduino Mega	Mega2590	$80.000
1	Raspberry Pi	Pi 4	N/A
1	Protoboard	ACC_0429	$6.000
2	Módulo regulador	LM2596	$8.700
3	Batería Lipo 5000MAH 7.4V		N/A
1	Cámara web STAR TEC HC-71	HC-71	$60.000
			$256.400

Para la alimentación del robot se usan varias (3) baterías lipo 5000MAH 7.4V con reguladores para los servomotores del manipulador y alimentación de la tarjeta Raspberry. Se incluyeron algunos elementos en la lista de materiales sin precio pues se tenían desde el anterior taller. Se incluyeron las referencias de ciertos componentes dentro de la bibliografía al final del presente documento. En la Figura 2 de la sección 1.2 Plano Mecánico se puede visualizar la estimación de espacio que ocupa la lista de materiales dentro del chasis diseñado para el robot y el espacio para el manipulador.

1.1. Plano Mecánico:

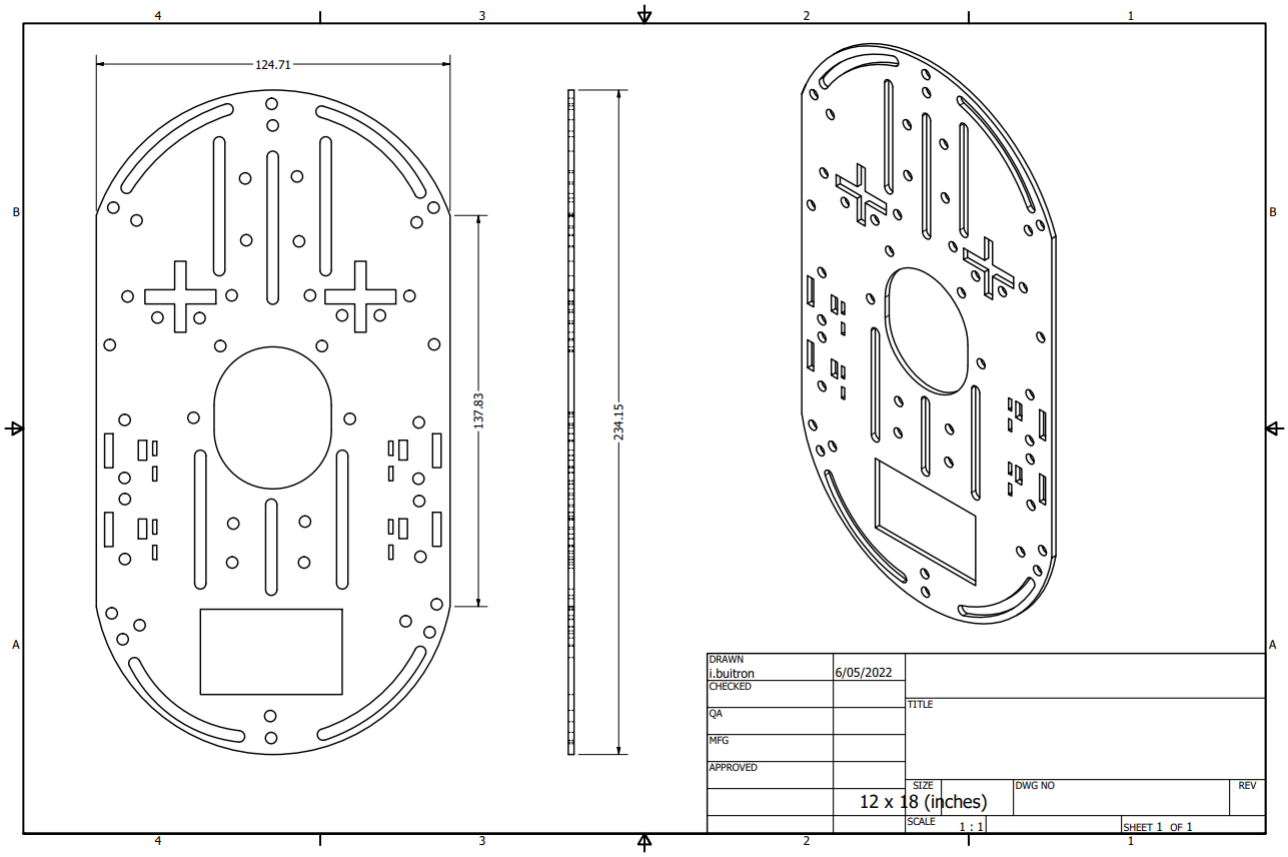

Figura 1: Plano mecánico chasis robot.

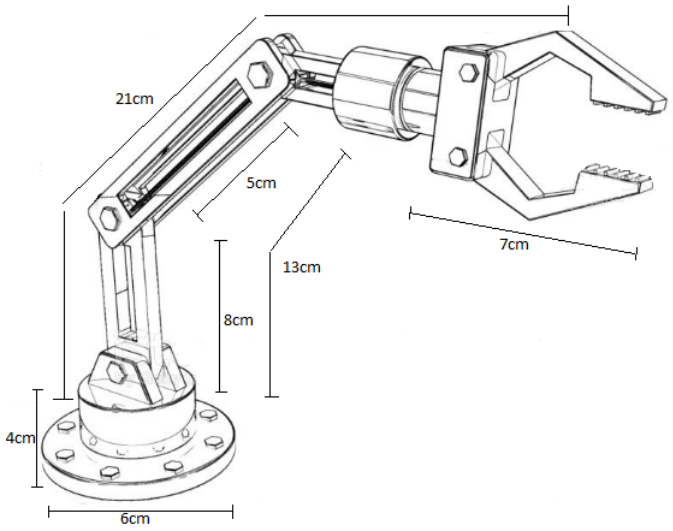

Figura 2: Plano manipulador.

Se amplió la superficie del chasis del Taller 2 para poder soportar el manipulador y tener más espacio para los componentes.

1.2. Plano Eléctrico/Electrónico:

A continuación se encuentra el plano eléctrico/electrónico de las conexiones entre los componentes solo para el manipulador. Aunque no se incluyó en el esquemático, pues no es estrictamente necesario, se hizo uso del regulador LM2596 cuando se alimenta el robot por baterías.

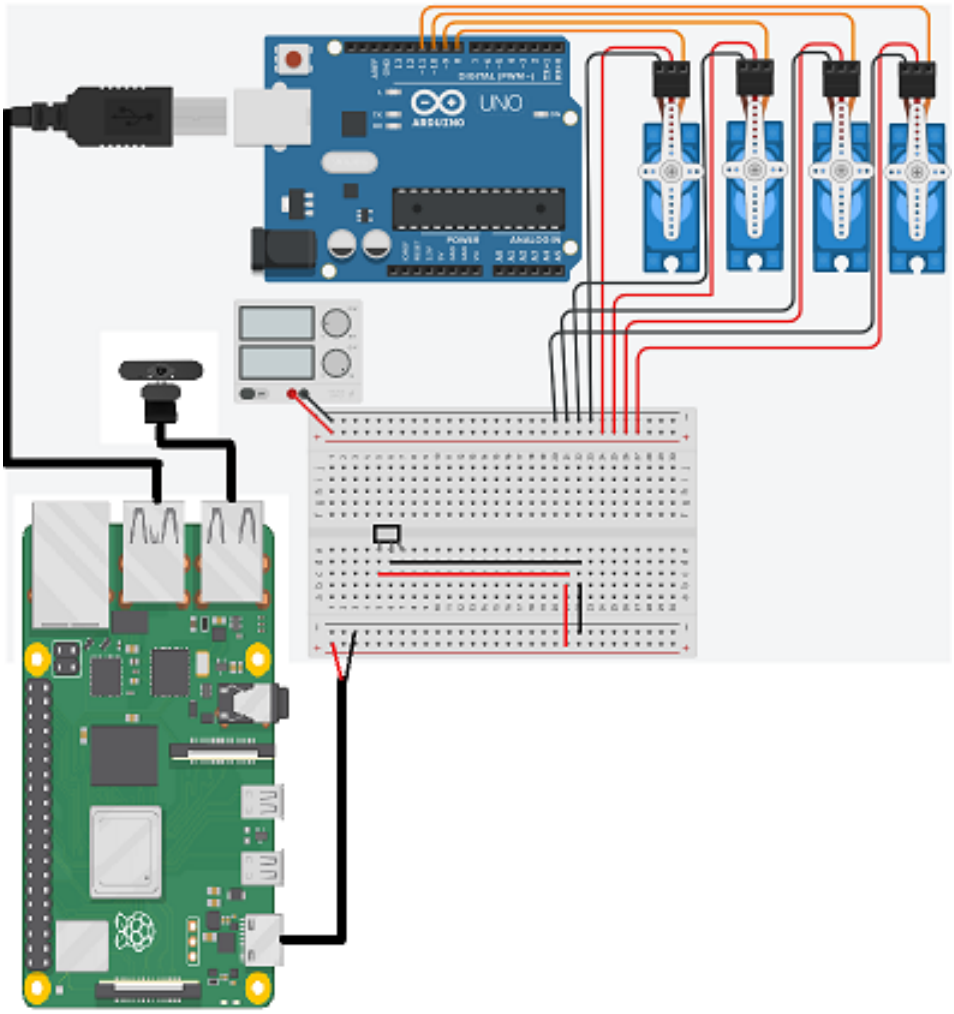

Figura 3: Plano Eléctrico

1.3. Integración Detallada de la Parte Eléctrica y Mecánica:

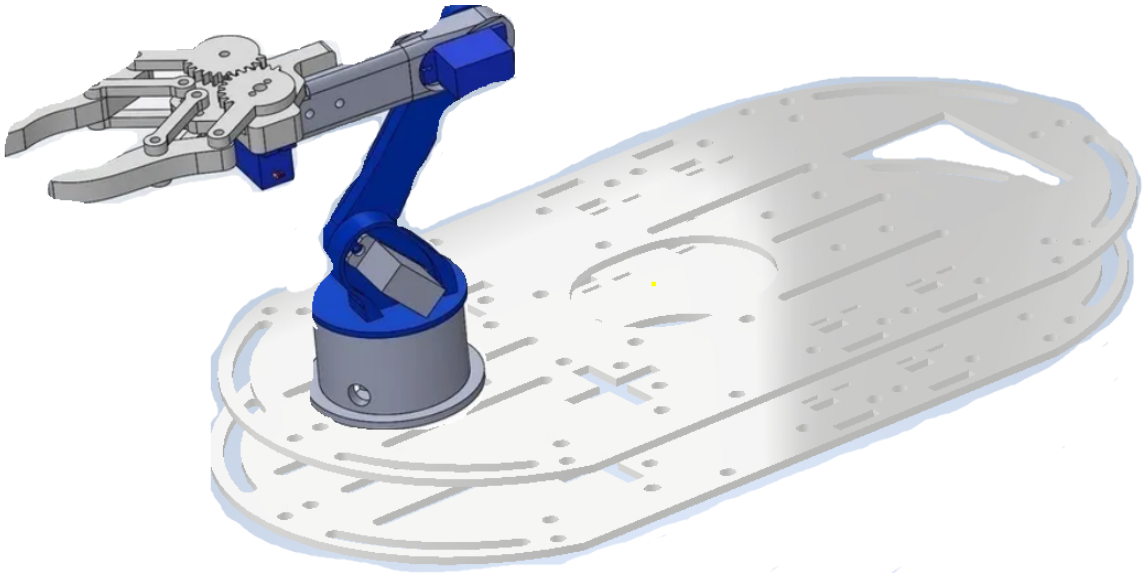

Figura 4: Vista simulada del robot físico (chasis).

Se realizó una vista simulada del componentes que se agregaron en este taller, como el manipulador. Se observa el chasis ampliado y los servomotores que se encuentran en el brazo robótico. Para integrar la parte eléctrica y mecánica de la solución se calibró y caracterizaron los servomotores de tal manera que el brazo tuviera un movimiento controlado para lograr los objetivos del taller 3. La parte mecánica, el chasis, se integra con la parte eléctrica en los orificios que se tuvieron en cuenta para pasar cables entre pisos. La conexión entre pisos se realizó con espaciadores modulares para ajustar los pisos del chasis al espacio de altura necesario teniendo en cuenta la superficie de donde se tomarán los ping pongs.

1.4. Descripción del Funcionamiento del Robot:

El robot realizado para el desarrollo de este taller consiste de dos motores DC con sus llantas, con los mismos componentes del propuesto para el taller 2 pero con el chasis más alargado para poder soportar el manipulador. Se utilizó Arduino Mega para realizar las tareas de movimiento del manipulador, dado que los servomotores pueden requerir pines PWM se consideró necesario hacer la transición de Arduino Uno a Arduino Mega para facilitar la adición de más componentes a la solución. Se hizo la integración Arduino - ROS a través del nodo /serial_node. Como se detallará en la siguiente sección de descripción de los nodos, el manipulador tiene 4 grados de libertad (uno por cada servomotor) y se mueve 180°. Las teclas *u y j* mueven la base del robot, la cual soporta la estructura y la mueve junto con la cámara para realizar el punto 4. Las teclas *i e k* mueven la juntura entre la base y el primer link 180°. Las teclas *o y

l* mueven la juntura que une el link con la pinza. Por último, las teclas *p y ñ* mueven la pinza. Los pares de teclas para el movimiento de los servomotores corresponden a la acción de ir hacia adelante y atrás. Se controla el movimiento con las teclas que ingresan por parámetro y es posible asignar una velocidad diferente a cada servomotor, también por parámetro. Es posible conocer la posición del end-effector en tiempo real (punto en la mitad de la pinza) a través de una gráfica 3D en la cual es posible cambiar el nombre por el usuario y se realiza una gráfica en tiempo real de su posición conforme se mueve. Para su funcionamiento, se usa una batería lipo 5000MAH 7.4V con regulador para la alimentación de la tarjeta Raspberry y para los servomotores se usa una batería lipo de 800mAh 7.4V.

2. Solución

2.1. Desarrollo de los Puntos

2.1.1. Explicación de nodos

- **Punto 1:**
  Este punto pide crear un nodo llamado */robot_manipulator_teleop*, el cual permite al usuario controlar el manipulador sobre el robot con las velocidades de cada una de las junturas que tiene el manipulador (entran por parámetro del usuario) con las teclas del computador. Si no se presiona ninguna tecla el manipulador se debe quedar quieto.

  Para ejecutar la solución del punto 1 se deben abrir tres terminales. En una terminal ejecutar el comando *roscore* y en la otra *rosrun rosserial_python serial_node.py /dev/ttyACM0.* Estando en una terminal se entrará en el ws (workspace) y se ejecutarán las siguientes líneas: source devel/setup.bash para otorgar permisos de ejecución seguido de rosrun turtle_bot_3 turtle_teleop.py (este último es el archivo utilizado para iniciar el funcionamiento de los servomotores desde Arduino y comunicarlo con ROS). Lo anterior abre el paquete del robot de grupo 3 e inicializa el nodo de teleop solicitado. En la terminal aparecerán los mensajes: *Ingrese la velocidad servo 1, 2, 3 y 4*, siendo 1 el servo *base*. Como se indica, en cada espacio se debe ingresar la velocidad lineal deseada para cada servomotor seguido de la tecla enter.

  Luego de esto ya debería ser posible presionar las teclas u, j, i, k, o, l p, ñ para mover el manipulador. Las parejas de comandos corresponden a movimientos hacia un lado y otro de los servos. Los servos están ubicados en la junturas del manipulador y la velocidad de cada juntura está en el marco de referencia local de cada una de ellas al iniciar el nodo. Lo anterior es posible por el uso de la librería pynput la cual lee el teclado y cada vez que se oprima una de las teclas de movimiento,

el nodo teleop publica en el tópico robot_manipulator_angles el valor que ingresa por parámetro. Ejecutando el comando *rostopic info robot_manipulator_angles* podemos ver que el tipo de mensaje es *std_msgs/Int8*. La razón de lo anterior es debido a que el funcionamiento del movimiento del manipulador consiste en que Arduino reciba las letras del teclado y las traduce a movimiento en los motores, dentro del código de Arduino las letras mencionadas se traducen a números como "1", "2", entre otros, que corresponden a los movimientos de ir hacia adelante o atrás y envían a los servomotores el valor PWM correspondiente. Se realiza la traducción con un mensaje de tipo Int8, se usan mensajes tipo Twist para robot_manipulator_angles pues tiene un componente lineal para las velocidades (x,y,z) y un componente angular en los ejes (x,y,z). En un terminal con el workspace de robótica abierto se escribe el comando *rqt_graph*. rqt_graph proporciona un complemento GUI (interfaz de usuario gráfica) para visualizar el gráfico de cálculo de ROS. En el gráfico, entre más gruesas las líneas, significa que la conexión tiene un gran rendimiento en Bytes/s.

- **Descripción nodos /robot_manipulator_teleop y /serial_node:**

Figura 5: Grafo rqt ROS punto 1.

Se observa que el nodo robot_turtlebot_teleop se comunica con los servomotores del robot a través del tópico /robot_manipulator_angles por medio del cual comunica las velocidades lineales para el movimiento de cada servomotor en la comunicación Arduino-ROS creada que se expresa en el nodo */serial_node*, como se dijo anteriormente, se usan mensajes tipo Twist. Para visualizar el sistema se usó el gráfico rqt. Lo anterior permite tomar mejores decisiones para la integración de los demás puntos del taller pues de presentarse un error de comunicación entre los nodos, se podrá detectar fácilmente el problema.

- **Punto 2:**
  Para este punto se pedía crear un nodo llamado robot_manipulator_interface3d, en donde se pueda ver en tiempo real la posición del end-effector del manipulador. Esto debe poder verse dentro de una interfaz y poder guardar como una imagen el recorrido del robot. Se presenta en tiempo real en el marco global de referencia. Además, muestra el camino recorrido por el mismo desde donde inició. La interfaz cuenta con el espacio para asignarle un nombre a la gráfica y poderlo guardar en el directorio deseado, al finalizar el recorrido.

Para esto se debe hacer lo siguiente en una terminal nueva (cabe resaltar que la terminal debe estar en el ws): *rosrun rosserial_python*source devel/setup.bash y rosrun turtle_bot_3 robot_manipulator_odometry.py. En una terminal nueva (dentro del workspace) ejecutar source devel/setup.bash y rosrun turtle_bot_3 robot_manipulator_interface3d.py. Una vez ejecutados estos comandos se abrirá la interfaz que permite visualizar la posición en tiempo real del robot. Luego de esto, debería aparecer una interfaz con 3 botones: Guardar recorrido, Usar recorrido guardado y pagina de gráfica. Si se presiona el botón de "página de gráfica.ªparece la gráfica 3D que muestra la posición actual del robot. Esta gráfica tiene la opción de poder guardar la imagen del recorrido que se realizó desde el momento que se inició la interfaz hasta el momento en el que se guarda la imagen.

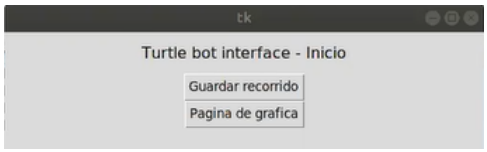

Figura 6: Visualización página 1 interfaz.

Al seleccionar página de gráfica, se visualizará el movimiento del robot en tiempo real (los comandos de movimiento los ejecuta el usuario a través de la terminal con las mismas teclas del punto anterior en 3D. Una vez se haya realizado el recorrido deseado, es posible guardarlo como imagen .PNG desde el menú inferior de la interfaz. Este permite cambiar el nombre de la imagen del recorrido y asignar una carpeta dentro de nuestro computador.

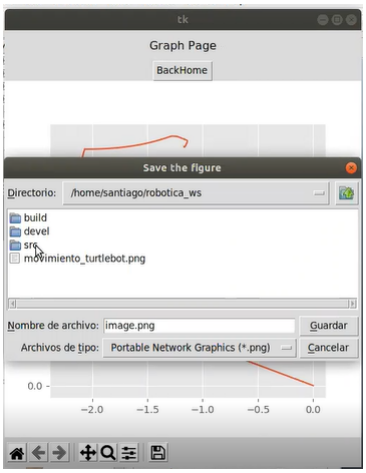

Figura 7: Guardar imagen del recorrido en el directorio deseado al finalizar la simulación.

- **Descripción de nodos:** se realizó un nodo para la interfaz (robot_manipulator_interface3d.py) que se suscribe al tópico /robot_manipulator_position. Se guarda la posición x, y

e z del robot en un vector y con la función *animation* de python se anima el movimiento. En la gráfica a continuación se evidencia la comunicación creada en ROS en donde el nodo /turtlebot_teleop (del punto 1, movimiento) se enlaza con el tópico /robot_manipulaor_angles para llegar al nodo de interfaz quien a su vez comunica la posición de turtlebot para publicarla en la interfaz en tiempo real. Este último tópico publica a los nodos /serial_node para la comunicación Arduino-ROS y al nodo /robot_manipulator_odom que describiremos más adelante.

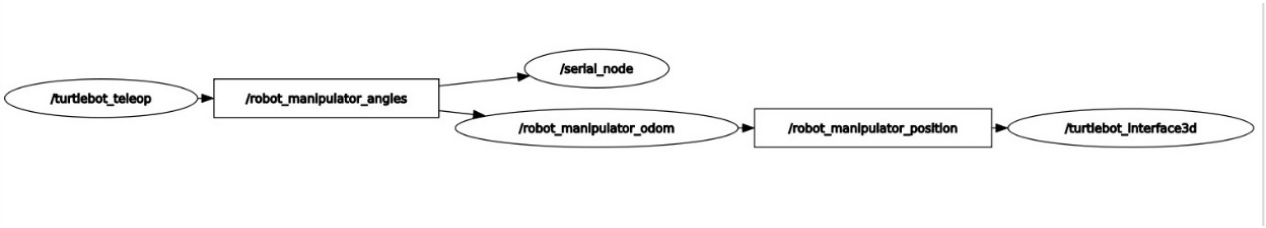

Figura 8: Grafo rqt ROS punto 2.

El nodo de odometría /robot_manipulator_odom se fundamenta en matrices que relacionan las distancias entre links y joints con los ángulos al moverse de tal manera que se pueda saber el recorrido de cada uno y del end-effector. Cuando recibe la información del tópico /serial_node, ejecuta los procedimientos de cálculo que se mencionaron previamente con la información que llega a través de la comunicación con Arduino el cual entrega las posiciones x, y, z e theta. El tópico /robot_manipulator_position expresa posición y orientación en un manifold 3D (float 64) a través de un mensaje tipo Pose2D para realizar el plot en la interfaz.

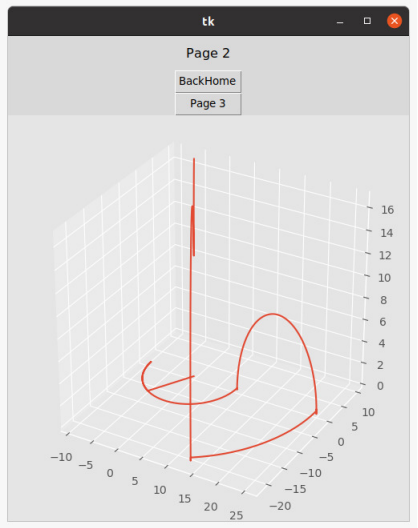

Figura 9: Gráfica 3D de la posición del end-effector y joints del manipulador en tiempo real.

- **Punto 3:**
  Para este punto se crea un nodo en ROS, llamado /robot_manipulator_planner, que permite a un usuario llevar el end-effector del robot a una posición deseada dentro de su volumen de trabajo. Este nodo incluye el cálculo de cinemática inversa o planeación de trayectorias para alcanzar esta posición.

  Volumen de trabajo: se definen los límites de cada uno de los grados de libertad como se muestra en la Figura 10. Cabe aclarar que el manipulador solo llega a puntos en la superficie de la esfera pues el joint de la mitad no se mueve.

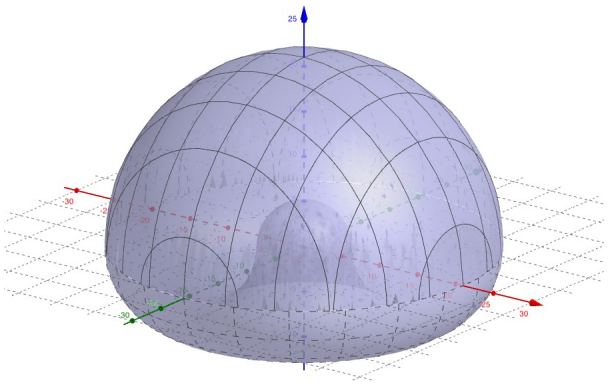

Figura 10: Gráfica 3D del volumen de trabajo del manipulador.

- **Descripción de los nodos:** la posición a alcanzar es publicada en el tópico /robot_manipulator_goal por terminal en tres posiciones x, y e z. Inicializa roscore y el movimiento del robot como en el punto 1. Por terminal, este proceso es el siguiente: se escribe en una terminal el comando *rosrun turtle_bot_3 robot_manipulator_planner.py* y *rostopic pub /robot_manipulator_goal geometry*$_m$*sgs/Point*"x : 0,0y : 0,0z : 0,0". $La Figura 11 muestra la comunicación del tópico de goal con el nodo /robot_manipulator$

Figura 11: RQT graph punto 3.

- **Punto 4:**

  El manipulador sobre el robot está en capacidad de tomar el ping pong del color especificado a través del tópico /robot_manipulator_ping_pong. Este color se especifica por terminal y contiene un rango RGB. Tras inicializar roscore y el movimiento del robot con la integración Arduino-ROS, en una nueva terminal se ejecuta dentro del workspace de robótica . *devel/setup.bash*. Para tomar el ping

pong del color especificado se colocó una cámara web en la parte frontal del manipulador de tal forma que al enfrentarse al panorama de la elección del ping pong por color, este identifique el color y gire la base del manipulador tal que quede centrada la imagen con el ping pong a tomar. Una vez realizado esto se realiza la planeación de trayectoria.

- **Descripción nodos:** como este punto utiliza bastantes recursos computacionales, se realiza una conexión master slave SSH para pasar el procesamiento de la Raspberry al computador. La cámara web es un nodo /usb_cam que se comunica a través del tópico /usb_cam (el cual contiene /usb_cam/Image_raw) con los tópicos de /robot_maipulator_camera y n_image_view los cuales se encargan de procesar la imagen recibida y publican en el tópico /robot_manipulator_goal la información de movimiento. Este tópico tiene comunicación con el nodo /robot_manipulator quien junto con /turtlebot_teleop se comunican con el tópico /robot_manipulator_angles para hacer el cálculo de cinemática inversa con el nodo de odomotería y movimiento con /serial_node. El tópico /robot_manipulator_position publica la posición del end-effector en la gráfica 3D (nodo) la cual ayuda a realizar la cinemática inversa.

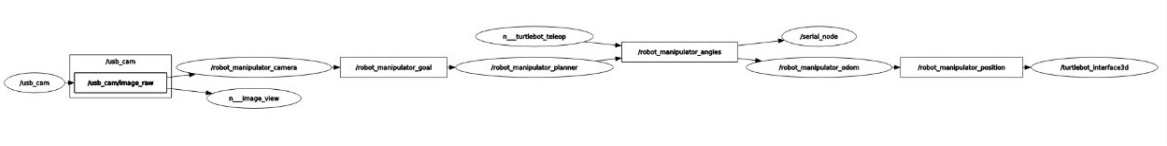

Figura 12: Grafo rqt ROS punto 4.

Referencias

[1] Chasis brazo robótico. $https://articulo.mercadolibre.com.co/MCO-514840737-brazo-robotico-arduino-4-servos-4-dof-grados-libertad-sg90-_JM?matt_tool=42035816matt_word=matt_source=googlematt_campaign_id=14634237770matt_ad_group_id=122266243170matt_match_type=matt_network=gmatt_device=cmatt_creative=545507349305matt_keyword=matt_ad_position=matt_ad_type=plamatt_merchant_id=219049012matt_product_id=MCO514840737matt_product_partition_id=1638503335577matt_target_id=aud-345731277262:pla-1638503335577gclid=CjwKCAjw3cSSBhBGEiwAVII0Z1OlQr2T4X52KJUhdqkEQlUvbkCBgS0P1uaUP64DuhvFf$

[2] Módulo Driver L298. https://tienda.tdrobotica.co/categoria/544-modulo-driver-l298.html

[3] Protoboard. https://tienda.tdrobotica.co/protoboards-cajas-soportes/429-protoboard-pequena.html