Robótica - Taller 2
Cinemática de Robots móviles

John Alejandro Duarte Carrasco - ja.duarte10@uniandes.edu.co - 2016140888
Jonathan Steven Roncancio Pinzon - js.roncancio@uniandes.edu.co - 201617312
Miguel Angel Mozo Reyes - ma.mozo@uniandes.edu.co - 201615973
Santiago Devia Valderrama - s.devia10@uniandes.edu.co - 201313766

12 de marzo de 2019

1. Punto 1

Enunciado: Un robot móvil se encuentra estático en el punto inicial $\varepsilon_{I0} = [1, -2, -\pi]^T$ y se dispone a realizar tres movimientos consecutivos de la siguiente manera: primero se empieza a mover con una velocidad constante de $\dot{\varepsilon}_R = [-0{,}5, 1{,}5, \pi/4]^T$ por 1.5 segundos, luego se mueve con una velocidad constante de $\dot{\varepsilon}_R = [0, 2, -\pi/4]^T$ por 2 segundos. Finalmente, se mueve con velocidad constante de $\dot{\varepsilon}_I = [2, 2, 0]^T$ durante 3 segundos. Determine la posición y dirección final del robot en el marco global de referencia pasados los 6.5 segundos. Muestre el procedimiento (ecuaciones y resultados parciales) paso a paso.

Solución:

En el tiempo $t_0 = 0$ se tiene:

$$\varepsilon_{I0} = [1, -2, -\pi]^T \qquad \dot{\varepsilon}_R = [-0{,}5, 1{,}5, \pi/4]^T$$

1. Para el tiempo $t_1 = 1{,}5s$

$$\varepsilon_{It_1} = \varepsilon_{It_0} + \dot{\varepsilon_I} t = \varepsilon_{It_0} + R(\theta)^{-1} \dot{\varepsilon_R} t$$

$$\varepsilon_{It_1} = \begin{bmatrix} 1 \\ -2 \\ -\pi \end{bmatrix} + \begin{bmatrix} cos(-\pi) & sin(-\pi) & 0 \\ -sin(-\pi) & cos(-\pi) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -0{,}5 \\ 1{,}5 \\ \pi/4 \end{bmatrix} t_1 = \begin{bmatrix} 1 \\ -2 \\ -\pi \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -0{,}5 \\ 1{,}5 \\ \pi/4 \end{bmatrix} [1{,}5]$$

$$\varepsilon_{It_1} = \begin{bmatrix} 1{,}75 \\ -4{,}25 \\ -5\pi/8 \end{bmatrix}$$

2. Para el tiempo $t_2 = t_1 + 2s$

$$\varepsilon_{It_2} = \varepsilon_{It_1} + \dot{\varepsilon_I} t = \varepsilon_{It_1} + R(\theta)^{-1} \dot{\varepsilon_R} t$$

$$\varepsilon_{It_2} = \begin{bmatrix} 1{,}75 \\ -4{,}25 \\ -5\pi/8 \end{bmatrix} + \begin{bmatrix} cos(-5\pi/8) & sin(-5\pi/8) & 0 \\ -sin(-5\pi/8) & cos(-5\pi/8) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2 \\ -\pi/4 \end{bmatrix} (t_2 - t_1)$$

$$\begin{bmatrix} 1{,}75 \\ -4{,}25 \\ -5\pi/8 \end{bmatrix} + \begin{bmatrix} -\frac{\sqrt{2-\sqrt{2}}}{2} & -\frac{\sqrt{2+\sqrt{2}}}{2} & 0 \\ \frac{\sqrt{2+\sqrt{2}}}{2} & -\frac{\sqrt{2-\sqrt{2}}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2 \\ -\pi/4 \end{bmatrix} [2]$$

$$\varepsilon_{It_2} = \begin{bmatrix} 5{,}4455 \\ -5{,}7807 \\ -\pi/8 \end{bmatrix}$$

3. Por último, para el tiempo $t_3 = t_2 + 3s$

$$\varepsilon_{It_3} = \varepsilon_{It_2} + \dot{\varepsilon_I} t$$

$$\varepsilon_{It_3} = \begin{bmatrix} 5{,}4455 \\ -5{,}7807 \\ -\pi/8 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} (t_3 - t_2) = \begin{bmatrix} 5{,}4455 \\ -5{,}7807 \\ -\pi/8 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} [3]$$

Así pues, la posición y dirección del robot pasados los 6.5 segundos es:

$$\varepsilon_{It_2} = \begin{bmatrix} 11{,}4455 \\ 0{,}2192 \\ -\pi/8 \end{bmatrix} \rightarrow x = 11{,}44m \quad y = 0{,}22m \quad \theta = -\pi/8 = -22{,}5^\circ$$

2. Punto 2

Enunciado: Cree un nodo en ROS que permita a un usuario controlar el robot *Pioneer_p3dx* en V-Rep. La interfaz de control (teclado, control, hardware, etc) es de libre decisión de cada grupo y debe ser descrita en el documento de entrega. Se espera que el control sea lo más natural posible (i.e., que sea fácil e intuitivo de controlar). Además, el usuario debe poder controlar la velocidad general con la que se mueve el robot. El ingreso de esta velocidad se realizará como argumento cuando se lanza el nodo; en ningún caso se admitirá tener que modificar el código para actualizar la velocidad. A medida que el robot se mueve, el programa deberá mostrar una pantalla donde se represente su posición en el plano global de referencia en tiempo real y se aprecie el camino recorrido por el mismo desde que se inició la simulación. Esta gráfica debe ser almacenada en un archivo .png en una carpeta **results** dentro del paquete creado al finalizar la simulación de V-Rep. Realice y entregue un vídeo mostrando cómo se hace el control del robot. **Bono**: *Se agregará una nota adicional de 0.5 al punto si se realiza un desarrollo adicional en el que se despliegue una ventana de interfaz gráfica que permita actualizar el valor de la velocidad del robot durante le ejecución del nodo.*

Solución:

Para generar el movimiento del robot de acuerdo a la dirección deseada, se modifican las velocidades de movimiento de cada uno de los motores del robot. Para este, caso el robot *Pioneer_p3dx* posee 2 ruedas con 2 motores principales con los siguientes nombres:

- "Pioneer_p3dx_leftMotor"
- "Pioneer_p3dx_rightMotor"

Para modificar cada una de las velocidades de los motores,

El control del robot se hará a través del teclado del usuario y la velocidad del mismo se escogerá a través de un texto de entrada *input box* para esto se hace uso del paquete de python *pynput* el cual permite reconocer la tecla presionada por el usuario, y el paquete de *Tkinter* que permite crear interfaces de comunicación con el usuario. Para realizar el control del robot se hace uso de las teclas arriba y abajo únicamente [Key.up(↑) o Key.down (↓)]. Este control se realiza a partir de las siguientes funciones principales

- **def callbackPioneerPosition(msg):** Obtiene la posición actual del robot en términos de de x,y y θ (posición angular), así mismo, guarda estas posiciones en un vector.
- **def ventanaDialogo():** Genera la ventana de dialogo a través de la cual el usuario indicará la velocidad que desea darle al robot.
- **def main():** Realiza la inicialización de las variables globales y los nodos de ros, permite la subscripción y la publicación de mensajes para la intercomunicación entre v-rep y ros. Así mismo, genera la gráfica de la posición del robot y al finalizar, la imagen del recorrido realizado.

La solución generada contempla el siguiente funcionamiento del robot y su respectivo control. Al digitar una velocidad sobre el robot, los motores de cada rueda empezarán a moverse a una velocidad constante igual a la especificada por el usuario, una vez que el robot esta en movimiento, el usuario puede hacer uso de las teclas ↓ y ↑ para controlar al robot. Si la tecla presionada es ↓, las velocidades del motor se modificarán para hacer que el motor de la rueda derecha ahora tenga una velocidad 2 veces superior a la velocidad que poseía anteriormente, mientras que la velocidad del motor de la rueda izquierda se mantendrá constante en el valor asignado por el usuario. En caso de que la tecla presionada sea ↑, las velocidades de los motores se modificarán para que ocurra lo inverso a lo mencionado anteriormente. De este modo se logrará hacer que el robot gire en una dirección o la otra de acuerdo a lo deseado por el usuario. Si el usuario desea que el robot se detenga, tendrá que escribir en la caja de dialogo *'text box'* el valor de 0.

En el siguiente diagrama se puede ver el funcionamiento general de la solución propuesta:

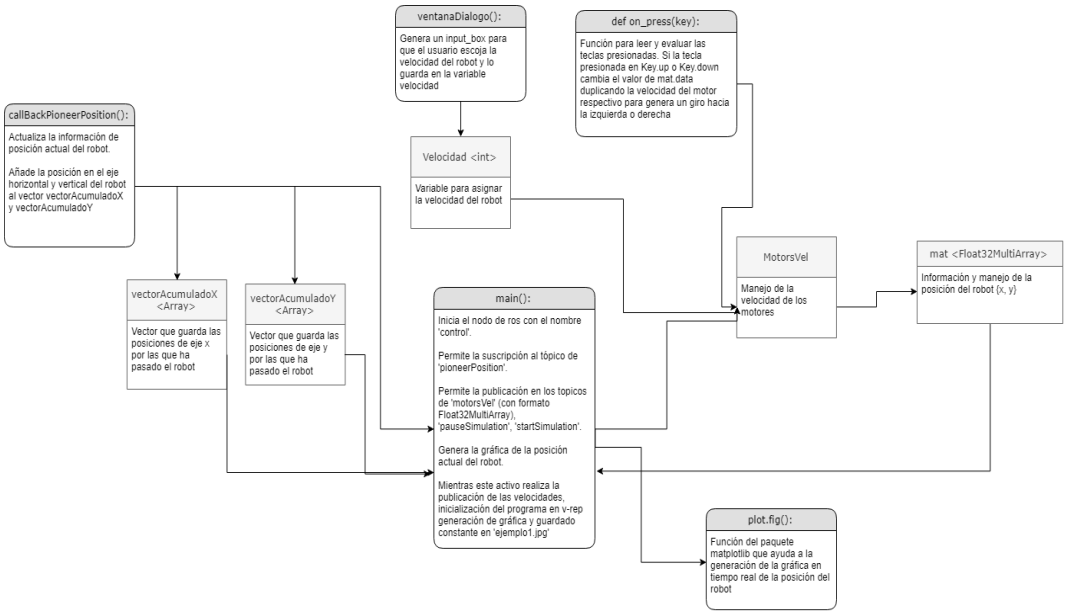

Figura 1: Diagrama funcionamiento control de robot

Al ejecutar la solución, se puede ver el funcionamiento y la interconexión entre los nodos de ros que publican y se suscriben para controlar al robot; en la figura 2 se puede visualizar esta interconexión.

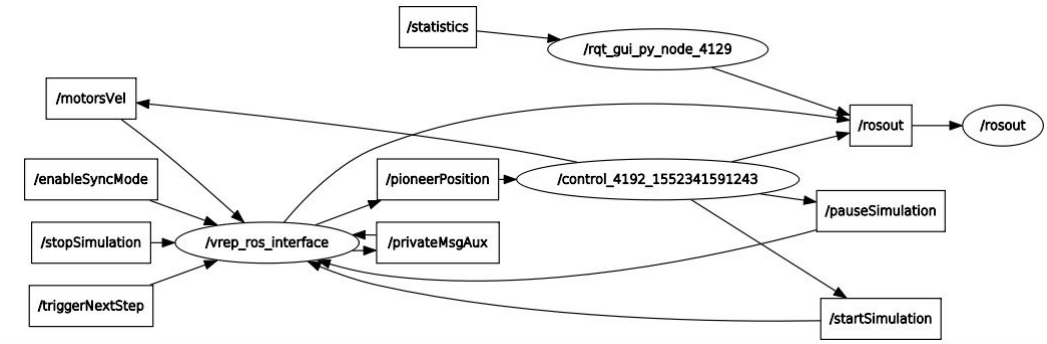

Figura 2: Diagrama de nodos y tópicos ROS punto 2

Durante la ejecución del programa se puede ver la gráfica de la posición actual del robot, al mismo tiempo que se mantiene abierta la ventana de texto para modificar la velocidad del mismo en cualquier momento que se desee; en el archivo de vídeo adjunto se puede ver el funcionamiento completo del mismo. En la siguiente imagen se muestra cómo se genera la gráfica durante la ejecución del código, y el cuadro de texto de velocidad.

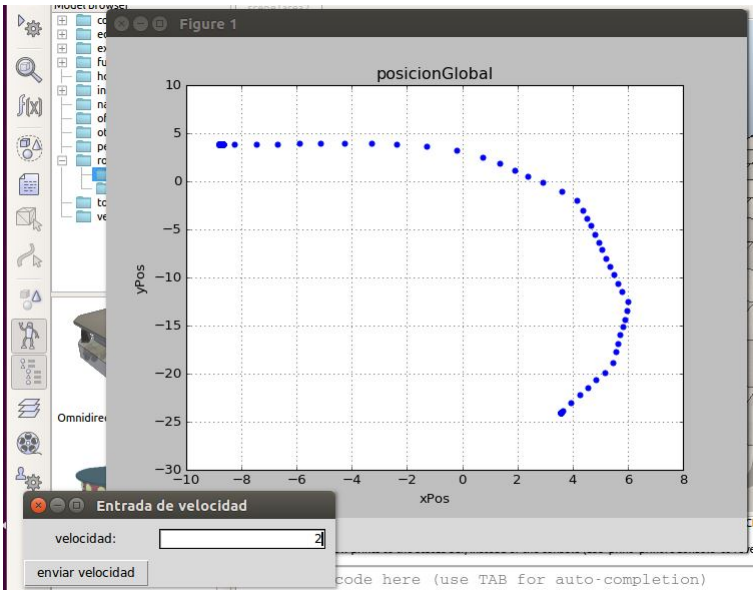

Figura 3: Funcionamiento punto 2

Una vez finalizada la ejecución del programa, se guarda la última gráfica proyectada en un archivo de nombre 'ejemplo1.JPG' el cual se ubicará en la misma carpeta desde donde se lanzo el código en python. La imagen creada en formato JPG se puede ser en la siguiente figura [1]:

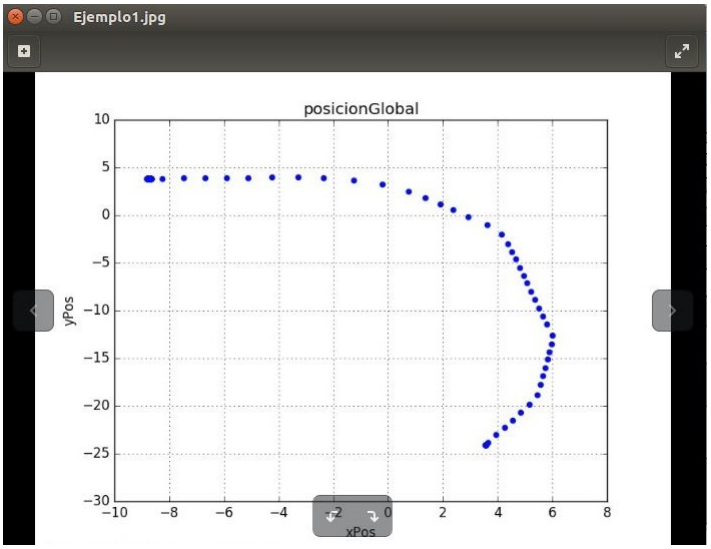

Figura 4: Imagen guardada en formato JPG

3. Punto 3

Enunciado: Cree un nodo en ROS que lea de un archivo de texto un perfil de velocidad para el robot *Pioneer_p3dx* de V-Rep y simule su movimiento en el marco de referencia global. El nombre del archivo se debe pasar como argumento cuando se lanza el nodo. Su programa debe generar una gráfica en el marco global de referencia que vaya mostrando la posición actual del robot (exacta) en tiempo real y el camino recorrido. En la misma gráfica se debe mostrar la posición y camino recorrido del robot simulado en V-Rep. Al mismo tiempo, el programa debe controlar el robot en V-Rep de manera que éste se mueva de acuerdo a los perfiles de velocidad dados. Al finalizar el movimiento del robot, se debe desplegar una ventana en la que se observe el error de posición del robot en V-Rep en el eje vertical y el tiempo de simulación en el eje horizontal. El cálculo de la cinemática del robot debe ser implementado por cada grupo; no se admite usar programas realizados por otros. Tanto la gráfica generada del error como la de la trayectoria realizada por el robot deben ser almacenadas en archivos .png en una carpeta **results** dentro del paquete al finalizar

[1] La figura muestra una captura del archivo en vez de mostrar directamente el archivo, esto se hace intencionalmente para mostrar el nombre con el que fue guardado y el formato del mismo.

la simulación. El archivo de entrada se debe almacenar en una carpeta **resources** dentro del paquete y su estructura será de la siguiente manera: La primera línea tendrá un número entero que corresponderá al número de velocidades que deberán ser simuladas por el robot. La línea 2 del archivo tendrá 3 números decimales separados por un espacio que corresponderán a la velocidad del motor derecho, la velocidad del motor izquierdo y el tiempo que se mantendrán dichas velocidades respectivamente. El tiempo de simulación específico debe ser leído de V-Rep e integrado en el modelo cinemático. En el informe escrito incluya las ecuaciones de la cinemática usadas en el programa junto con el valor de los parámetros. Realice y entregue un vídeo mostrando cómo se mueve el robot en la simulación.

Solución:

En primer lugar se crea el nodo $P3_simuladorPosicionPiooner$, se suscribe el nodo a los tópicos *pioneerPosition* y *simulationTime* con el fin de conocer la posición/orientación del robot y el tiempo de simulación publicado por vrep . Finalmente se crea una relación entre el nodo inicializado y el tópico *motorsVel* donde el primero publica la velocidad de los motores en el segundo. Al ser lanzado, el nodo toma el archivo pasado por parametro y lo descompone en tres vectores: el vector velocidadesX, el vector velocidadesY y el vector tiempoVelocidades que corresponden a las velocidades para el motor derecho, las velocidades para el motor izquierdo y los tiempos que deben durar cada par de velocidades respectivamente.

Utilizando dicha información, se utiliza la función *determinarVelocidadActual* para saber que par de velocidades debe tener el robot en ese preciso instante. Para esto, la función utiliza la variable *deltaTiempoVelocidades* que me dice cuanto tiempo llevan activas las velocidades actuales, y en caso de que este valor sea mayor o igual al tiempo que deben estar esas velocidades el sistema cambia automáticamente al siguiente par de velocidades registrado en el archivo. Siguiendo el mismo proceso para cada arreglo de velocidades registrado en el archivo. En caso de no tener mas velocidades para leer, el sistema pone las velocidades de los motores en 0.

Ya con el nodo publicando las velocidades, se hace uso de la función *callbackPioneerPosition* para actualizar la posición del robot simulado y agregarla a los vectores *posicionXActualSimulada y posicionYActualSimulada.* Por otro lado, en esta misma función se llama al método *calcularRecorridoExacto* que utiliza cinemática directa para calcular con las velocidades actuales de los motores a que punto *x,y,*θ debería llegar el robot según la teoría. El planteamiento de la cinemática inversa del pioneer se presentan a continuación:

El planteamiento de la cinemática inversa comienza por encontrar los parámetros α, β para cada rueda, dichos parámetros corresponden a $\alpha_1 = -\frac{\pi}{2}, \beta_1 = \pi$ para la rueda1(rueda derecha) y $\alpha_2 = \frac{\pi}{2}, \beta_2 = 0$ para la rueda2(rueda izquierda). A continuación, se halla la variable l que esta asociada a la distancia desde el punto P definido en el chasis del robot y el punto de contacto con el suelo de las llantas. Dado que el punto P se ubicó en la parte trasera del chasis del robot, sobre el eje de giro de las llantas y equidistante de cada una se halla que $l = AnchoRobot/2 - AnchoLlantas/2$,cabe aclarar que en este calculo se asume que *AnchoRobot* es la distancia desde una llanta hasta la otra en el robot.

$$l = \frac{AnchoRobot}{2} - \frac{AnchoLlantas}{2} = \frac{39{,}3cm}{2} - \frac{4,75cm}{2} = 17{,}275cm$$

Las características físicas del robot PioneerP3DX utilizadas para los cálculos fueron halladas en el manual de usuario del robot encontrado en la pagina web [2].

Luego, se plantea la matriz de restricciones de la siguiente forma:

$$J_1 = \begin{bmatrix} \sin(\alpha_1+\beta_1) & -\cos(\alpha_1+\beta_1) & -l\cos(\beta_1) \\ \sin(\alpha_2+\beta_2) & -\cos(\alpha_2+\beta_2) & -l\cos(\beta_2) \\ \cos(\alpha_1+\beta_1) & \sin(\alpha_1+\beta_1) & l\sin(\beta_1) \\ \cos(\alpha_2+\beta_2) & \sin(\alpha_2+\beta_2) & l\sin(\beta_2) \end{bmatrix}$$

Al remplazar se obtiene que:

$$J_1 = \begin{bmatrix} \sin(-\frac{\pi}{2}+\pi) & -\cos(-\frac{\pi}{2}+\pi) & -l\cos(\pi) \\ \sin(\frac{\pi}{2}) & -\cos(\frac{\pi}{2}) & -l\cos(0) \\ \cos(-\frac{\pi}{2}+\pi) & \sin(-\frac{\pi}{2}+\pi) & l\sin(\pi) \\ \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & l\sin(0) \end{bmatrix} = \begin{bmatrix} \sin(-\frac{\pi}{2}+\pi) & -\cos(-\frac{\pi}{2}+\pi) & -l\cos(\pi) \\ \sin(\frac{\pi}{2}) & -\cos(\frac{\pi}{2}) & -l\cos(0) \\ \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & l\sin(\pi) \\ \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & l\sin(0) \end{bmatrix}$$

Dado que $\sin(\pi) = \sin(0)$ se llega finalmente a la siguiente ecuación:

[2] $https://www.inf.ufrgs.br/\ prestes/Courses/Robotics/manual_pioneer.pdf$

$$J_1 = \begin{bmatrix} \sin\left(-\frac{\pi}{2}+\pi\right) & -\cos\left(-\frac{\pi}{2}+\pi\right) & -l\cos\left(\pi\right) \\ \sin\left(\frac{\pi}{2}\right) & -\cos\left(\frac{\pi}{2}\right) & -l\cos\left(0\right) \\ \cos\left(\frac{\pi}{2}\right) & \sin\left(\frac{\pi}{2}\right) & l\sin\left(\pi\right) \end{bmatrix}$$

Así mismo, se utiliza la matriz de rotación $R(\theta)$ y el vector $r\dot{\phi}$.

$$R(\theta) = \begin{bmatrix} \cos\left(\theta\right) & \sin\left(\theta\right) & 0 \\ -\sin\left(\theta\right) & \cos\left(\theta\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad r\dot{\phi} = \begin{bmatrix} r\dot{\phi}_1 \\ r\dot{\phi}_2 \\ 0 \end{bmatrix}$$

Donde r corresponde al radio de las ruedas del robot y $\dot{\phi}_1, \dot{\phi}_2$ son las velocidades en la rueda derecha e izquierda respectivamente. Como penúltimo paso, se plantea la ecuación del vector de velocidades globales $\dot{\xi}_I$:

$$\dot{\xi}_I = R(\theta)^{-1} J_1^{-1} r\dot{\phi}$$

Por ultimo, se realiza el calculo de la posición y orientación del robot. Para esto, se utiliza la variable *pasoDeSimulacion* que define el tiempo que duran las velocidades globales calculadas en el robot, o visto de otra forma, cada cuanto tiempo muestreo el sistema físico.

$$PosicionActualTeorica = PosiciónTeoricaAnterior + \dot{\xi}_I * pasoDeSimulacion$$

$$PosicionActualXTeorica = PosicionXAnteriorTeorica + \dot{x} * pasoDeSimulacion$$
$$PosicionActualYTeorica = PosicionYAnteriorTeorica + \dot{y} * pasoDeSimulacion$$
$$\theta_{Actual} = \theta_{Anterior} + \dot{\theta} * pasoDeSimulacion$$

Las nuevas posiciones *x,y* teóricas son almacenadas en los vectores posicionXActualExacta y posicionYActualExacta. En cuanto a θ se hicieron dos análisis con el fin de ver las diferencias entre los movimientos del robot simulado y teórico.

Para los dos casos se utilizo un archivo de coordenadas con los valores consignados en la tabla 1

Motor Derecho[rad/sg]	Motor Izquierdo[rad/sg]	Tiempo[sg]
7	0	8
2	1	10
6	2	10
-4	-2	10

Cuadro 1: Tabla de velocidades

Para el primer caso, se tomo el θ arrojado por vrep en el tópico pioneerPosition y se obtuvieron los siguientes resultados:

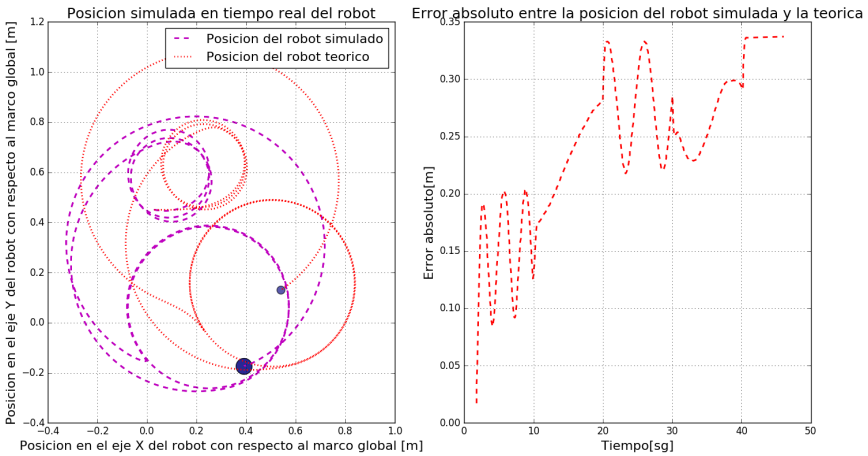

Figura 5: Resultados obtenidos con θ del simulador

Para el segundo caso, se tomo θ como el valor calculado después de hacer cinemática directa sobre el robot teórico y se obtuvieron los resultados mostrados en la gráfica 6.

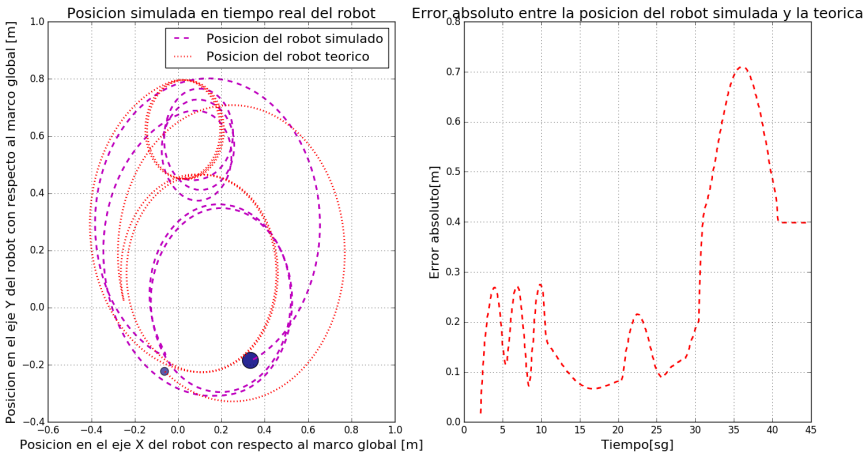

Figura 6: Resultados obtenidos con θ teórico

Como se puede observar en la figura 6 el error al tomar θ teórico es similar en los primeros instantes, pero una vez se llega a 30sg el error tiene un pico de crecimiento que alcanza su máximo, aproximadamente, en el segundo 37 de ejecución. Según el análisis realizado, esta diferencia en los errores se debe a que la función *calcularRecorridoExacto*, que es la encargada de calcular θ teórico, es ejecutada cada vez que el tópico *pioneerPosition* es actualizado. Debido a esto, existe una diferencia de tiempos entre el θ simulado y el θ teórico que gracias a la propagación del error crece a medida que avanza el tiempo, se pudo evidenciar que para tiempos grandes(mayores a 15sg) la diferencia entre los dos ángulos podía ser de hasta 1 radian o 1.5 radianes, es decir que mientras θ simulado estaba en 2 radianes su análogo teórico se encontraba en 1 radian o incluso 0.5 radianes.

Para finalizar, se presentan el diagrama de bloques del código y el grafo de conexiones en vrep.

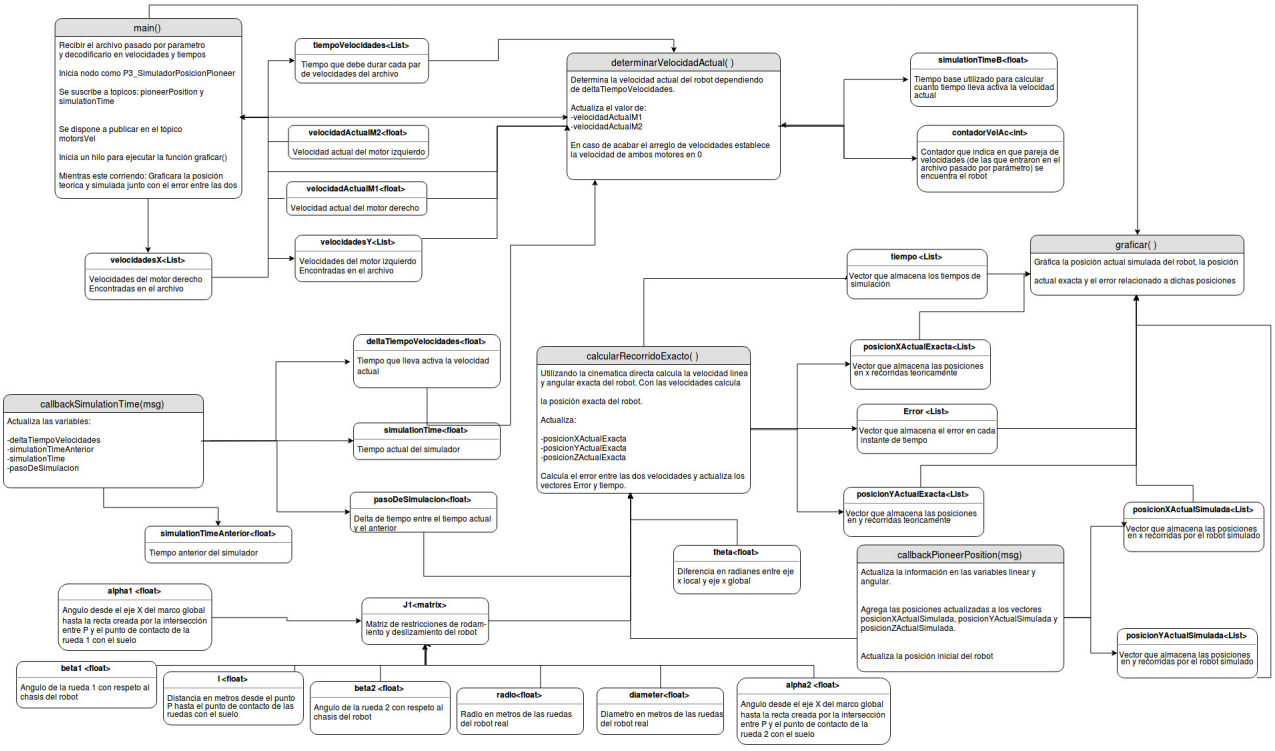

Figura 7: Diagrama de bloques para el punto 3

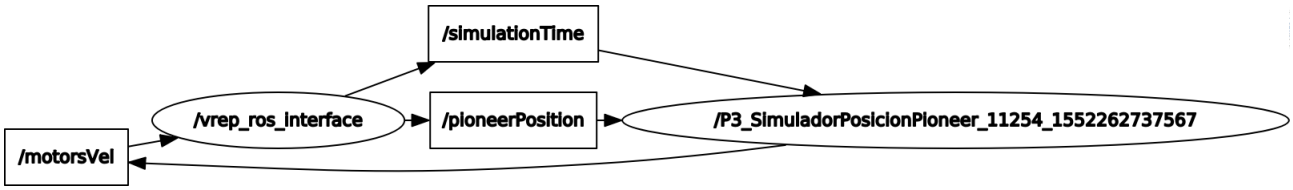

Figura 8: Grafo de conexines para el punto 3

4. Punto 4

Cree un nodo en ROS que reciba como parámetro de entrada (al lanzar el nodo) 3 números que corresponderán a la posición final del robot $\varepsilon_{If} = [x_f, y_f, \theta_f]^T$ e implemente la cinemática y ley de control necesarios para llevar al robot desde la posición inicial hasta la posición final. Asegúrese que el robot en V-Rep inicia la simulación estando en la posición inicial $\varepsilon_{I0} = [0, 0, -\pi]^T$. Programe su nodo para que si el usuario no ingresa parámetros, se asuma que la posición final es $\varepsilon_{If} = [40, 40, \pi/2]^T$. Al ejecutar el nodo se deberá mostrar en tiempo real una gráfica en el marco global de referencia con la posición actual del robot (exacta) y la posición del robot en el simulador en tiempo real y el camino recorrido en los dos casos. Al finalizar el movimiento del robot, se debe desplegar una ventana en la que se observe el error de posición del robot en V-Rep en el eje vertical y el tiempo de simulación en el eje horizontal. El cálculo de la cinemática del robot y de la ley de control deben ser implementados por cada grupo; no se admite usar programas realizados por otros. Tanto la gráfica generada del error como la de la trayectoria realizada por el robot deben ser almacenadas en archivos .png en una carpeta **results** dentro del paquete al finalizar la simulación. El tiempo de simulación debe ser leído de V-Rep. En el informe escrito incluya las ecuaciones de la cinemática y control usadas en el programa junto con el valor de los parámetros. Especifique cómo se escogieron los valores de k_ρ, k_α y k_β. Realice y entregue un vídeo mostrando cómo se mueve el robot en la simulación.

Solución:

Para la implementación de este código partió del hecho de la ley de control proporcional aplicada para este sistema (ver figura 9).

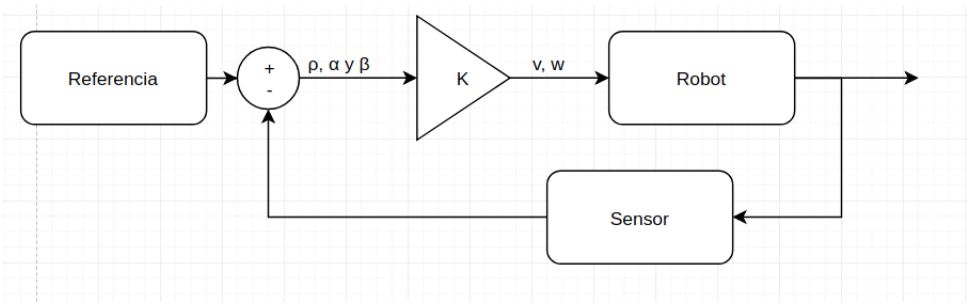

Figura 9: Diagrama de control implementado en el robot.

En este, se tiene primero una referencia: el valor al que se desea llegar. En este caso éste era de $[0, 0, \theta_f]$, siendo θ_f el ángulo en el que debe quedar apuntado el robot. Segundo se tiene el *Sensor*, que es básicamente el vector de posición actual del robot en dichas coordenadas ρ, α y β. A partir de estos, se obtenía el error como:

$$e(t) = [-\rho_R, -\alpha_R, \theta_B - \beta_R]$$

Este error pasa a través de un multiplicador proporcional k, que tiene un valor de:

$$k = [k_\rho = 0{,}02, k_\alpha = 0{,}4, k_\beta = 0{,}01]$$

A partir del producto de estos arreglos, se puede obtener $v = k_\rho \rho$ y $\omega = k_\alpha \alpha + k_\beta \beta$, que entran al modulo del Robot. Este genera a partir de la siguiente ecuación los valores de velocidad de los motores:

$$\Phi_i = \frac{1}{r}[sin(\alpha_{ri} + \beta_{ri}) - cos(\alpha_{ri} + \beta_{ri}) - lcos(\beta_{ri})]R(\theta)\dot{\varepsilon_I}$$

siendo α_{ri}, β_{ri}, r y l los parámetros de las llantas, R(θ) la matriz de transformación del plano global al local, que depende del θ actual del robot, y $\dot{\varepsilon_I}$, que se define como:

$$\dot{\varepsilon_I} = [v * cos(\theta), v * sin(\theta), \omega]$$

Resultando al final:

$$\Phi_i = \frac{1}{r}[sin(\alpha_{ri} + \beta_{ri}) - cos(\alpha_{ri} + \beta_{ri}) - lcos(\beta_{ri})] \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (k_\rho \rho) * cos(\theta) \\ (k_\rho \rho) * sin(\theta) \\ k_\alpha \alpha + k_\beta \beta \end{bmatrix}$$

Para determinar los valores de $k_\rho, k_\alpha, k_\beta$ se partió por determinar primero el valor de k_α. Disponiendo los otros de k en 0, se vario este en diferentes rangos hasta encontrar aquel valor que permitía variar la rotación del robot de manera que quedara apuntando a la dirección del punto final. Así mismo, se variaba de manera que el *overshoot* no fuera muy alto, pero que llegara al ángulo indicado en un tiempo considerablemente rápido. Luego de haber determinado el k_α, inicialmente en 0.5, se empezó a variar el valor de k_ρ. Este valor era subido poco a poco, partiendo en 0.001, de manera que el robot empezara a dirigirse al punto de destino. Aquí, se incrementada éste hasta un punto donde el robot no empezara a descontrolarse pero que fuera lo más rápido posible. Este pues se termino determinando en 0.02. Por último, se varió k_β, de manera que cuando el robot llegara al punto final empezara a girar a la posición indicada. Este valor tenia que disponerse en un valor pequeño pues tendía a influir mucho sobre ω y terminaba generando que el robot fuera a otra dirección. En este punto, se redujo un poco el valor de k_α a 0.04, de manera que se pudiera aumentar un poco el de k_β, hasta un valor final de 0.01.

La implementación del código se puede evidenciar en la figura 10.

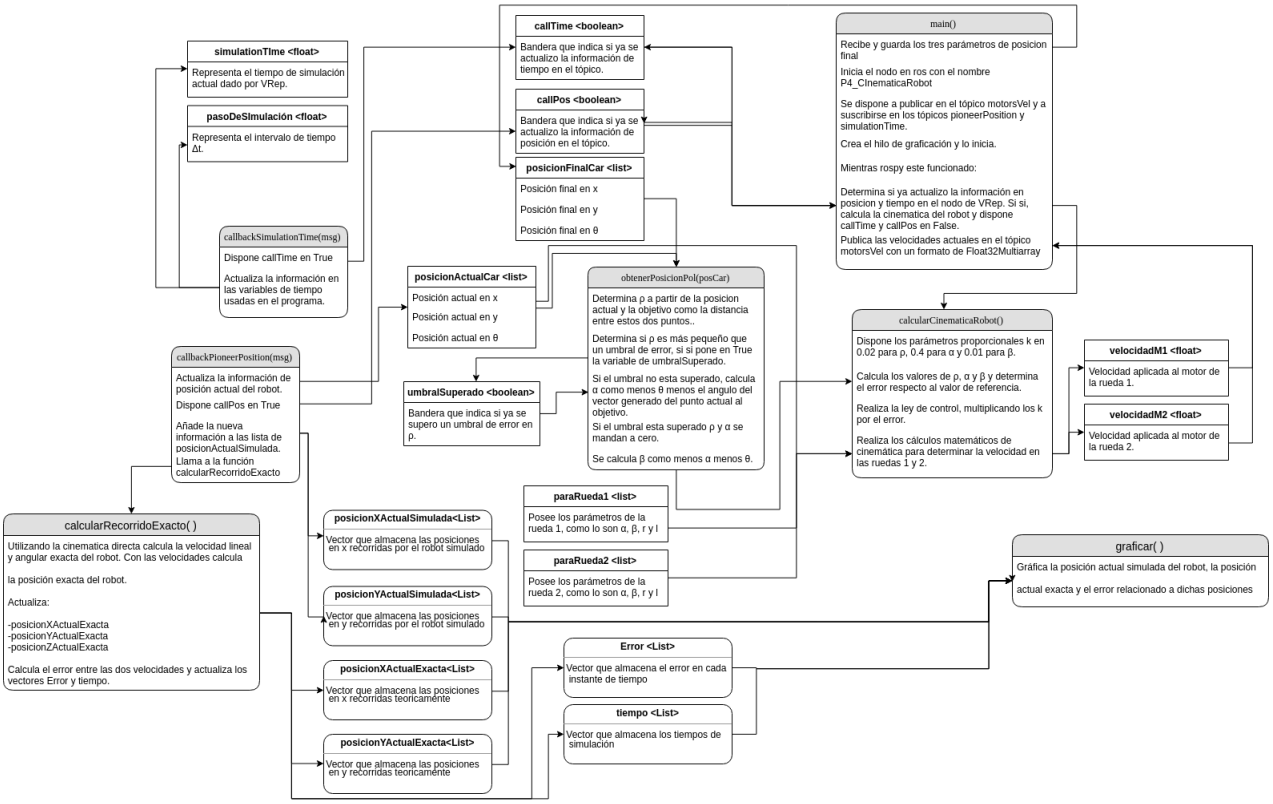

Figura 10: Diagrama de bloques del punto 4.

El código inicia pidiendo los valores de posición final (si no se envían se tiene unos por defecto). Luego de esto inicia el nodo ante ROS, y se dispone a publicar en el tópico *motosVel* y suscribirse a los tópicos *simulationTime* y *pioonerPosition*. Al suscribirse se asignan funciones callback, que son llamadas una vez se actualiza la información de los tópicos.

Luego, la función main() creaba el hilo de graficación, que tiene el mismo funcionamiento que el descrito en el punto 3.

En este punto, se fijaba la tasa de actualización del nodo en 20Hz, y se creaba un ciclo while que mantiene al programa en ejecución. Durante cada ciclo se determina si la información en los tópicos ya fue actualizada, haciendo uso de las variables *callPos* y *callTime*. Si efectivamente la información habia sido actualizada, se llama a la función calcularCinematicaRobot(), que realiza el sistema de control descrito anteriormente para determinar las velocidades de los motores. Luego de esto publica dicha información en el tópico *motorsVel*.

Respecto al funcionamiento del método calcularCinematicaRobot(), este primero determina la posición en términos de las variables ρ, α y β haciendo uso de la función obtenerPosicionPol(). Con esta, se obtiene el error y se

multiplica este por los valores K. A partir de esto se determina la ley de control con las ecuaciones anteriormente mencionadas.

Es importante mencionar que durante el proceso de determinación de la posición en coordenadas polares, se tiene una condición que verifica si el valor de ρ ya ha superado cierto umbral. Este umbral es de 0.05, y una vez ρ es más pequeño que este valor, significando que ya se esta prácticamente en el punto final deseado, se impone que $\alpha = 0$ y $\rho = 0$. Esto se hace puesto que computacionalmente nunca se va a obtener un cero, y esto provocara que el valor de α tienda a variar muy abruptamente, generado que el sistema de control genere errores. Con estos valores fijos en cero, el valor de β, a partir de la ley de control, empieza a aproximarse al valor de θ deseado.

Así mismo, el hilo de graficación despliega una interfaz con dos gráficas. La primera contiene dos curvas, donde la primera (roja) representa la curva teórica de recorrido por el robot, y la segunda (morada punteada) representa la curva recorrida en simulación. La segunda gráfica corresponde al error entre la teórica y la simulada a lo largo del tiempo. Al final, se termina generando una imagen con la captura final de estas gráficas en la carpeta *results*. Una captura de esta gráfica se puede evidenciar en la figura 11.

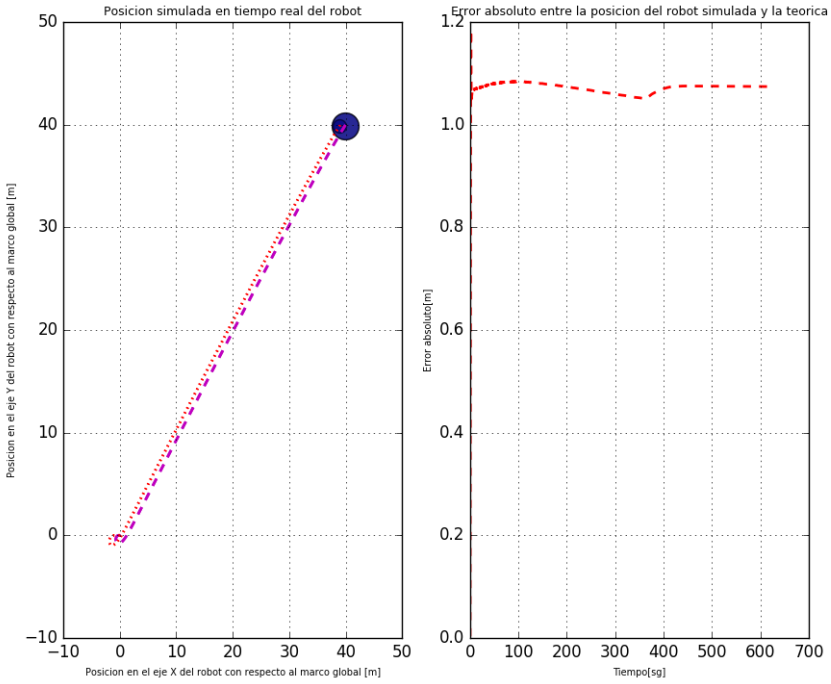

Figura 11: Gráfica generada por el código implementado en el punto 4, para la implementación de una ley de control en un robot Pioneer. En esta, la posición final que se fijo fue de [40,40,π/2]

En esta, se puede observar una ligera diferencia entre el teórico y simulado, esto se puede deber al delta de tiempo que se produce entre el calculo de la teórica y la simulada, lo que genera un pequeño desplazamiento en la gráfica al comienzo de esta.

A continuación se puede evidenciar el diagrama de nodos de ROS. En este punto, se recomienda iniciar primero el código, y después la simulación, de modo que el sistema este preparada para recibir información.

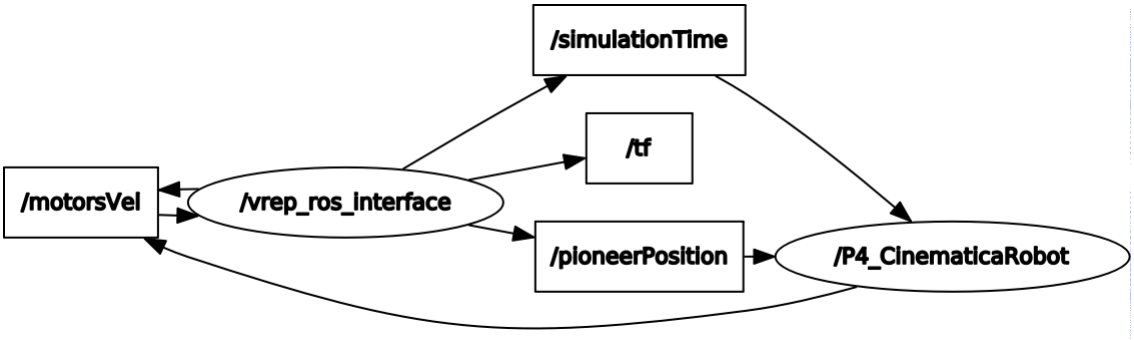

Figura 12: Diagrama de nodos de ROS del punto 4.

En el siguiente link `https://youtu.be/Y-RfUdAyKVA` se puede ver un vídeo donde se aprecia el funcionamiento de este código.