

Robótica – Trabajo Práctico N° 1:

Cinemática de un robot móvil diferencial

Integrantes:

- Federico Coppede
- Matías Paramidani

Consignas:

El desarrollo del TP1 consistirá en la modelación cinemática del robot educativo N6 de la empresa robogroup, el cual será el hardware disponible para probar el modelo obtenido durante la realización de la práctica.

- Calcular en Matlab la trayectoria.
- Programar el movimiento de Cinemática directa del Robot diferencial.
- Realizar un plot del encoder o trayectoria medida.
- Comparar El modelo Matlab con la programación real y extraer conclusionesy gráficos.

Hipótesis básicas simplificadoras:

- El robot se mueve sobre una superficie plana sin irregularidades ni defectos del suelo.
- Los ejes de guiado son perpendiculares al suelo.
- Se supone que las ruedas se mueven con rodadura pura (el deslizamiento es despreciable)
- Robot sin partes flexibles.
- (Durante un período de tiempo suficientemente pequeño en el que se mantiene constante la consigna de dirección, el vehículo se moverá de un punto al siguiente a lo largo de un arco de circunferencia.) [Olleros]

Introducción

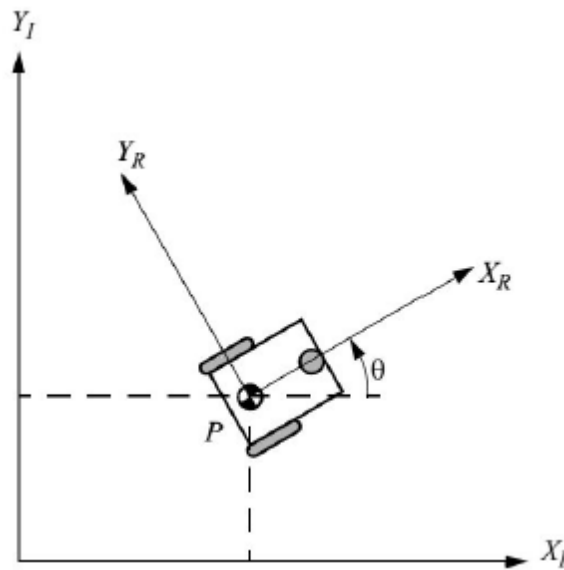
El robot móvil en cuestión tiene 3 grados de libertad respecto a una referencia: posición en el plano (X,Y) y orientación (Θ). Idealmente, independientemente de donde inicie, el robot debe poder moverse a cualquier posición y orientación (X,Y, Θ).

La manera de encontrar el modelo cinemático del movimiento del robot se logra con un proceso bottom-up, es decir, cada rueda individual contribuye al movimiento total del robot, y al mismo tiempo impone restricciones al movimiento del mismo.

Por lo tanto se modelará al robot como un cuerpo rígido sobre ruedas, circulando sobre un plano horizontal.

Para establecer la posición del robot en el plano, se establece una relación entre el sistema global de referencias (llamado también inercial), y el sistema local de referencia del Robot.

Por lo tanto posicionar el punto P en el sistema global se logra con el vector: $\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$



Para describir el movimiento del robot en términos de los componentes de movimiento, se mapea el movimiento desde los ejes de referencia a los del robot usando la matriz de rotación.

$$\dot{\xi}_R = R(\theta) \dot{\xi}_I \quad R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para el robot de la figura tenemos:

- Ruedas de Diámetro “r”
- El punto “P” situado en medio de las dos ruedas a una distancia “l” a cada lado del punto P.
- El ángulo de rotación “θ”
- Velocidad de giro de cada rueda “ $\dot{\phi}_1$ ” y “ $\dot{\phi}_2$ ”.

El modelo cinemático directo predice la velocidad completa del robot en el sistema de referencia XI,YI dado por:

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\phi}_1, \dot{\phi}_2)$$

Supongamos al robot alineado en el sistema local moviéndose en el eje X. La contribución de cada rueda a la velocidad del punto P en la dirección X es:

$$\dot{x}_{r1} = (1/2)r\dot{\phi}_1 \quad \dot{x}_{r2} = (1/2)r\dot{\phi}_2$$

Para la configuración de robot diferencial, simplemente se suman para hallar la velocidad del punto P en el eje X: $v_x = \frac{r \cdot \dot{\phi}_1}{2} + \frac{r \cdot \dot{\phi}_2}{2}$

Para la contribución local moviéndose en el eje Y tendremos que cada rueda para el caso de robot diferencial es: $v_y = 0$.

Por último la velocidad de rotación ω para cada rueda se obtiene como $\omega = \frac{v}{r}$, por lo tanto cada rueda aporta según su sentido de giro:

$$\omega_1 = \frac{r \cdot \dot{\phi}_1}{2 \cdot l} \quad y \quad \omega_2 = \frac{r \cdot \dot{\phi}_2}{2 \cdot l}$$

De donde resulta:

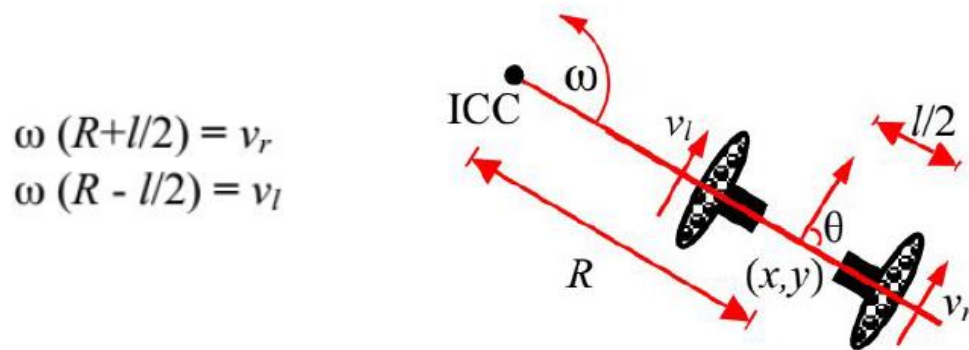
$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} \frac{r\dot{\phi}_1}{2} + \frac{r\dot{\phi}_2}{2} \\ 0 \\ \frac{r\dot{\phi}_1}{2l} + \frac{-r\dot{\phi}_2}{2l} \end{bmatrix}$$

Para un robot con manejo diferencial, el par de ruedas está montado sobre un eje común.

Si las ruedas están girando en el suelo (es decir, no hay deslizamiento), entonces hay un punto de ICC (proporcionado $v_r \neq v_l$) alrededor del cual ambas ruedas giran. Variando v_r , v_l y ICC, el robot se moverá en diferentes trayectorias.

Un concepto central para la derivación de las ecuaciones de cinemática es la velocidad angular ω del robot. Se define de la siguiente manera: cada rueda gira alrededor de ICC a lo largo de un círculo con radio r .

Dado que para cada r y v de cada rueda tenemos el mismo ω , de lo contrario las ruedas se moverían una respecto de la otra, por lo tanto, tenemos:



Donde R es la distancia entre ICC y el punto medio del eje de la rueda, y l es la longitud del eje de la rueda. Resolviendo las ecuaciones para ω y R tendremos:

$$R = l/2(v_l + v_r) / (v_r - v_l)$$

$$\omega = (v_r - v_l) / l$$

Con estas ecuaciones de R y ω podremos resolver el problema de cinemática directa del robot. Supongamos que el robot gira alrededor de ICC con velocidad angular ω por δt segundos. Esto cambiará el frente de acuerdo con: $\theta' = \omega \cdot \delta t + \theta$.

El centro de rotación ICC está dada por trigonometría básica como:

$$ICC = [ICC_x, ICC_y] = [x - R \cdot \sin\theta, y + R \cdot \cos\theta]$$

Luego, dada una posición de partida (x, y) , la nueva posición (x', y') puede ser calculada utilizando una matriz de rotación 2D:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\omega \cdot \delta t) & \sin(\omega \cdot \delta t) \\ \sin(\omega \cdot \delta t) & -\cos(\omega \cdot \delta t) \end{pmatrix} \begin{pmatrix} x - ICC_x \\ y - ICC_y \end{pmatrix} + \begin{pmatrix} ICC_x \\ ICC_y \end{pmatrix}$$

La rotación de cada rueda se puede medir, por ejemplo por los encoders de rueda. Estos sensores están montados en los ejes de las ruedas y entregan una señal binaria para cada paso que las ruedas giran.

Las señales alimentan a los contadores digitales de tal manera que $v\delta t$, las distancias recorridas desde el tiempo t al $t+\delta t$, lo cual produce un aumento de valor del contador n : $n = v\delta t$ paso. A partir de esto, v puede calcularse como: $v = n \text{ step}/\delta t$.

Por lo tanto, si el robot está en la posición inicial (x, y, θ) y se mueve n_l (cuentas del encoder izquierdo) y n_r (cuentas del encoder derecho) cuentas, durante un intervalo de tiempo δt , la nueva posición (x', y', θ') viene dada por:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega. \delta t) & -\sin(\omega. \delta t) & 0 \\ \sin(\omega. \delta t) & \cos(\omega. \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x - ICCx \\ y - ICCy \\ 0 \end{bmatrix} + \begin{bmatrix} ICCx \\ ICCy \\ \omega. \delta t \end{bmatrix}$$

Donde:

$$R = \frac{l (n_l + n_r)}{2 (n_l - n_r)} ; \omega. \delta t = \frac{(n_l + n_r) \text{step}}{l} ; ICC = [x - R. \sin\theta, y + R. \cos\theta]$$

Desarrollo

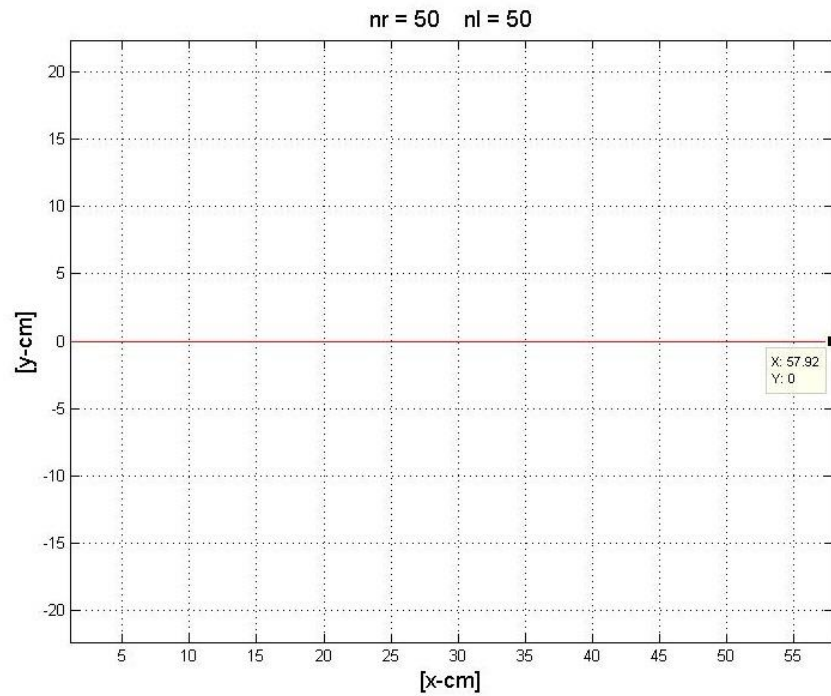
Utilizando el software Matlab 2013 simulamos dos trayectorias arbitrariamente elegidas:

1. 50 pasos para la rueda derecha y 50 para la izquierda
2. 50 pasos para la rueda derecha y 100 para la izquierda

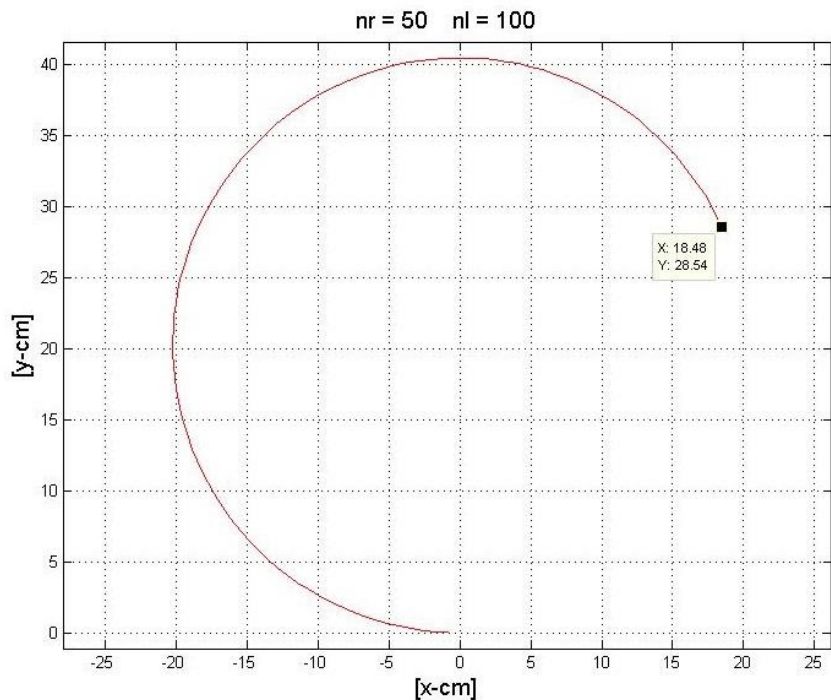
Para ambos casos asumimos coordenadas iniciales: $\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

Los resultados obtenidos fueron los siguientes:

1. $\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} 57.92 \\ 0 \\ 0 \end{pmatrix}$



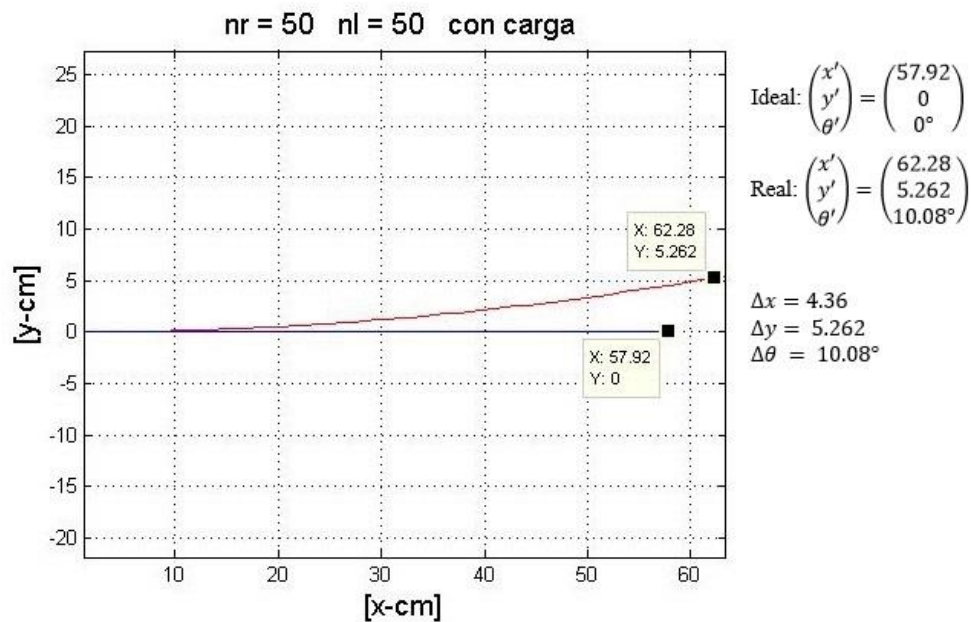
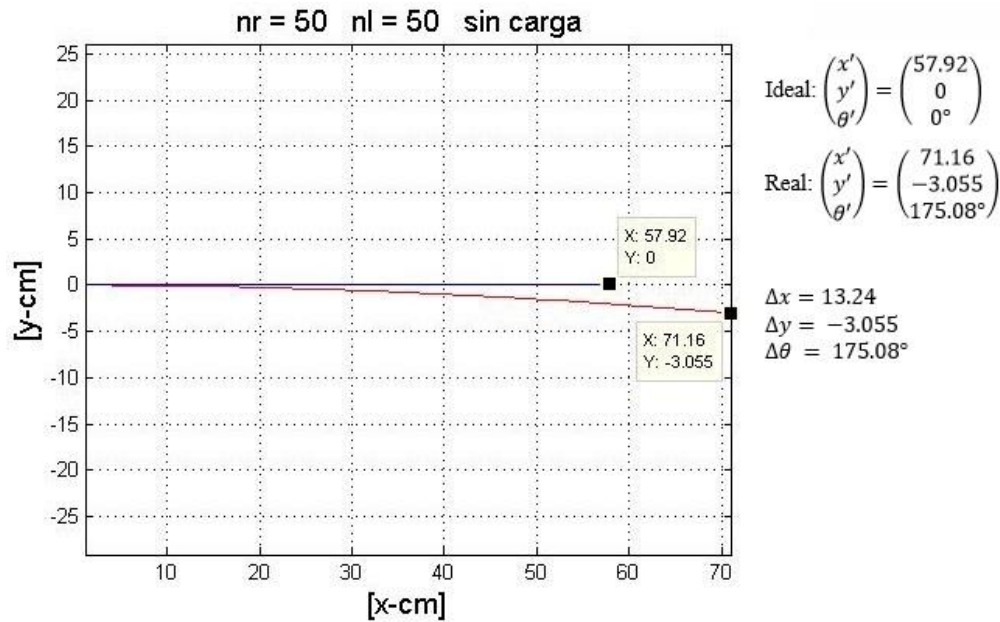
2. $\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} 18.48 \\ 28.54 \\ 245.83^\circ \end{pmatrix}$

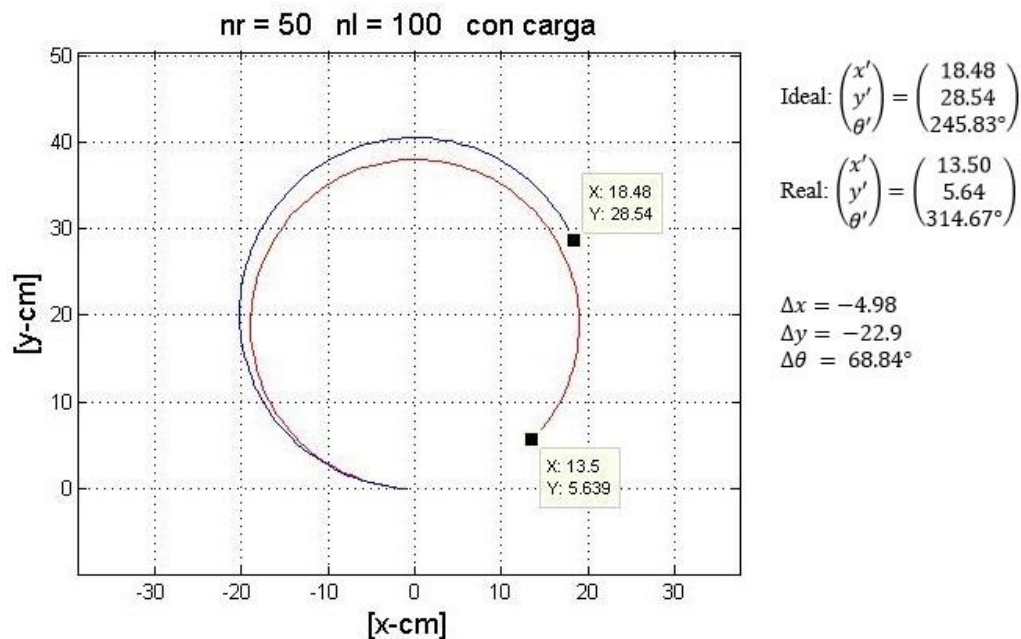
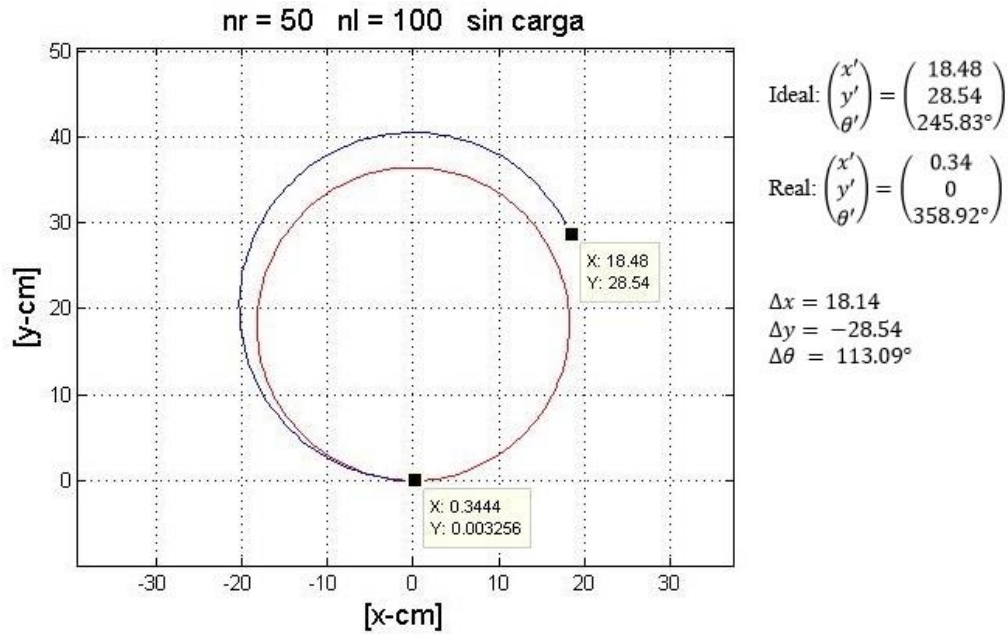


El paso siguiente fue programar el robot Multiplo N6 para que realice las trayectorias establecidas y verificar a través de la medición de sus encoders el resultado obtenido. Fue necesario definir, también de forma arbitraria, un tiempo para la trayectoria, en este caso 2 segundos.

Ambas trayectorias fueron probadas en dos condiciones: primero sin que las ruedas hagan contacto en ninguna superficie (prueba a la que denominaremos sin carga) y luego apoyando el robot sobre una superficie de madera laminada (con carga). De esta forma buscamos aislar en la primera instancia los efectos de la superficie sobre la trayectoria a realizar.

A continuación se muestran los resultados obtenidos:





Se pueden observar diferencias considerables entre las trayectorias ideal y real en todos los casos.

Para entender estas diferencias, empezamos haciendo un estudio más detallado del movimiento de las ruedas, analizando la señal analógica producida por los encoders en cada caso. Lo haremos primero para el caso de 50 pasos por rueda, sin carga, y luego mostraremos los resultados obtenidos en cada caso.

Calculo de la velocidad de cada rueda:

Conociendo la cantidad de pasos que tiene que dar cada rueda (R_{steps} y L_{steps}), diámetro de las ruedas (ϕ_{rueda}) y la cantidad de pasos del encoder ($N_{porVuelta}$) se puede calcular la distancia que recorre cada rueda en esa cantidad de pasos:

Perímetro de la rueda: $P_{rueda} = \pi \cdot \phi_{rueda} = \pi \cdot 59mm = 185.26mm$

Distancia recorrida por las ruedas:

$$Dist_{right} = \frac{P_{rueda}}{N_{porVuelta}} \cdot R_{steps} = \frac{185.26mm}{16} \cdot 50 = 578.9375mm$$

$$Dist_{left} = \frac{P_{rueda}}{N_{porVuelta}} \cdot L_{steps} = \frac{185.26mm}{16} \cdot 50 = 578.9375mm$$

Dividiendo por el tiempo obtenemos la velocidad tangencial de la rueda:

$$v_{right} = v_{left} = \frac{Dist_{right}}{t} = \frac{578.9375mm}{2s} = 289.4688 \frac{mm}{s}$$

$$v_{right} = v_{left} = 28.9469 \frac{cm}{s}$$

Luego es necesario convertir la velocidad calculada a r.p.m.:

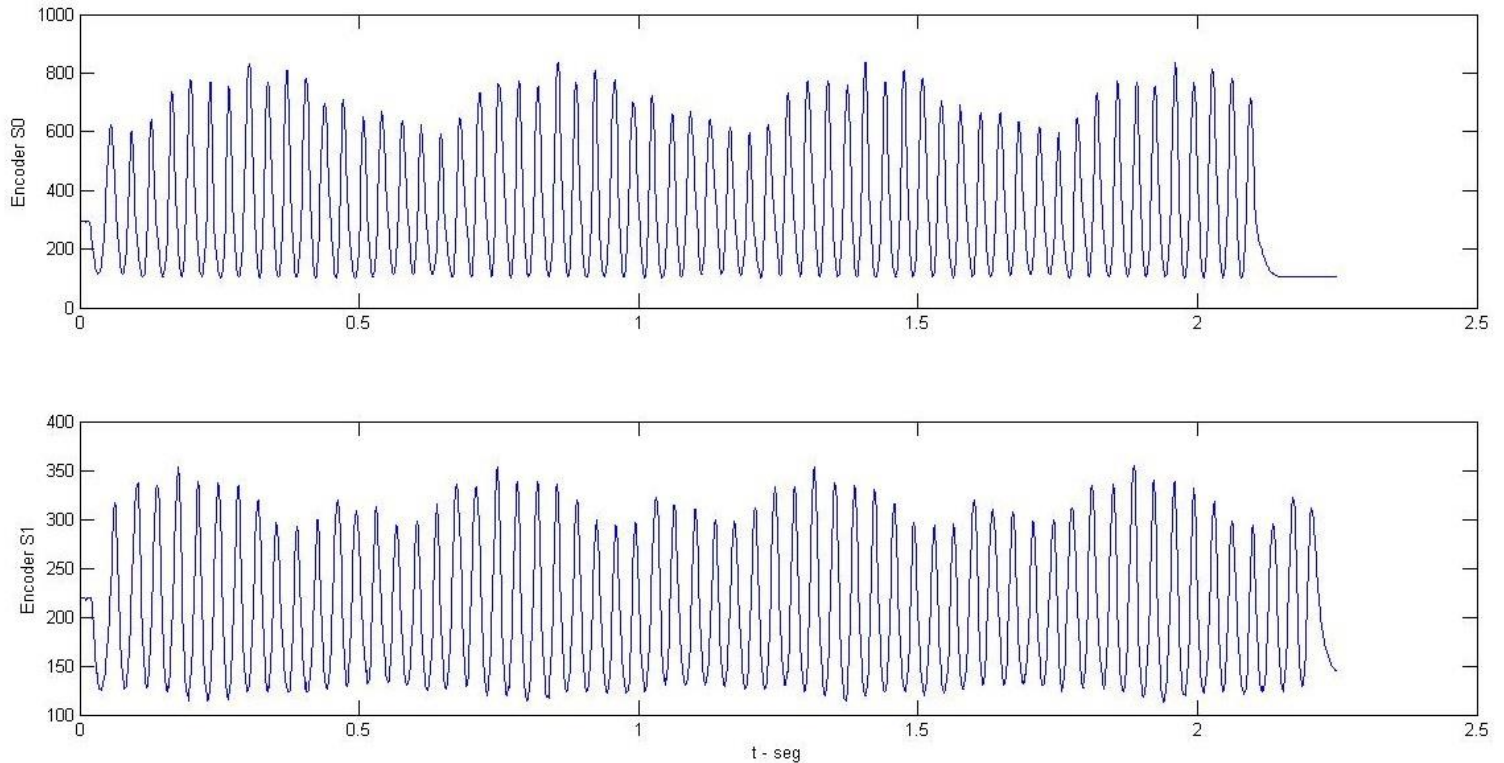
$$v_{rpm} = \frac{v \cdot 60}{P_{rueda}} = 93.75 \text{ rpm}$$

Por último la función “motor.setspeed” de las librerías de DuinoBot necesita que se ingrese como parámetro la velocidad en un valor porcentual a la velocidad máxima del motor que según el fabricante es de 200 rpm:

$$v_{\%} = \frac{v_{rpm} \cdot 100}{200 \text{ rpm}} = 46.875\%$$

Medición de la velocidad de cada rueda:

Utilizando los encoders ubicados en cada rueda y tomando muestras con una frecuencia de muestreo de $F_s = 1\text{KHz}$, obtuvimos los gráficos que se muestran a continuación:

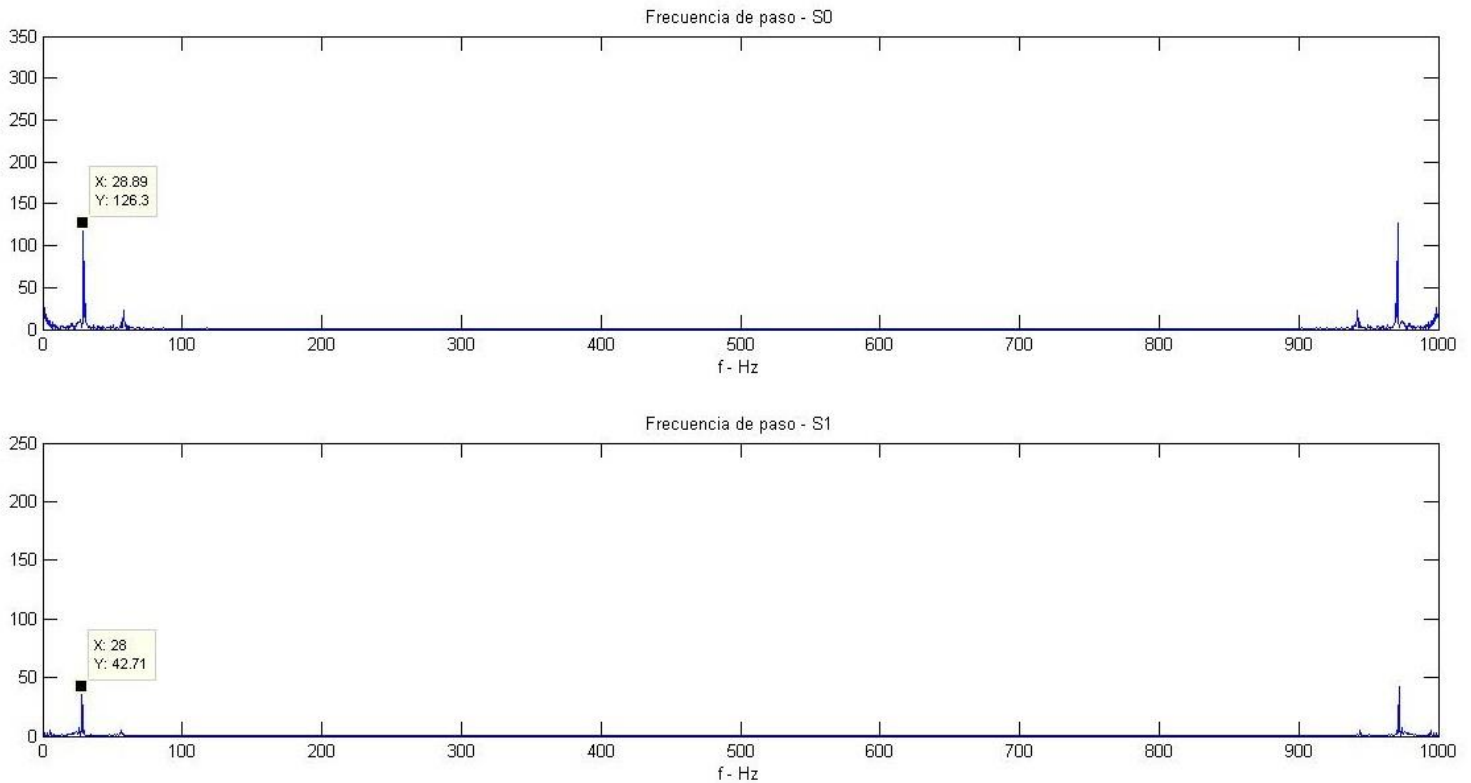


El sensor S0 está asociado a la rueda derecha y el S1 a la izquierda. El eje de abscisas representa el tiempo. Los ejes de ordenadas representan los valores de las entradas digitales a las cuales están asociados los encoders.

Los picos positivos de la señal representan los blancos de la rueda y los picos negativos los negros. Se observa que el valor de los picos no es constante, lo cual indica que la rueda no es uniforme, es decir que las franjas de cada paso del encoder no son iguales una respecto de las otras.

Por otro lado, a pesar de que se configuró al robot para moverse durante dos segundos, se observa que el mismo se mueve por un tiempo superior a este. Motivo por el cual la medición arroja más pasos que los calculados: 61 pasos para la rueda derecha y 62 para la izquierda. Este error podría ser debido a imprecisiones en la base de tiempo del Arduino Uno.

Teóricamente el tiempo entre paso y paso es de $\Delta t_{porPaso} = \frac{2s}{50\text{ pasos}} = 40\text{ ms}$. Para medir este tiempo a partir del gráfico anterior aplicamos un algoritmo de FFT (Fast Fourier Transform) a la señal medida:



La transformada de Fourier de la señal muestra una componente en 28.89Hz para la rueda derecha y una en 28Hz para la izquierda, entonces:

$$\text{Rueda derecha: } \Delta t_{porPaso} = \frac{1}{28.89\text{Hz}} = 34.61\text{ms}$$

$$\text{Rueda izquierda: } \Delta t_{porPaso} = \frac{1}{28\text{Hz}} = 35.71\text{ms}$$

	$\Delta t_{porPaso}$	$\Delta t_{porVuelta}$	v_{rpm}	$v_{cm/s}$	Δv
Rueda derecha	34.61 ms	553.82 ms	108.34	33.45 cm/s	+ 4.50 cm/s
Rueda izquierda	35.71 ms	571.43 ms	105	32.42 cm/s	+ 3.47 cm/s

Esta diferencia en la velocidad de cada rueda puede deberse por un lado a que la velocidad máxima de los motores no sea exactamente 200rpm como indica el fabricante y por otro lado a imprecisiones en la calibración de fábrica del robot.

Repetimos las mediciones realizadas para ambas trayectorias planteadas en las condiciones antes mencionadas. El cuadro que se muestra a continuación muestra los resultados obtenidos:

Trayectoria	nR=50 nL=50				
	Ideal	Sin carga		Con carga	
Rueda	Der/lzq	Derecha	Izquierda	Derecha	Izquierda
Tiempo (s)	2	2.132	2.228	2.195	2.235
$\Delta_{tiempo}\%$	-	6.6	11.4	9.75	11.75
Cant. De pasos	50	61	62	56	54
$\Delta_{pasos}\%$	-	22	24	12	8
$\Delta t_{porPaso}$ (ms)	40	34.61	35.71	36.89	39.48
$\Delta t_{porVuelta}$ (ms)	640	553.82	571.43	590.24	631.68
V (rpm)	93.75	108.34	105	101.65	94.98
V (cm/s)	28.95	33.45	32.42	31.39	29.33
Δv (%)	-	15.54	11.99	8.43	1.31

Trayectoria	nR=50 nL=100					
	Ideal		Sin carga		Con carga	
Rueda	Derecha	Izquierda	Derecha	Izquierda	Derecha	Izquierda
Tiempo (s)	2	2	2.144	2.232	2.14	2.24
$\Delta_{tiempo}\%$	-	-	7.2	11.6	7	12
Cant. De pasos	50	100	62	135	58	122
$\Delta_{pasos}\%$	-	-	24	35	16	22
$\Delta t_{porPaso}$ (ms)	40	20	33.58	16.19	34.61	17.44
$\Delta t_{porVuelta}$ (ms)	640	320	537.27	258.98	553.82	279.09
V (rpm)	93.75	187.5	111.68	231.68	108.34	214.99
V (cm/s)	28.95	57.89	34.48	71.53	33.45	66.38
Δv (%)	-	-	19.10	23.56	15.54	14.67

Conclusiones:

Podemos concluir entonces que la diferencia entre la trayectoria idealmente trazada y la trayectoria real obtenida con el Multiplo N6 se debe a una sumatoria de efectos enumerados a continuación:

- Imprecisiones en la base de tiempo.
- Falta de uniformidad en los encoders de las ruedas.
- Por desviaciones propias del fabricante la velocidad real de las ruedas difiere de la velocidad programada.
- Falta de simetría entre ambas ruedas.

Se observa una disminución en los errores de velocidad cuando el robot se prueba con carga, lo cual puede significar que esta prueba se acerque un poco más a las condiciones al momento de la calibración.

Por algún motivo desconocido en todas las pruebas la rueda izquierda demora más de una décima de segundo más en detenerse lo cual provoco una desviación notoria en ambas trayectorias, más evidente en la trayectoria rectilínea.

Robótica – Trabajo Práctico N° 2:

Dinámica de un robot móvil diferencial

Integrantes:

- Federico Coppede
- Matías Paramidani

Consignas:

El desarrollo del TP2 consistirá en la modelación dinámica del robot educativo N6 de la empresa robogroup, el cual será el hardware disponible para probar el modelo obtenido durante la realización de la práctica.

- Calcular el modelo dinámico en Matlab para una trayectoria establecida.
- Implementar dicho modelo en el N6.
- Comparar El modelo Matlab con la programación real y extraer conclusiones y gráficos.

Hipótesis básicas simplificadoras:

- El robot se mueve sobre una superficie plana sin irregularidades ni defectos del suelo.
- Los ejes de guiado son perpendiculares al suelo.
- Se supone que las ruedas se mueven con rodadura pura (el deslizamiento es despreciable)
- Robot sin partes flexibles.
- Todas las partes utilizadas en la modelación coinciden con el modelo real del robot.

Introducción

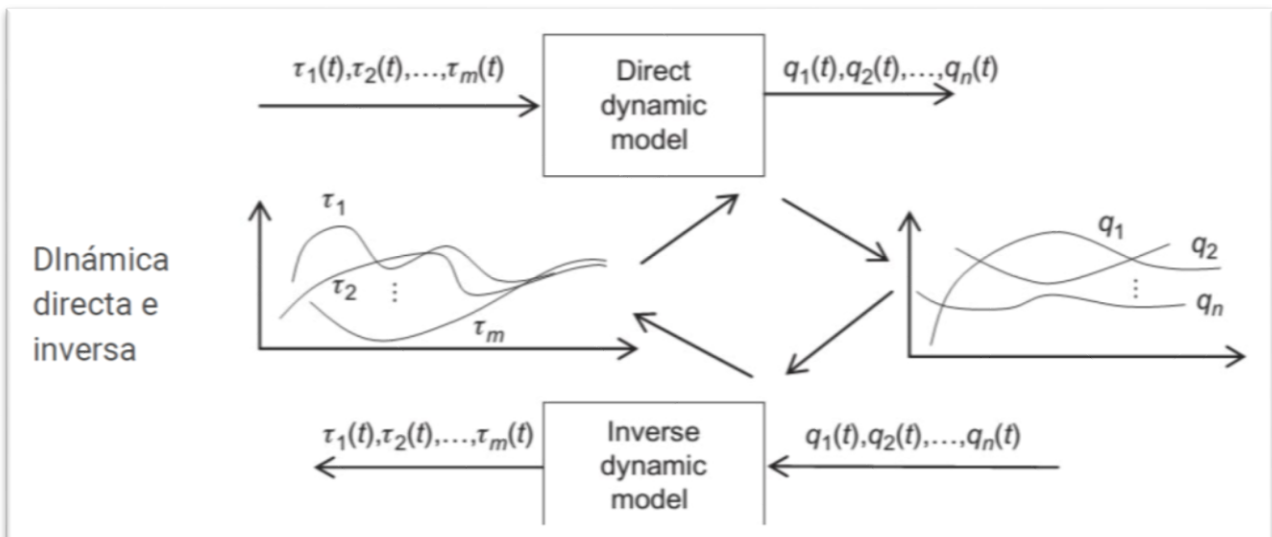
El robot móvil en cuestión tiene 3 grados de libertad respecto a una referencia: posición en el plano (X,Y) y orientación (Θ). Idealmente, independientemente de donde inicie, el robot debe poder moverse a cualquier posición y orientación (X,Y, Θ).

La dinámica del robot se deriva de las ecuaciones dinámicas del movimiento del robot. Partiendo de dos metodologías:

- Newton-Euler
- Lagrange

Siendo la complejidad del método de Newton-Euler es $O(n)$, mientras que el de Lagrange será $O(n^3)$, en donde n es el número de grados de libertad.

Modelo dinámico directo e inverso:

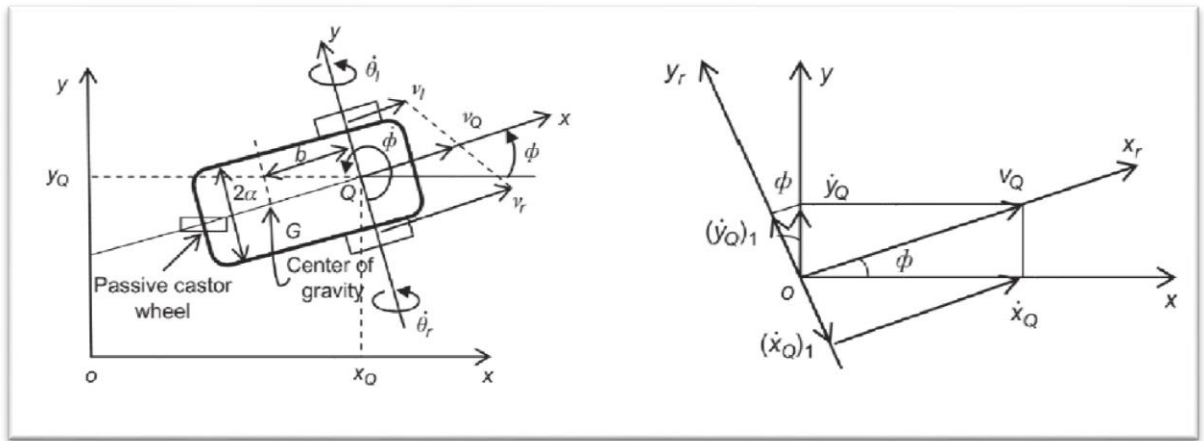


El objetivo de este estudio, como se muestra en el cuadro explicativo es, en función de los parámetros físicos del objeto (masa, momentos de inercia, centro de gravedad, etc.) obtener la respuesta dinámica con respecto a una referencia al aplicar un torque en determinada parte del objeto (ruedas).

Cuando se parte de las propiedades físicas y de los requerimientos de aceleración del objeto, para llegar al torque necesario en los motores, se trata de un estudio de *dinámica inversa*.

El análisis planteado en este caso contempla dos casos posibles para el desarrollo de las ecuaciones:

- 1) Caso 1 (modelo simplificado). Cuando el COG (Centro de Gravedad) coincide con el punto central entre ambas ruedas del robot diferencial (Q), no existe fricción en las ruedas, y el conjunto motor – rueda tiene inercia cero.



Partiendo de las ecuaciones de Newton-Euler se obtiene:

$$\dot{v} = \frac{(t_r + t_l)}{m * r}$$

$$\dot{w} = \frac{2 * a}{I * r} * (t_r - t_l)$$

Donde:

\dot{v} : Aceleración

t_r : Torque en la rueda derecha

t_l : Torque en la rueda izquierda

I : Momento de inercia

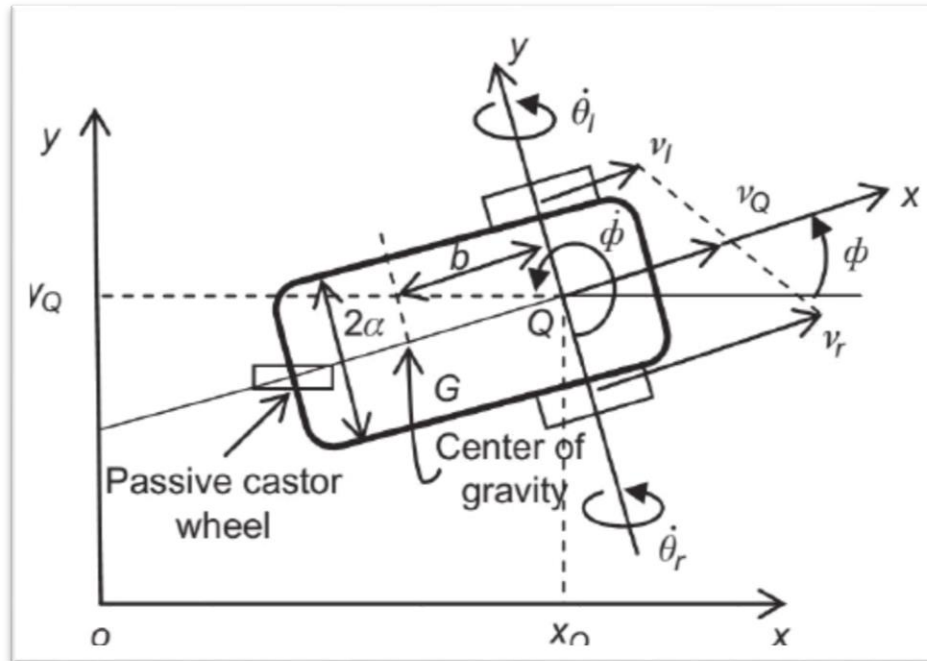
r : Radio de las ruedas

m : Masa del robot

\dot{w} : Aceleración angular

a : Distancia entre ruedas

- 2) Caso 2 (modelo completo). Cuando no coincide el punto Q con el centro de gravedad, existe fricción con el mismo coeficiente en ambas ruedas y el conjunto motor – rueda no tiene inercia cero.



Para este caso de análisis hay que contemplar la distancia "b" que existe entre el punto Q y el centro de gravedad.

Partiendo del modelo de Lagrange y dado que hay movimiento solo en el plano x-y, se plantea el lagrangiano sabiendo que la energía potencial es el cero, y este resulta igual a la energía cinética dada por $K = K_1 + K_2 + K_3$

Siendo:

$$K_1 = \frac{1}{2}mv_G^2 = \frac{1}{2}m(\dot{x}_G^2 + \dot{y}_G^2)$$

$$K_2 = \frac{1}{2} I_Q \dot{\phi}^2$$

$$K_3 = \frac{1}{2} I_o \dot{\theta}_r^2 + \frac{1}{2} I_o \dot{\theta}_l^2$$

$$\begin{aligned}\dot{x}_G &= \dot{x}_Q + b\dot{\phi} \sin \phi \\ \dot{y}_G &= \dot{y}_Q - b\dot{\phi} \cos \phi\end{aligned}$$

La energía cinética se expresa directamente en términos de las velocidades angulares de las ruedas motrices. La función de Lagrange L es igual a K, ya que el robot se mueve en el plano horizontal y por lo tanto la energía potencial P es cero, siendo:

$$\begin{aligned}\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{\theta}_r} \right) - \frac{\partial K}{\partial \theta_r} &= \tau_r - \beta \dot{\theta}_r \\ \frac{d}{dt} \left(\frac{\partial K}{\partial \dot{\theta}_l} \right) - \frac{\partial K}{\partial \theta_l} &= \tau_l - \beta \dot{\theta}_l\end{aligned}$$

Donde β es el coeficiente de fricción de las ruedas y t_l, t_r son el torque en cada rueda.

Finalmente se obtiene:

$$\begin{aligned}D_{11}\ddot{\theta}_r + D_{12}\ddot{\theta}_l + \beta\dot{\theta}_r &= \tau_r \\ D_{21}\ddot{\theta}_r + D_{22}\ddot{\theta}_l + \beta\dot{\theta}_l &= \tau_l\end{aligned}$$

Donde:

$$\begin{aligned}D_{11} = D_{22} &= \left[\frac{mr^2}{4} + \frac{(I_Q + mb^2)r^2}{8a^2} + I_o \right] \\ D_{12} = D_{21} &= \left[\frac{mr^2}{4} - \frac{(I_Q + mb^2)r^2}{8a^2} \right]\end{aligned}$$

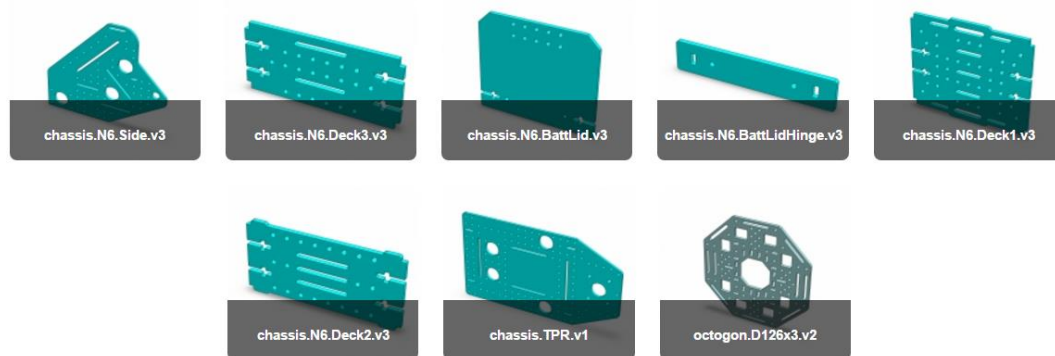
Desarrollo

La dificultad en la aplicación de este método radica en la correcta medición de todos los parámetros que incluyen al modelo físico, si bien algunos son fácilmente obtenibles como son el peso, radio de las ruedas, distancia entre rudas etc., otros parámetros son más difíciles de obtener, centro de gravedad y momento de inercia principalmente.

Para obtener todos los parámetros físicos se utilizó un CAD mecánico, por medio del cual, luego de diseñar y asignar los materiales de cada pieza apropiadamente, se obtuvieron todos los datos necesarios para el desarrollo de este trabajo.

Pasos para la obtención de todos los parámetros:

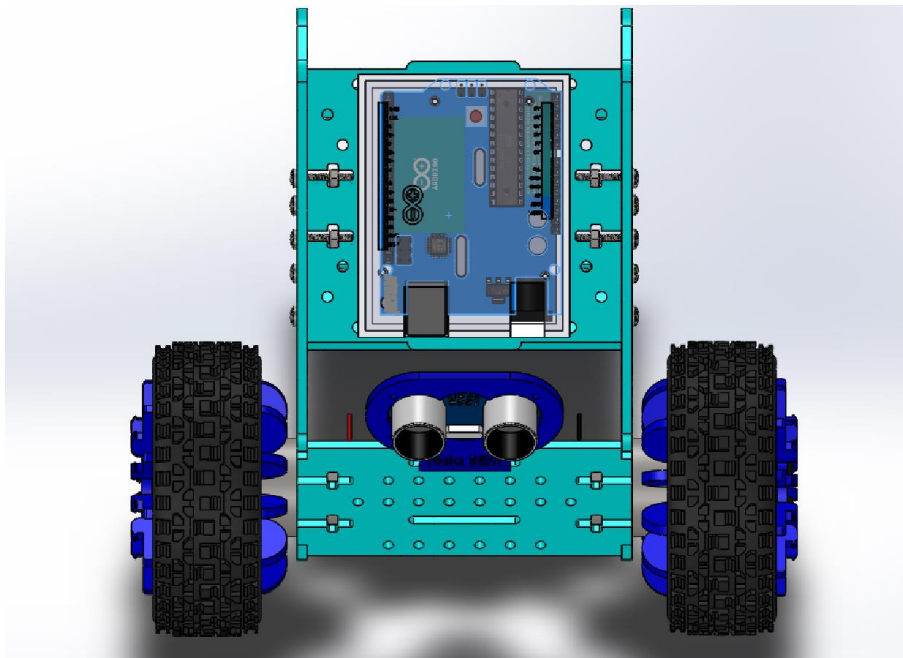
- 1) El primer paso consiste en la elección del CAD mecánico, en este caso y por simplicidad elegimos el software SolidWorks, ya que la mayoría de las piezas estaban diseñadas por el fabricante del robot y esto nos permitió obtener un modelo muy aproximado al real.
- 2) Partiendo de las piezas básicas se armó un modelo aproximado al cual le harían falta algunos retoques para obtener el modelo final. La mayoría de las piezas utilizadas en este paso se pueden descargar de: <http://multiplo.org/es/chassis-1/>

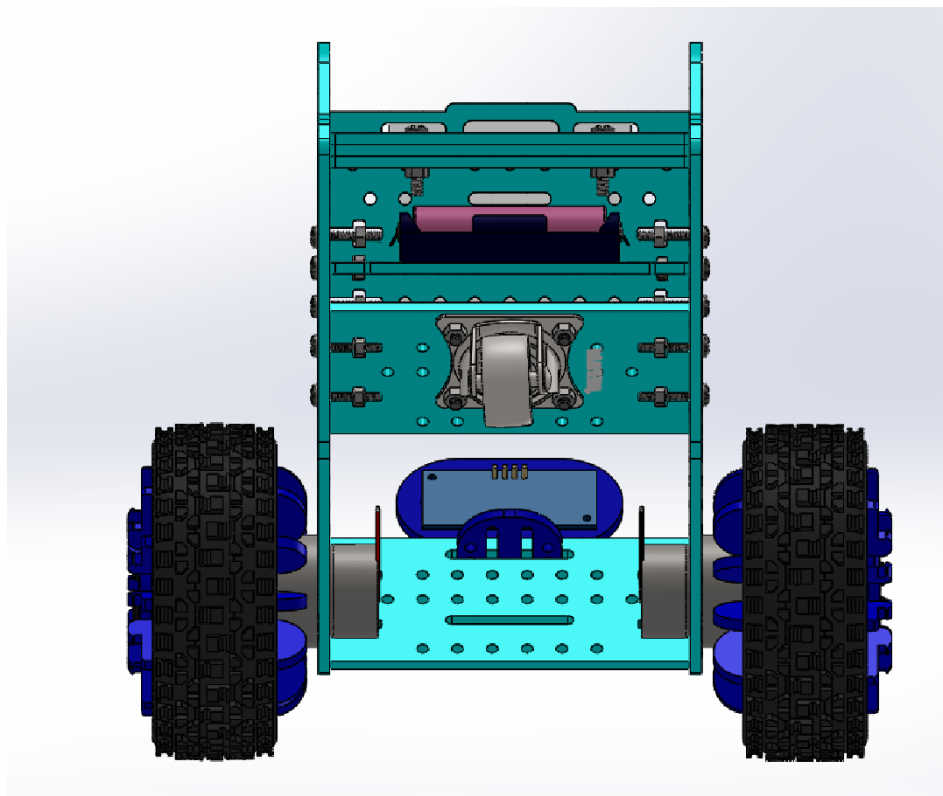
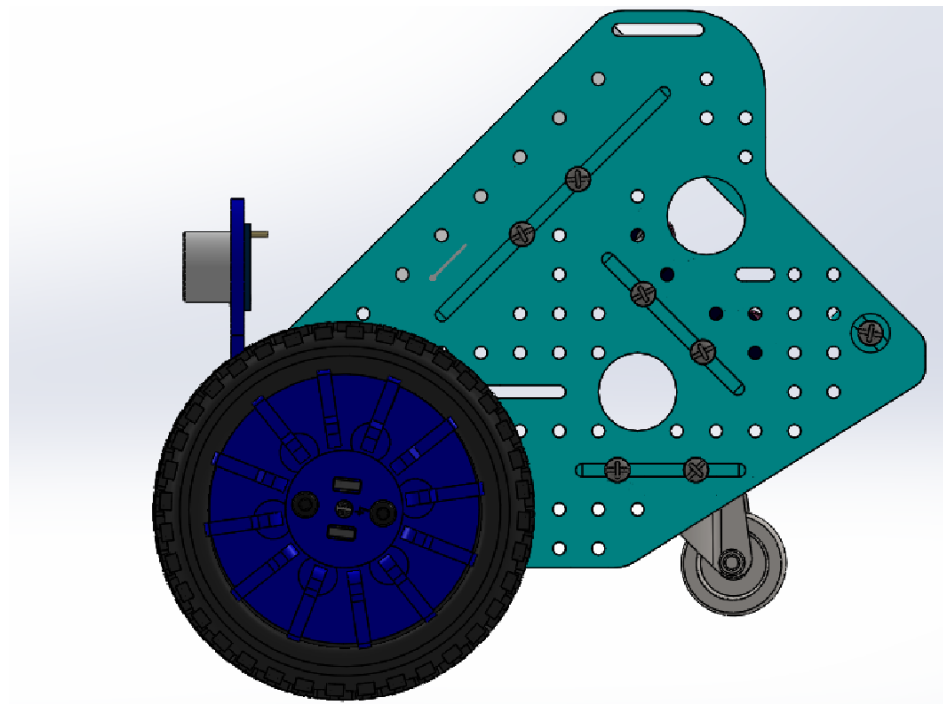


Como se puede ver, están disponible la mayoría de las piezas que componen la estructura principal del robot.

- 3) Partiendo de la estructura básica, se mejoró el modelo agregando ruedas, tornillos y tuercas. Todas las piezas utilizadas en este paso se pueden descargar de: <https://github.com/multiplo/mechanics/>
- 4) El diseño se finalizó agregando las partes restantes que más influyen en la dinámica del movimiento, estas son:
 - Motores (2 motores DC)
 - Pilas (3 AA)
 - Placa principal (Arduino)
 - Sensor ultrasónico

Capturas tomadas del diseño finalizado:





Datos obtenidos:

Masa = 745.33 gr

Volumen = 436039.84 mm³

Área de superficie = 382041.21 mm²

Centro de masa:

X = -26.93 mm

Y = -12.82 mm

Z = -0.01 mm

Ejes principales de inercia y momentos principales de inercia: (gr*mm²)

Medido desde el centro de masa.

lx = (-0.01, 0.00, -1.00) Px = 1957887.03

ly = (-0.87, 0.50, 0.01) Py = 2404760.73

lz = (0.50, 0.87, 0.00) Pz = 3369170.11

Momentos de inercia: (gr*mm²)

Obtenidos en el centro de masa y alineados con el sistema de coordenadas de resultados.

Lxx = 2641550.69 Lxy = -415111.57 Lxz = 3718.70

Lyx = -415111.57 Lyy = 3132345.85 Lyz = -1344.39

Lzx = 3718.70 Lzy = -1344.39 Lzz = 1957921.33

Momentos de inercia: (gr*mm²)

Medido desde el sistema de coordenadas de salida.

lxx = 2764101.88 lxy = -157738.54 lxz = 3998.22

lyx = -157738.54 lyy = 3672862.61 lyz = -1211.29

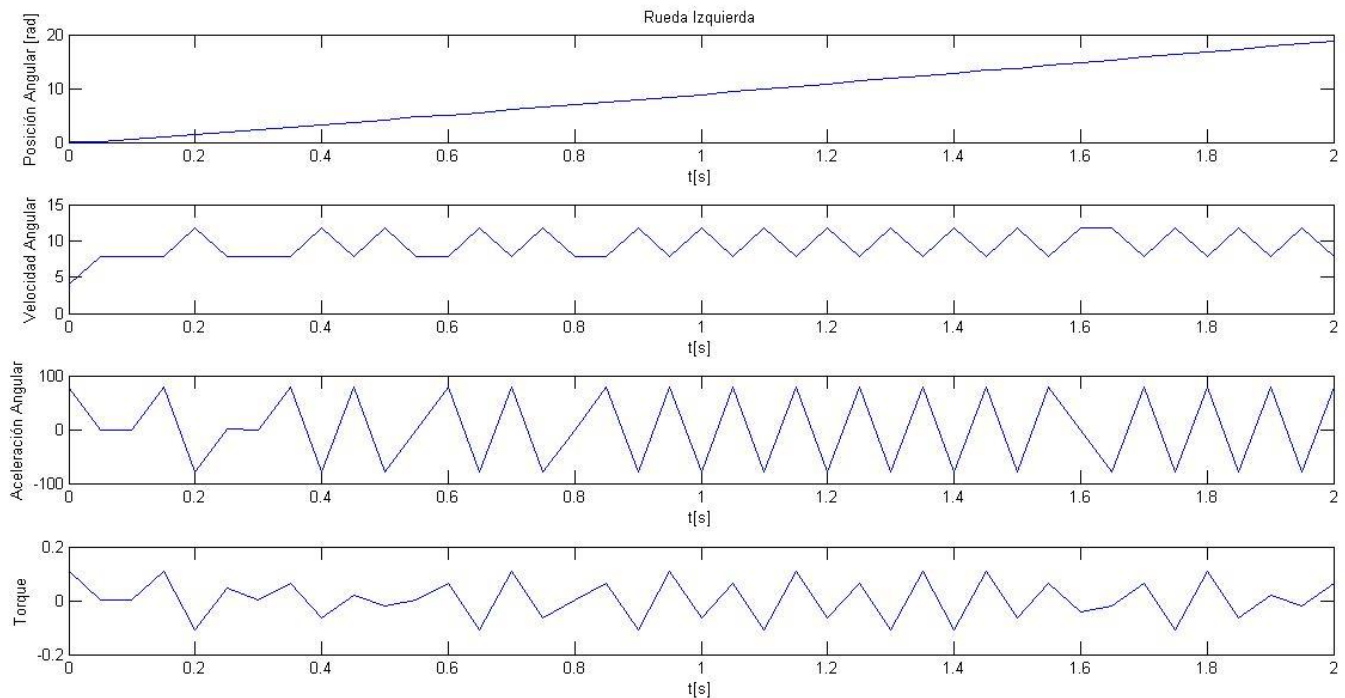
lzx = 3998.22 lzy = -1211.29 lzz = 2620989.00

Mediciones

Una vez obtenidos los parámetros del robot, procedimos a analizar las magnitudes en cuestión, posición angular, velocidad angular, aceleración angular y torque, para la trayectoria realizada en el trabajo práctico número 1.

Como dicha trayectoria era a velocidad constante idealmente deberíamos obtener una aceleración y un torque nulos.

Sin embargo los resultados obtenidos fueron los siguientes:



Se observa que la posición angular no es una recta perfecta, esto se debe a que la velocidad en ambas ruedas no es perfectamente constante. Sin embargo el valor medio de la velocidad coincide con el esperado.

Debido a las variaciones, positivas y negativas, de la velocidad se obtiene una aceleración con picos aún más pronunciados y con valores positivos (aumento de velocidad) y negativos (disminución de la velocidad). Sin embargo otra vez el valor medio de la aceleración da cero como era de esperarse.

El torque a su vez presenta picos positivos y negativos, debido a las variaciones en la aceleración antes mencionados, y también posee el valor medio nulo esperado.

A continuación modificamos la trayectoria del robot de la siguiente manera:

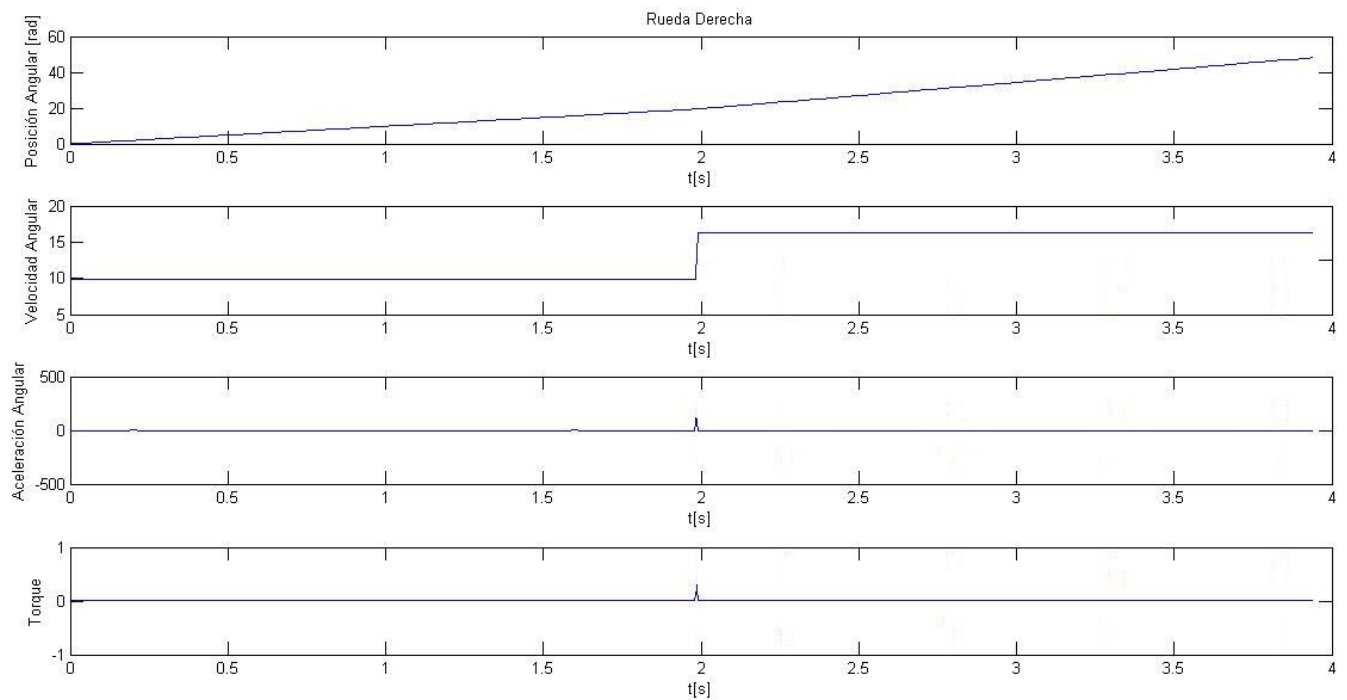
- Rueda derecha: 50 pasos en 2 segundos y luego 75 pasos en dos segundos
- Rueda izquierda: 50 pasos en 2 segundos y luego 75 pasos en dos segundos

Repitiendo los cálculos realizados en el tp1 la velocidad teórica de cada rueda es:

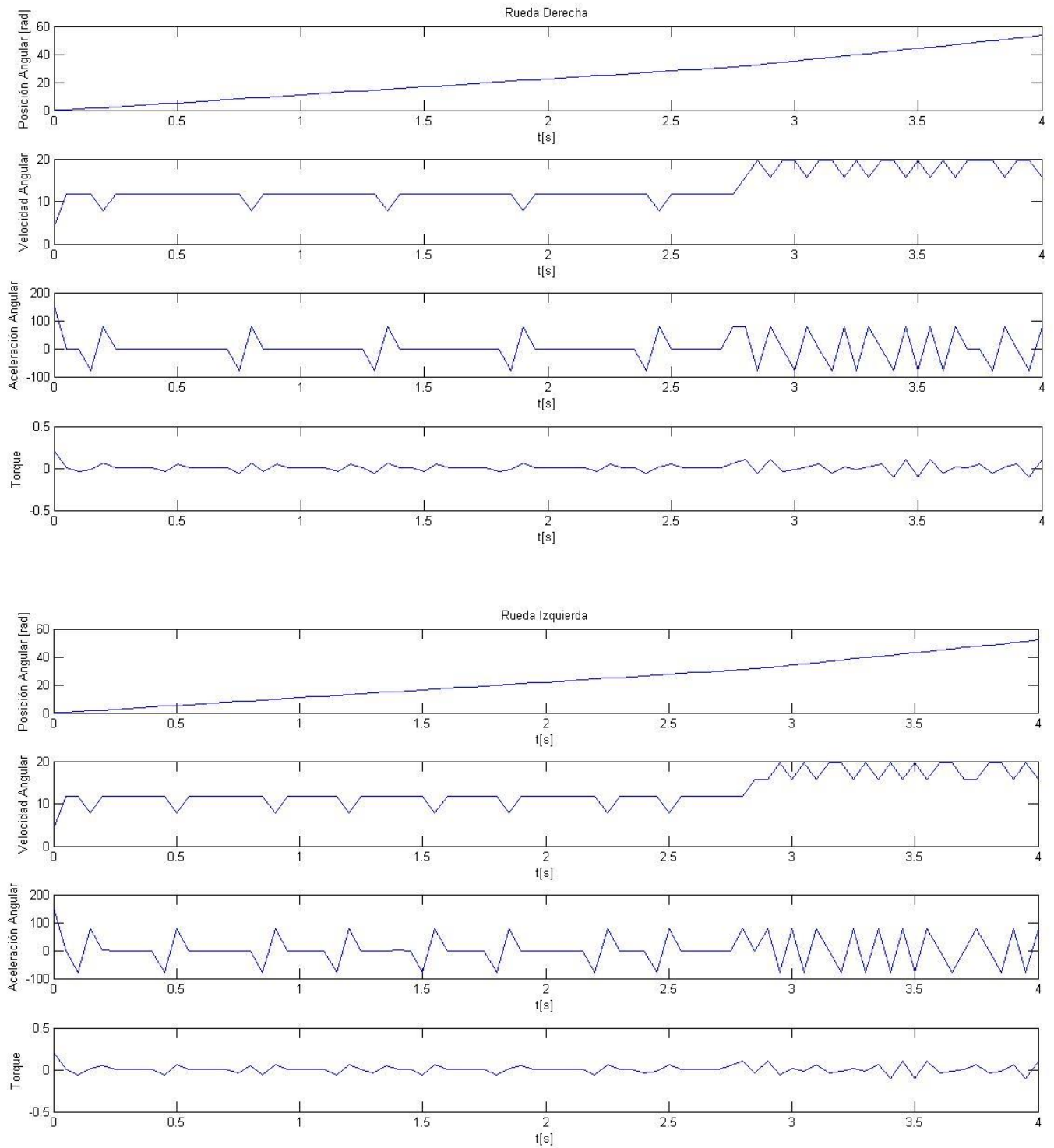
$$\text{Para } 0s < t < 2s: v_{rpm} = \frac{v \cdot 60}{P_{rueda}} = 93.75 \text{ rpm} = 9.81 \text{ rad/s}$$

$$\text{Para } 2s < t < 4s: v_{rpm} = \frac{v \cdot 60}{P_{rueda}} = 140.62 \text{ rpm} = 14.73 \text{ rad/s}$$

En teoría únicamente debería aparecer una aceleración en la transición entre ambos segmentos de la trayectoria. El fabricante no da ningún dato sobre el tiempo que tarda cada motor en aumentar la velocidad, es por eso que se hace imposible calcular la aceleración teórica y más aún el torque, pero la forma de ambas señales en función del tiempo idealmente es la siguiente:



Para la trayectoria programada en el Multiplo N6 los resultados fueron los siguientes:



De las mediciones se puede observar varias cosas:

- La velocidad de las ruedas no cambia a los 2s sino un tiempo más tarde, aproximadamente 2.75s, este mismo efecto se observó en el trabajo práctico nro. 1, puede deberse a un error en la base de tiempo del Arduino y/o a un retardo de activación propio de la función SetSpeed de la librería DCMotor de DuinoBot. Sin embargo las velocidades medias de cada tramo de la trayectoria se acercan bastante a las esperadas, a pesar del error que ya se demostró en el trabajo práctico anterior.
- Al igual que en la medición anterior la velocidad no es perfectamente constante en cada tramo de la trayectoria, esto hace que la aceleración no sea nula en cada tramo y por lo tanto lo mismo pasa con el torque. El error se va arrastrando, debido al algoritmo de cálculo, y se vuelve más evidente para el torque.
- Debido a este error antes mencionado el torque realizado por el motor para cambiar de velocidad no sobresale del resto de los valores temporales del mismo.
- Es posible que se puedan optimizar las mediciones utilizando un algoritmo un poco más complejo.

Robótica – Trabajo Práctico Final:

Compilador grafico para robot diferencial

Integrantes:

- Federico Coppede
- Matías Paramidani

Consignas:

El desarrollo del TP Final consistirá en el desarrollo de un compilador para el Múltiplo N6. Para esto diseñamos un compilador que genera un código de salida a partir de comando ingresados gráficamente, permitiendo generar trayectorias de forma simple e intuitiva utilizando solo los botones del mouse.

Introducción

El objetivo principal de este proyecto es simplificar la generación de código a partir de una trayectoria deseada del robot, permitiendo al usuario independizarse del tedioso cálculo matemático requerido para obtener los parámetros de configuración de los motores del robot diferencial.

La generación de trayectorias se hace a partir de una interfaz de usuario sencilla, en la que cada punto sobre un mapa representa el principio, y/o final de una trayectoria, pudiendo ser estas, trayectorias rectas, o curvas.

Llamaremos “mapa” a una imagen ingresada por el usuario que representa la vista aérea de un terreno en el cual podrían encontrarse obstáculos que deben ser atravesados especificando puntos por los que debe pasar el robot.

Esto permite al usuario, partiendo de parámetros básicos, como por ejemplo, la velocidad de desplazamiento deseada, obtener el código listo para cargar utilizando DuinoPack.

El código se ejecuta en Linux y requiere tener instalado OpenCV 2.4 o superior, y el archivo de salida es compatible con la geometría del robot diferencial Múltiplo N6.

Interfaz de usuario

Para ejecutar el código se debe abrir una terminal en Linux (previamente se debe instalar OpenCV 2.4) pasando como parámetro un “mapa” el cual se usara como base para generar las trayectorias del robot.

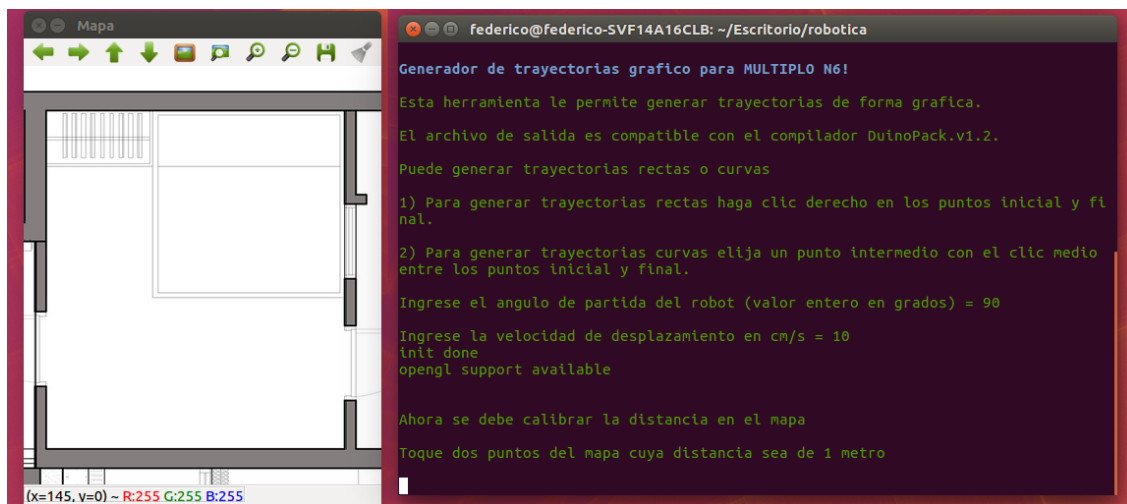
Una vez ejecutado el programa, se deberán configurar 2 parámetros, el ángulo inicial del robot, y la velocidad de desplazamiento deseada en cm/s (todos los movimientos del robot se harán a velocidad constante).

Nota: El ángulo que se pasa como parámetro sigue las coordenadas X, Y estándar, y sentido de giro anti horario.

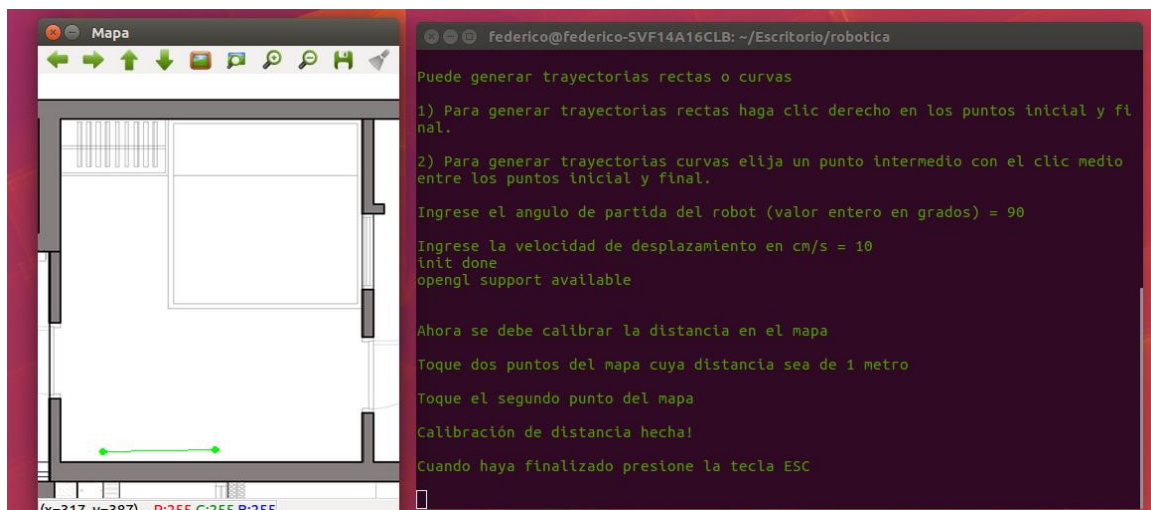
Ejemplo:

`./programa mapa.png`

Cuando se hayan pasado los dos parámetros necesarios, se abrirá la siguiente ventana:



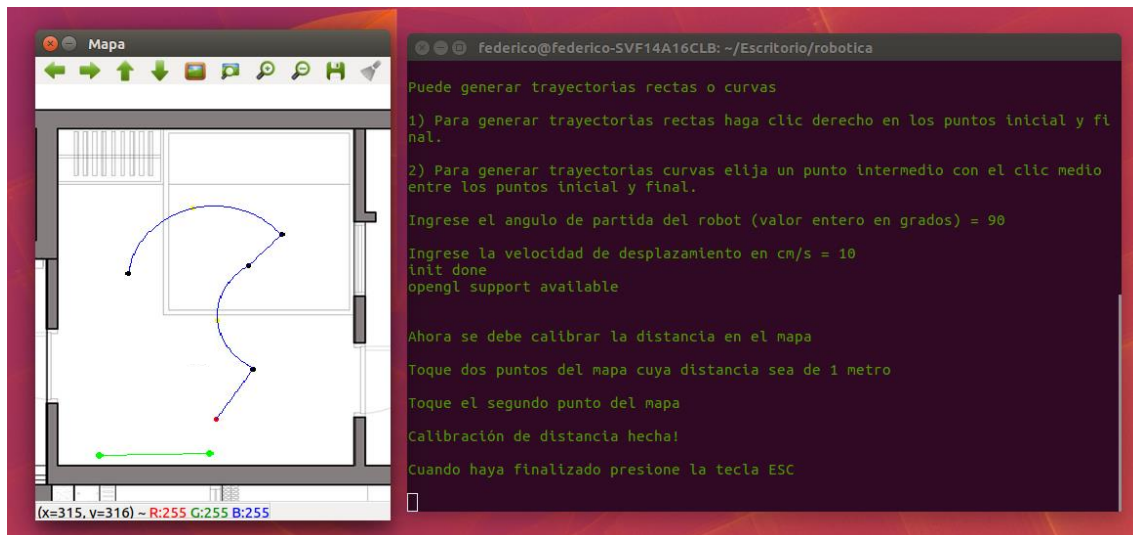
En este punto lo primero que se deberá hacer es calibrar la distancia que relacione la cantidad de pixeles con una distancia en metros. Para esto se seleccionan dos puntos del mapa que estén alejados entre sí, 1 metro.



Una vez calibrada la distancia en el mapa, se pueden empezar a ingresar puntos para generar las trayectorias deseadas. Como se mencionó anteriormente, el robot siempre se moverá a la velocidad seleccionada (no hay movimientos con aceleración) por lo tanto, solo existen dos tipos de trayectorias posibles:

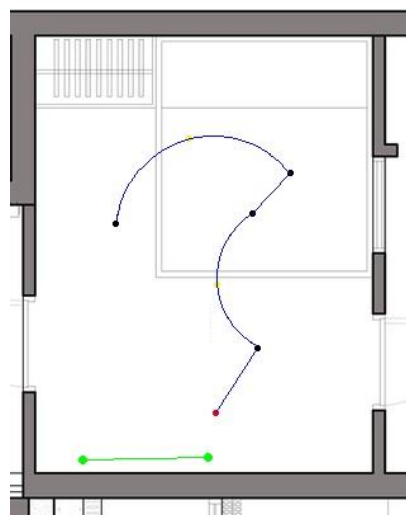
Trayectorias rectas: Para ingresar una trayectoria recta simplemente se debe hacer clic en el punto inicial y final de la trayectoria.

Trayectoria circular: Para ingresar una trayectoria circular, en medio de los puntos inicial y final, se debe ingresar un punto utilizando el clic medio del mouse, el cual se dibujara en amarillo. Esto es así ya que para poder definir una circunferencia se necesitan al menos, 3 puntos.



Al finalizar el recorrido deseado, se debe presionar la tecla “ESC” para cerrar el programa. Una vez hecho esto, se generará en el directorio del terminal un archivo llamado “output.pde”, el cual es compatible con la interfaz DuinoPack, además de una imagen llamada “trayectoria.jpg” que servirá para corroborar luego los movimientos del robot.

Imagen de salida:



Archivo de salida (más adelante se explica en detalle):

```
#include <DCMotor.h>

DCMotor motor0(M0_EN, M0_D0, M0_D1);
DCMotor motor1(M1_EN, M1_D0, M1_D1);

void setup()
{
  motor0.setClockwise(false);

  motor0.setSpeed( -16.19 );
  motor1.setSpeed( 16.19 );
  delay( 388 );
  motor0.setSpeed( 16.19 );
  motor1.setSpeed( 16.19 );
  delay( 6135 );

  motor0.setSpeed( 16.19 );
  motor1.setSpeed( -16.19 );
  delay( 1110 );
  motor0.setSpeed( 14.43 );
  motor1.setSpeed( 17.94 );
  delay( 12885 );

  motor0.setSpeed( 16.19 );
  motor1.setSpeed( -16.19 );
  delay( 163 );
  motor0.setSpeed( 16.19 );
  motor1.setSpeed( 16.19 );
  delay( 4430 );

  motor0.setSpeed( 16.19 );
  motor1.setSpeed( -16.19 );
  delay( 961 );
  motor0.setSpeed( 17.58 );
  motor1.setSpeed( 14.79 );
  delay( 18541 );

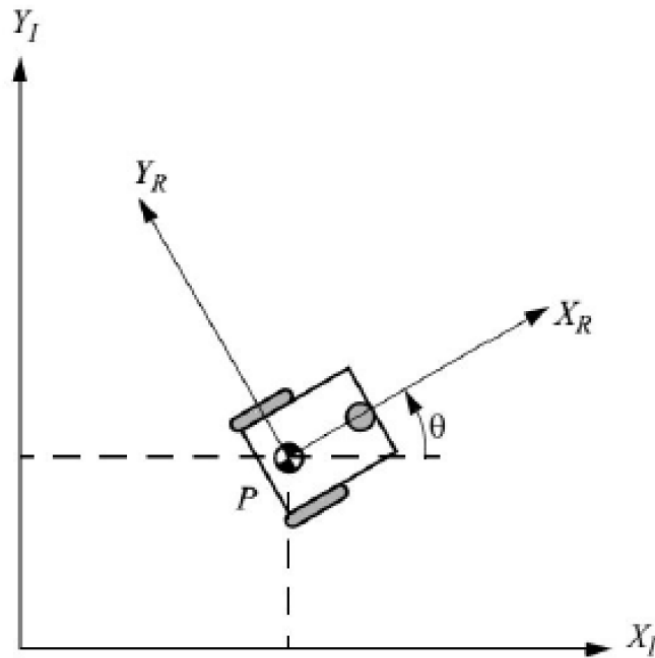
  motor0.brake();
  motor1.brake();
}

void loop()
{ }
```

Cinemática inversa del robot diferencial

A continuación se explicara brevemente la matemática involucrada en la cinemática inversa del robot diferencial. Recordemos que la cinemática inversa consiste en obtener la velocidad de cada rueda a partir de la trayectoria deseada, y en este caso, los cálculos son relativamente más complejos que en la cinemática directa, en donde ocurría exactamente lo contrario.

La primera hipótesis simplificadora será que la velocidad es siempre constante, y se medirá del punto medio entre las dos ruedas:



Existen solo dos trayectorias posibles partiendo de la hipótesis de velocidad constante, las trayectorias rectas, y las trayectorias que describen un círculo.

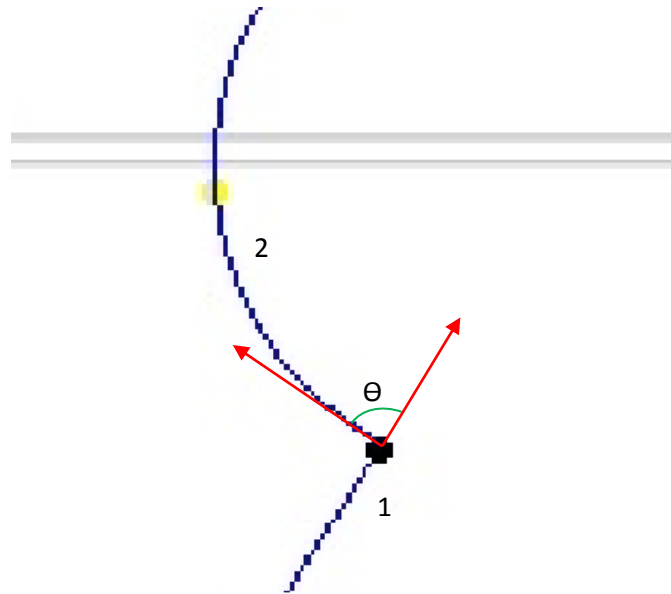
Otro parámetro que se debe tener en cuenta es el ángulo del robot, este no es un dato menor, ya que el ángulo con el que finaliza una trayectoria será diferente al ángulo con el que deberá iniciar la siguiente.

Habiendo dicho esto, se llega a que la cantidad de movimiento que podrá realizar el robot son 3:

- 1) Correcciones de ángulo: El robot gira sobre su propio eje para posicionarse en el ángulo correcto previo a iniciar la trayectoria deseada, este será el ángulo de salida de la trayectoria.
- 2) Trayectorias rectas: Se generan a partir del punto inicial y final, ambas ruedas giran a la misma velocidad, y el ángulo del robot se mantendrá constante.
- 3) Trayectorias curvas: Requieren de un punto intermedio, el robot describirá un arco de circunferencia, y el ángulo de salida será distinto al de llegada.

- 1) Correcciones de ángulo: Este es un movimiento rotatorio que tiene el fin de posicionar al robot en el ángulo de inicio de la siguiente trayectoria. Este movimiento se hará siempre antes de recorrer cualquier trayectoria, ya sea recta o curva.

Ejemplo:



Como se observa en la imagen anterior, a partir de los ángulos de finalización del recorrido 1, y el ángulo de inicio del recorrido 2, se puede calcular el ángulo de corrección, para posicionar el robot en el ángulo de inicio del recorrido 2.

Dependiendo del valor de cada ángulo, el código determina el sentido de giro más conveniente, es decir, horario o anti horario.

Sabiendo el ángulo y el sentido de rotación, se puede calcular el arco de circunferencia que se deberá recorrer. Utilizando los parámetros de velocidad introducidos cuando se ejecutó el código, la distancia entre ejes y el radio de las ruedas, se determina la velocidad y sentido de rotación de cada rueda.

- 2) Trayectorias rectas: Este es el tipo más simple de trayectoria ya que ambas ruedas giran a la misma velocidad y el mismo sentido.
Para generar una trayectoria recta se pasan como parámetro las coordenadas X, Y de los puntos inicial y final.
Usando las coordenadas, y el parámetro de correlación entre distancias en unidades de pixeles a metros, se calcula la distancia que deberá recorrer para alcanzar el punto final.
Utilizando la distancia, velocidad (elegida al principio por el usuario) y el radio de las ruedas se calcula la velocidad y el tiempo de giro.

El ángulo inicial es el mismo que el ángulo final, y se calcula sabiendo los desplazamiento en X e Y respectivamente.

- 3) Trayectorias curvas: Para poder obtener los parámetros necesarios para configurar el robot en una trayectoria de este tipo, primer se deben conocer los parámetros que definen la curva o arco de circunferencia, es decir, el ángulo total del arco y el radio.

Estos datos, que luego se usaran para setear los motores, deben obtenerse a partir de las coordenadas de 3 puntos que definen el arco de circunferencia deseado.

Para lograr esto, el código utiliza la siguiente ecuación que define una circunferencia:

$$0 = x^2 + y^2 + D * x + E * y + F$$

Resolviendo un sistema de 3 ecuaciones a partir de las 3 coordenadas, se pueden despejar las 3 incógnitas D, E, F. Se llega a lo siguiente:

$$D = - \frac{x_0^2 * (y_1 - y_2) - x_1^2 * (y_0 - y_2) + (x_2^2 + (y_0 - y_2) * (y_1 - y_2)) * (y_0 - y_1)}{x_0 * (y_1 - y_2) - x_1 * (y_0 - y_2) + x_2 * (y_0 - y_1)}$$

$$E = \frac{x_0^2 * (x_1 - x_2) - x_0 * (x_1^2 - x_2^2 + (y_1 + y_2) * (y_1 - y_2)) + x_1^2 * x_2 - x_1 * (x_2^2 - (y_0 + y_2) * (y_0 - y_2)) - x_2 * (y_0^2 - y_1^2)}{x_0 * (y_1 - y_2) - x_1 * (y_0 - y_2) + x_2 * (y_0 - y_1)}$$

$$F = - \frac{x_0^2 * (x_1 * y_2 - x_2 * y_1) - x_0 * (x_1^2 * y_2 - (x_2^2 - (y_1 - y_2) * y_2) * y_1) + (x_1^2 * x_2 - x_1 * (x_2^2 - (y_0 - y_2) * y_2)) - x_2 * (y_0 - y_1) * y_1 * y_0}{x_0 * (y_1 - y_2) - x_1 * (y_0 - y_2) + x_2 * (y_0 - y_1)}$$

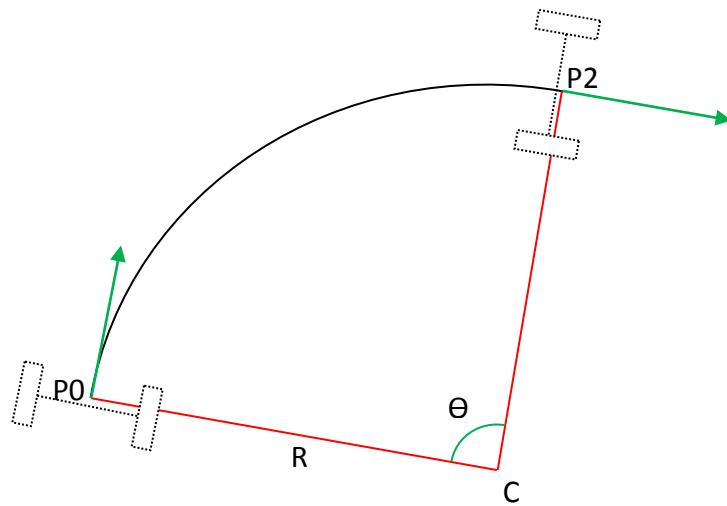
Donde (x0, y0), (x2, y2) corresponden a los puntos inicial y final respectivamente, y (x1, y1) al punto intermedio.

Utilizando las ecuaciones anteriores se pueden calcular los parámetros D, E, F, y a partir de estos, el radio y centro de la circunferencia.

$$C = (-\frac{D}{2}, -\frac{E}{2})$$

$$R = \sqrt{\frac{D^2}{4} + \frac{E^2}{4} - F}$$

Utilizando el radio, centro, y puntos inicial y final se obtiene el ángulo del arco de circunferencia que debe ser recorrido.



A partir del ángulo del arco, se calcula la longitud de arco. Sabiendo la longitud de arco, y la velocidad de desplazamiento ingresada por el usuario, se obtiene la velocidad y el tiempo para el punto medio entre las ruedas.

Finalmente, con la distancia entre ruedas y el radio de las mismas, se llega a la velocidad de cada motor.

El sentido de giro de cada rueda se utiliza utilizando como referencia el punto medio del arco, y los ángulos respectivos de los puntos inicial y final.

El ángulo de corrección se obtiene sabiendo los ángulos de P0 y P2, para esto se suma el desfase de 90° a cada ángulo.

Una vez obtenido todos los parámetros, se calcula el porcentaje de velocidad de cada motor, sabiendo que la velocidad máxima es de 200 r.p.m. para cada rueda.

Archivo de salidas – output.pde

A medida que se ingresan los distintos tramos de la trayectoria, el programa realiza escrituras en un archivo de salida, especificando velocidades y tiempos para cada motor, utilizando el formato de código compatible con el programa DuinoPack.

El formato es el siguiente:

Inicialización: Se incluye la librería para manejo de motores, y se asigna una salida a cada motor, siendo M0 el motor derecho y M1 el motor izquierdo.

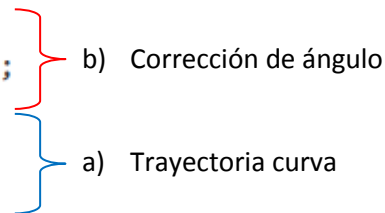
```
#include <DCMotor.h>

DCMotor motor0(M0_EN, M0_D0, M0_D1);
DCMotor motor1(M1_EN, M1_D0, M1_D1);

void setup()
{
  motor0.setClockwise(false);
```

Tramo de trayectoria: Todos los tramos se subdividen en dos bloques, el bloque a) corresponde a la corrección de ángulo previo a toda trayectoria, y el bloque b) corresponde a la trayectoria propiamente dicha. Como se observa en este ejemplo en el bloque b) las dos velocidades son distintas porque se trata de una trayectoria curva.

```
motor0.setSpeed( 16.19 );
motor1.setSpeed( -16.19 );
delay( 1110 );
motor0.setSpeed( 14.43 );
motor1.setSpeed( 17.94 );
delay( 12885 );
```



Fin del archivo: El archivo se finaliza con las directivas de detener los motores y un loop donde finaliza el programa:

```
motor0.brake();
motor1.brake();
}

void loop()
{ }
```

Archivo de salidas – trayectoria.jpg

Además del código listo para ser compilado con DuinoPack, el programa genera una imagen de salida en la que se registra la trayectoria punto a punto que recorrerá el robot utilizando dicho código de salida.

En la imagen de salida se puede ver:

- 1) La calibración de distancia de referencia (línea verde)
- 2) Punto de partida (punto rojo)
- 3) Todos los tramos recorridos (líneas azules)

