

Taller 2 Diseño de un robot diferencial

Marcela Sabogal Guerrero

Juan José Valencia Moreno

Juan Sebastian Espinosa Arciniegas

Electrical & Electronic Engineering Dept.



2 de abril de 2022

Introducción

En el presente trabajo se exponen los resultados del taller 2 en donde se buscó emplear el lenguaje de ROS para darle funcionalidad a un robot diseñado por nosotros. Para su realización fue necesario conocer las especificaciones mecánicas, eléctricas y de diseño que requería el robot. También profundizar nuestros conocimientos en raspberry para lograr controlar el robot mediante el sistema operativo de robots (ROS) en el sistema operativo Linux.

1. Diseño del robot

1.1. Lista materiales

Componente	Referencia
1. Raspberry	Raspberry pi 4
2. Arduino	Arduino mega 2560
3. Bases acrílicas	Laminas acrílicas de 3mm con corte laser
4. Rueda loca	Rueda universal de bola de acero W420
5. Separadores	Espaciadores/postes (diferentes tamaños)
6. 2 Pilas 3.7v	Bateria 18650 LiPo 3.7v 5800mAh
7. Jumpers	cables puente hembra-macho
8. Llantas	Lanta con rin para motores cuyo eje es de 3.8 mm
9. Puente H	Motor DC puente H L298 Arduino robótica
10. Motores	Motores con encoders
11. Regulador de voltaje	Modulo LM2596 regulador de voltaje DC-DC
12. Cable plug jack	Conecotor plug jack 2-1 DC 2.1mm
13. Adaptador tipo C	Conecotor USB tipo C

Cuadro 1: Materiales empleados en el diseño del robot



Figura 1: Materiales empleados en el diseño del robot

1.2. Plano mecánico

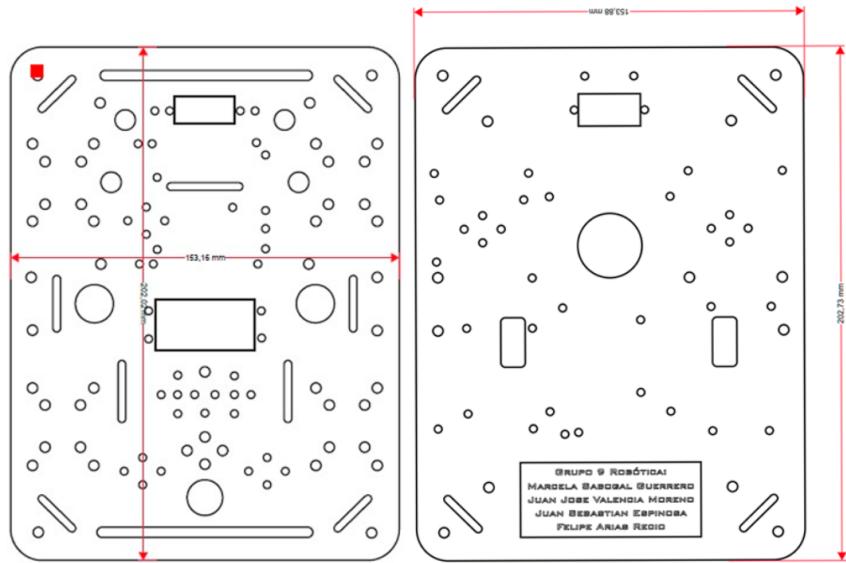


Figura 2: Placas acrílicas

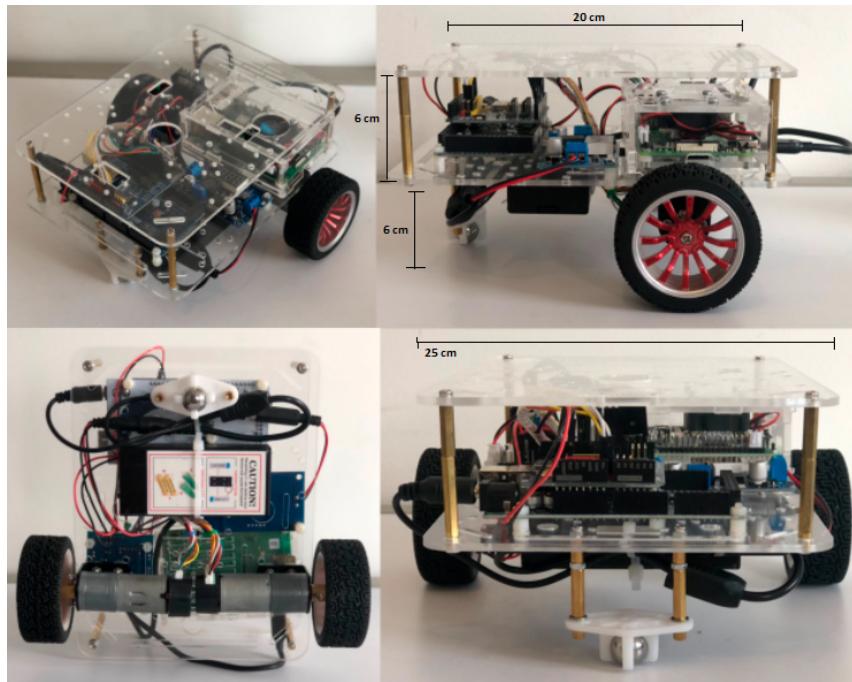


Figura 3: Placas acrílicas

1.3. Plano eléctrico/electrónico

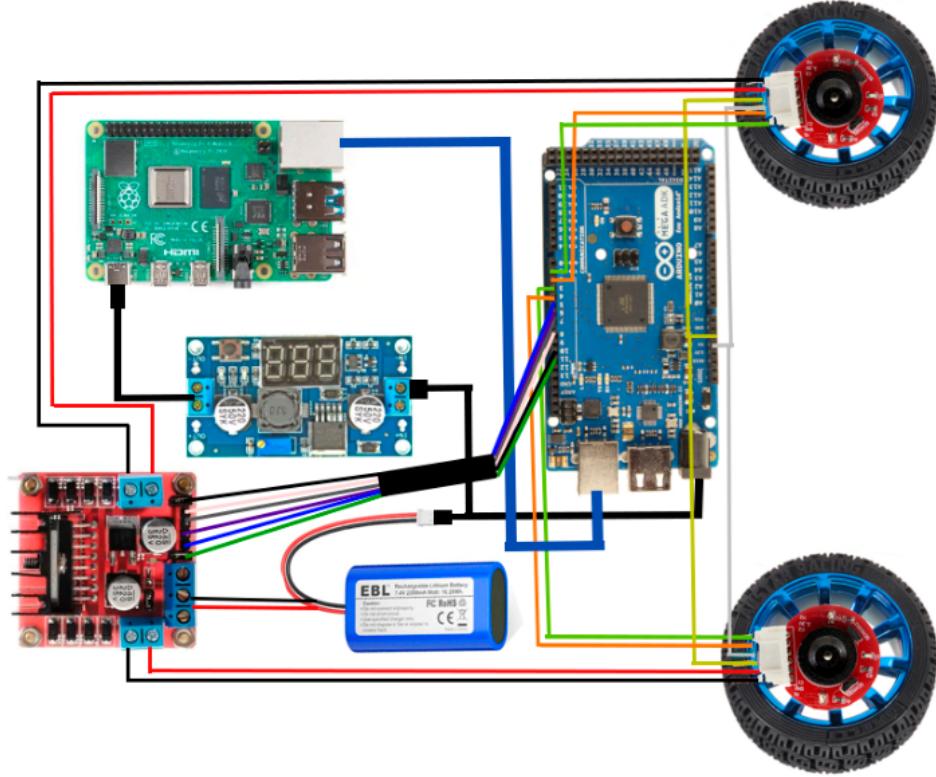


Figura 4: Plano eléctrico empleado

1.4. Descripción de funcionamiento del robot

Principalmente se requirió de una alimentación para la mayoría de componentes de 7,4v y 5800mAh los cuales fueron empleados en los componentes, encoders, puente h, raspberry y arduino. El robot emplea un puente h para permitir que un motor eléctrico DC pueda girar en ambos sentidos, avance y retroceso. Para llevar eso a cabo presenta 6 pines conectado a arduino, para el uso de dos motores. Los pines ENA, IN1, IN2 corresponden a las entradas para controlar el motor A y los pines ENB, IN3, IN4 permiten controlar el motor B. Los pines ENA y ENB funcionan para habilitar o deshabilitar sus respectivos motores, generalmente se utilizan para controlar la velocidad, ingresando una señal de PWM por estos pines.

En cuanto a raspberry este se comunica con arduino por medio de su entrada USB y para su alimentación emplea el puerto tipo C. Para esta alimentación fue necesario emplear un modulo regulador de voltaje de 7.4V a 5V, debido a las especificaciones que requería la alimentación del raspberry.

Por ultimo los motores usados son alimentados por el puente h y sus encoders por algunos pines de arduino, estois permiten generara impulsos, codificar o transductar de posición angular, es un dispositivo electromecánico. Su uso fue relevante puesto que proporcionaba mediciones y controles precisos sobre la velocidad del motor, la velocidad lineal, el posicionamiento angular o lineal.

En cuanto a la comunicación del robot, esta se da principalmente por parte de raspberry el cual tiene instalado el sistema operativo Ubuntu desktop. Por medio de este y el sistema operativo para robots (ROS) es posible tener la comunicación con los pines de arduino y a su vez con el punte de H enlazados al arduino. De esta manera comunicar las velocidades lineal y angular que usará el robot junto con las direcciones que este debe tomar.

1.5. Integración de la parte eléctrica y mecánica

En el diseño de nuestro robot buscamos que los componentes empleados tuvieran la mejor distribución/ localización optimizando el cableado y su estética, de esta manera facilitar el entendimiento de los puertos usados en cada sección de robot. Como se mencionó previamente la principal integración de la parte eléctrica con la mecánica se da al emplear los motores con sus encoders. En la Figura 4 se presenta la distribución de la alimentación que logró alimentar y enlazar todos los componentes implicados. La parte eléctrica nos permitió tener una comunicación y alimentación assertiva para dar cumplimiento con los requerimientos que tenía el robot. La integración de esta con la parte mecánica fue de suma relevancia para poder establecer correctamente la información que recibían tanto los encoders como el puente de H que logrará la dirección y alimentación específica que recibian los motores para ejecutar la velocidad lineal, angular y las direcciones a las que debí ir. La activación y dirección de los motores es controlada por el puente de H el cual invierte la polaridad de la conexión eléctrica del motor en cuestión, logrando de alguna manera intercambiar los cables que alimentan el dispositivo.

2. Solución

En este caso, similar al taller 1, se contaba con un nodo llamado *turtlebot_eleop.py* el cual al ejecutarse pide por consola valores de velocidad lineal y y angular, y estos valores enteros son los que se publican al tópico *locomotion_arduino* y proceden a manipularse en mediante las ecuaciones de odometria en arduino con el fin de relacionar el valor de velocidad ingresada por el usuario con un valor de PWM que es la manera en como arduino indica la velocidad (rpm) de cada motor.

En primera instancia, se encontró que la mínima velocidad a la que puede andar el robot con 25 (cm/s) que está relacionado con un ($PWM = 60$) y la máxima velocidad 60 (cm/s) relacionado con el ($PWM = 255$). Cabe destacar que para la obtención de los anteriores valores, se recurrió al *datasheet* de los motores en cuestión.

-
- Package Included: DC Motor with JST XH Connector 6pin(1.5mm) ;
 - Standard Operating Conditions--Rated Voltage: DC 9.0V
 - Performance Of Motors--No-load Speed: $11500 \pm 10\%$ rpm
 - Performance Of Gear Motor--Output Speed: $150 \pm 10\%$ rpm
 - Dimension--The Outside Shaft Length: 14.5mm

Figura 5: Datos motores.

Como se observa en la figura 5 estos motores bajo carga pueden ofrecer a lo más 150 rpm que para un diámetro de 6,5 cm de las ruedas que fueron usadas, se puede relacionar la velocidad lineal como sigue:

(1)

Se procedió de manera similar para obtener la velocidad angular en términos de las rpm de cada una de las ruedas; nuestro robot gira sobre el eje central de las dos ruedas, por lo anterior, la ecuación que describe esto es:

$$\omega_c = \frac{2\psi}{l} \quad (2)$$

Donde:

- ψ : es la velocidad de cada rueda y se tiene $\psi_r = -\psi_l$
- l : longitud entre las ruedas móviles.

Por ultimo, se realizó una caracterización de los motores para modelar de manera aproximada la relación entre *Velocidad lineal y PWM* que se realizó mediante la medición de rpms de cada motor haciendo uso de encoders; los resultados de la caracterización arrojaron una relación (cuasi-lineal)

en un rango de velocidades, (i.e., $[20 - 60]$ (cm/s)) y de igual manera para la velocidad angular del robot.

Con todo lo anterior, fue posible obtener un perfil de velocidades deseado según lo introducido por el usuario y con ello proceder a hacer los cálculos de odometría que se realizaron mediante:

$$x(t) = x_{ini} + \frac{1}{2} \int_0^t (V_r(t) - V_l(t)) \cos(\theta(t)) dt$$

$$y(t) = y_{ini} + \frac{1}{2} \int_0^t (V_r(t) - V_l(t)) \sin(\theta(t)) dt$$

$$\theta(t) = \theta_{ini} + \frac{1}{l} \int_0^t (V_r(t) - V_l(t)) dt$$



Figura 6: Conexión de los nodos `/turtle_bot_teleop` y `/serial_node` mediante el tópico `/loco-motion_arduino`

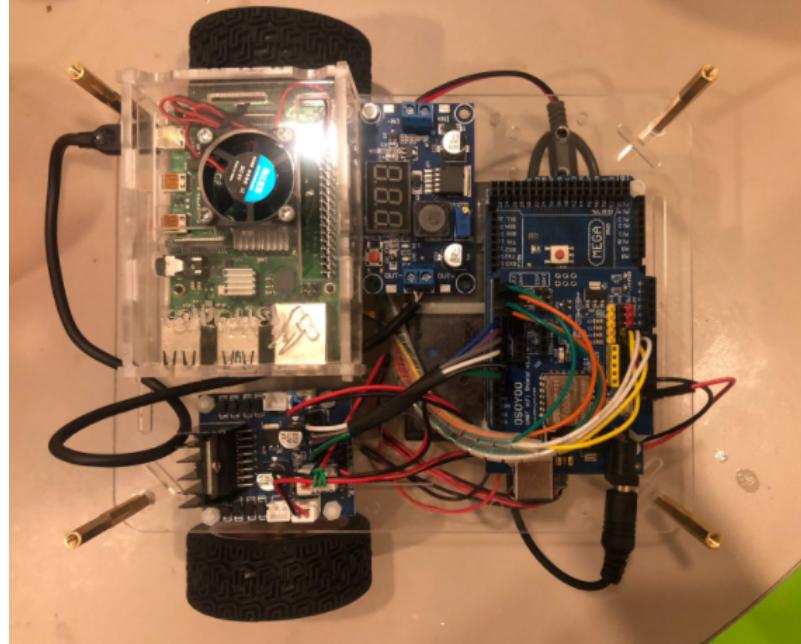


Figura 7: funcionamiento punto 1

3. Visualización de la posición en tiempo real del robot en interfaz gráfica

Para poder visualizar la posición en tiempo real del robot en tiempo real fue necesario realizar una interfaz en python usando la librería tkinter para visualizar el recorrido del robot y representar la posición del robot en el marco global de referencia en tiempo real y evidencie el camino recorrido por el mismo desde el inicio hasta el fin de su recorrido. Para ello se corrió el nodo *turtle_bot_9 turtle_bot_interface.py* fue importante otorgarle los permisos a este archivo. Luego se ejecuta el nodo escribiendo en la terminal *rosrun turtle_bot_9 turtle_bot_interface.py*, al tener ejecutado el *turtle_bot_teleop.py* y mover el robot, se observa como la interfaz replica en cada instante de tiempo los movimientos de 1 robot y permite guardarlos en una pestaña emergente en donde se especifica la ruta de guardado.

En la Figura 8 se presentan las conexiones usadas y el respectivo tópico que las conecta corroborando que existe esta conexión la cual permite enviar información principalmente de la interfaz a **Arduino** mediante el topico *locomotion_arduino* y poder presentar gráficamente el recorrido del robot para posteriormente guardarlo en una ruta específica.

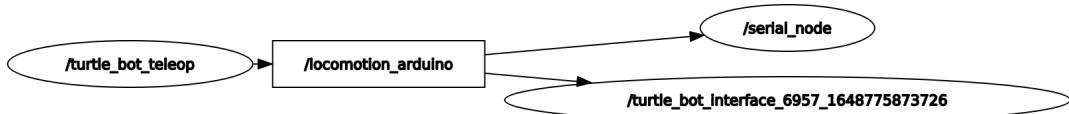


Figura 8: Conexión de los nodos */turtle_bot_teleop*, */serial_node* y */turtle_bot_interface_6957_1648775873726* mediante el tópico */locomotion_arduino*

4. Complementación de los nodos */turtle_bot_teleop* y el nodo */turtle_bot_interface*

Para que al iniciar el nodo se le pregunta al usuario en la interfaz si desea guardar el recorrido del robot en necesario ejecutar *turtle_bot_teleop.py*, se realizan los movimientos del robot y luego se presiona la tecla ESC para abrir la interfaz y guardar los movimientos realizados por el robot. Este archivo será de tipo .txt y se guardará en la siguiente ruta */home/robotica/talleres_robotica/src/turtle_bot_9 /scripts/* para nuestro caso.

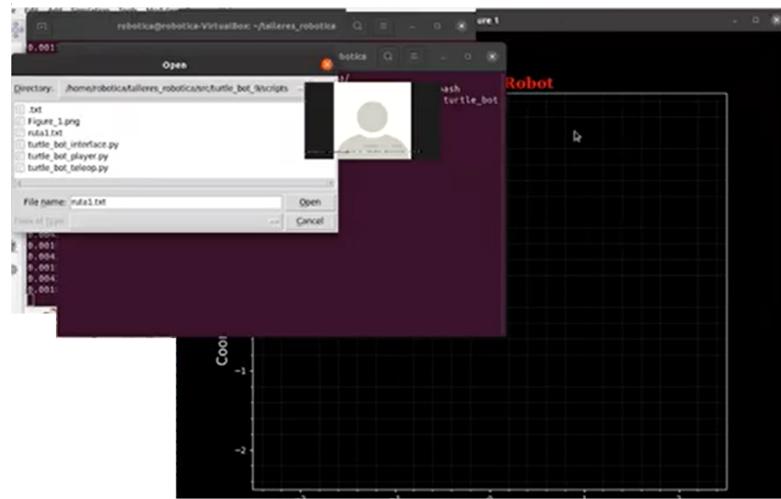


Figura 9: Guardado de recorrido del robot con ruta especificada

En la Figura 9 se puede observar la ventana emergente para guardar el recorrido y especificar la ruta.

5. Creación del nodo */turtle_bot_player* para reproducir la secuencia de acciones del robot

Para crear el nodo */turtle_bot_player* debe proveer el servicio de recibir el nombre de la partida a reproducir es necesario ejecutar nuevamente el roscore, tenerlo abierto y reiniciar el escenario del taller 1 en CoppeliaSim. se accede al workspace *talleres_robotica/* y se otorgan los respectivos permisos en el nodo *turtle_bot_player.py*. En la ventana emergente, dar click en el botón Open y seleccionar el archivo .txt que fue guardado en el tercer punto, de esta manera se creara el recorrido en CoppeliaSim.

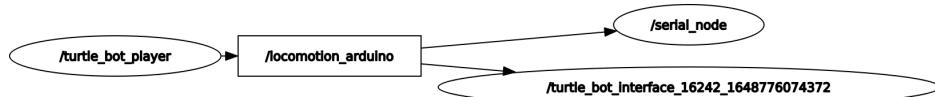


Figura 10: Conexión de los nodos */turtlebot_player*, */serial_node* y */turtlebot_interface_16242_1648776074372* mediante el t */locomotion_arduino*

En la Figura 11 se presenta el recorrido el robot dado un archivo .txt del cual obtiene todos los movimientos que fueron guardados en el punto 3. en cuanto a la Figura 10 se observan las conexiones

realizadas de los nodos mediante los tópicos para que se pueda correr la escena guardada en el punto 3.

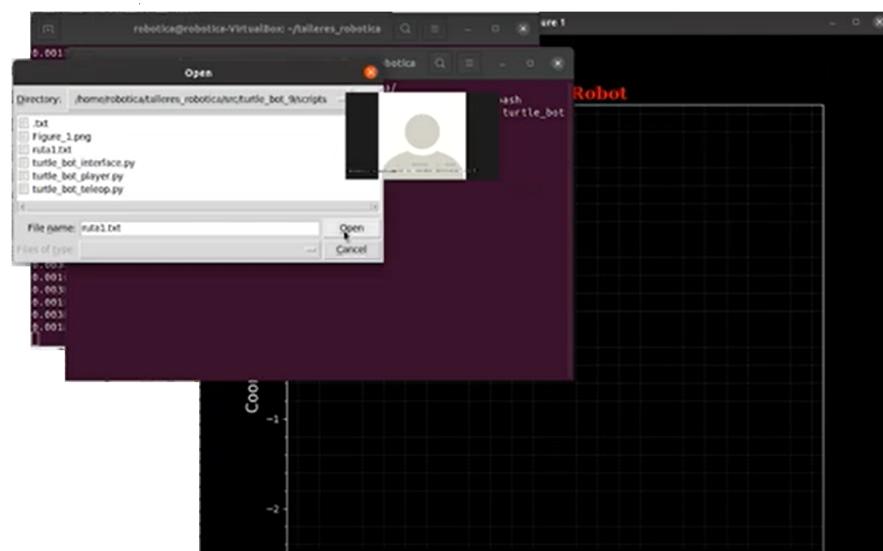


Figura 11: Recorrido dado un archivo .txt

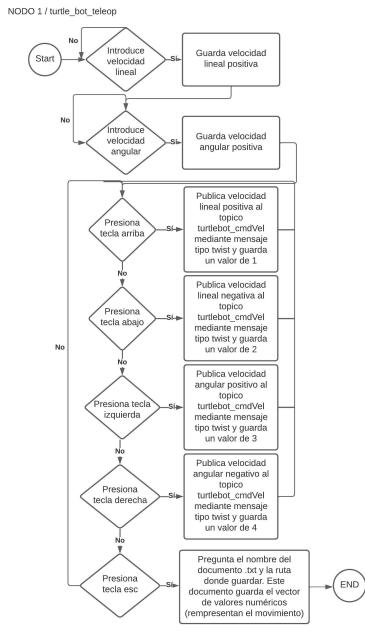


Figura 12: Nodo: Turtle_bot_teleop

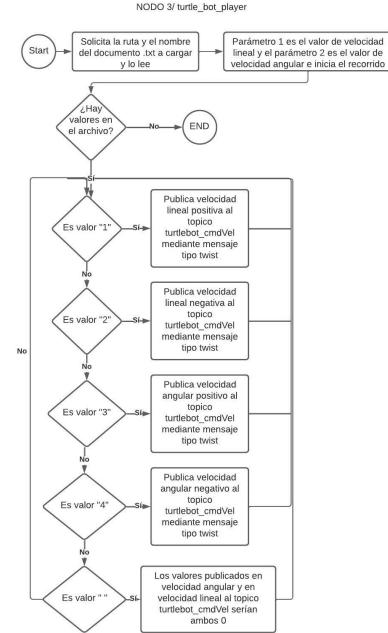


Figura 14: Nodo: Turtle_bot_player

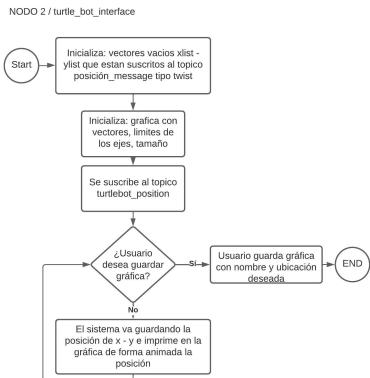


Figura 13: Nodo: Turtle_bot_interface

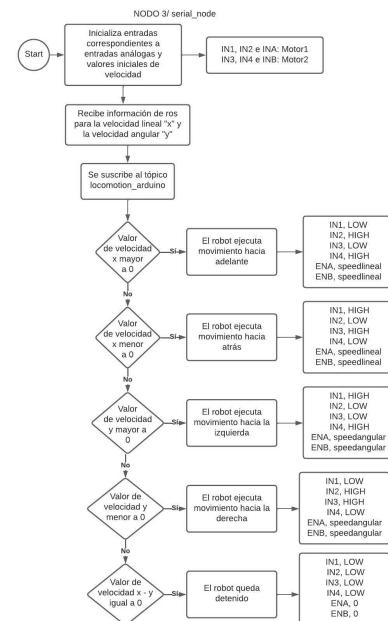


Figura 15: Nodo: Serial_node