UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, implementación y control de un vehículo de tres ruedas omnidireccionales

Trabajo Fin de Máster

Máster en Ingeniería Mecatrónica

Alumno: Domingo Lacasa Collado

Tutor: Vicente Fermín Casanova Calvo

Resumen

En este proyecto, con el objetivo de realizar el Trabajo Fin de Máster, se desarrollará un vehículo de tres ruedas omnidireccionales.

Gracias al uso de este tipo de ruedas, se obtendrá un grado de libertad extra, por lo que se podrán realizar movimientos imposibles que otros vehículos no podrían realizar. Para ello se utilizará una disposición triangular de las mismas, cuya combinación de movimientos individuales permitirá el movimiento omnidireccional del vehículo.

Para las distintas pruebas se utilizará el software Matlab Simulink, con el que se comunicará vía Bluetooth al vehículo las velocidades que deben llevar las ruedas, y con el que se realizará la monitorización del conjunto completo.

Para el control del vehículo se utilizará un microcontrolador Arduino, que recibirá las órdenes y gestionará las ruedas mediante un control de velocidad.

Índice

Índice de figuras

1. Introducción

Desde hace unos años, es cada vez más común el uso de robots en la industria, ya sea para realizar tareas que serían difíciles para humanos, o simplemente para incrementar la eficiencia. Para ello se ha indagado en aspectos como por ejemplo el ambiente de trabajo, el tipo de tracción deseado o el medio de movilidad a utilizar.

Para mejorar los robots e incrementar el número o tipo de trabajos a realizar por los mismos, se ha buscado incrementar notablemente la maniobrabilidad y reducir el espacio necesario para llevar a cabo sus determinados movimientos. Para ello se han estudiado distintos tipos y configuraciones de ruedas. Es aquí donde entran en juego las ruedas omnidireccionales, que permiten cualquier tipo de desplazamiento a lo largo del plano de trabajo, evitando además realizar giros y desplazamientos innecesarios.

Un robot omnidireccional, es un tipo de robot móvil cuya configuración le permite desplazarse en cualquier dirección sin la necesidad de alcanzar previamente una orientación específica. Es decir, es capaz de realizar traslaciones (hacia adelante, en reversa, laterales) o rotaciones, a partir de cualquier posición en que se encuentre.

Para un robot omnidireccional es imperativo contar con al menos tres ruedas. Por ello, un punto importante del diseño es la decisión de cuántas ruedas emplear. Cada una de ellas proporciona al robot una fuerza normal al eje del motor y paralela a la superficie sobre la cual se desplaza. La suma de éstas permite la traslación y rotación de la estructura. Por lo general, presentan una configuración mecánica de tres o cuatro ruedas.

Los robots de cuatro ruedas tienen más tracción que los de tres, pues se adiciona la potencia entregada por el motor adicional. Lo cual se traduce a menos deslizamiento en las ruedas si el peso de la carga se encuentra distribuido uniformemente sobre ellas, un mayor consumo de energía, costo y posiblemente, la necesidad de incorporar un sistema de suspensión para distribuir las fuerzas sobre las ruedas. La configuración de tres ruedas es mecánicamente más simple que la de cuatro. No obstante, este último permite una mayor aceleración al robot.

Existen varios modelos de ruedas, en función del tamaño, número de rodillos, o disposición de los mismos, por ejemplo:

Ilustración 1: Tipos de ruedas

En este proyecto se estudiarán las ruedas omnidireccionales, en concreto las de la marca “Vex”. Estas ruedas disponen de una serie de rodillos en su parte exterior, que además del movimiento perpendicular al eje que nos permiten las ruedas convencionales, posibilitan el movimiento en la misma dirección del eje de rotación de la rueda.

Ilustración 2: Ruedas Vex

Esto se debe a que los rodillos presentan buena tracción con el suelo cuando la rueda está girando, pero además de eso, cuando se produce una fuerza transversal al eje de los rodillos, éstos giran con bastante facilidad, permitiendo un deslizamiento al mismo tiempo que mantienen la estabilidad del vehículo.

Teniendo en cuenta todo lo mencionado anteriormente, se ha vuelto posible la construcción de robot móviles que disponen de una mayor libertad de movimientos, son más compactos gracias a que necesitan menos mecanismos para una mayor variedad de movimientos, y permiten que espacio de trabajo se menor gracias a la facilidad que aportan a la hora de maniobrar.

2. Planteamiento del problema inicial

Se desea realizar el diseño y montaje de un vehículo de tres ruedas omnidireccionales. Para ello será necesario la simulación del conjunto para comprobar que el estudio de la cinemática sea correcto.

Posteriormente debe realizarse el montaje físico del mismo, para trasladar los ensayos realizados mediante simulaciones al mundo real. Para ello, se realizará un control de velocidad mediante un microcontrolador, conectado mediante comunicación inalámbrica al sistema encargado de la supervisión.

El vehículo deberá ser capaz de seguir las trayectorias preprogramadas, monitorizando la trayectoria deseada y la trayectoria estimada del mismo, así como el sentido de giro y velocidad de las distintas ruedas.

Finalmente, deberá poderse realizar un control manual, pudiéndose indicar las coordenadas a las que debe moverse el robot en su siguiente movimiento mediante el uso de un gamepad conectado al PC.

3. Estudio teórico

Gracias al uso de las ruedas omnidireccionales, es posible el movimiento del vehículo en varias direcciones sin necesidad de que haya rotación en el mismo. Son capaces de moverse de forma perpendicular a su eje de rotación (tal como las ruedas convencionales) ya que la posición de los rodillos permite una buena tracción con el suelo. Pero además de este tipo de movimiento, permiten un desplazamiento de forma perpendicular a la propia rueda, gracias al deslizamiento que proporcionan los rodillos en este caso.

Para poder realizar los dos tipos de movimientos mencionados anteriormente, será necesaria una disposición particular de las ruedas en el vehículo. Además del movimiento general producido por la rotación del eje del motor, para el movimiento perpendicular a una rueda en particular será necesario un "empuje" producido por el resto de ruedas. Para ello, la situación correcta de las ruedas será la siguiente:

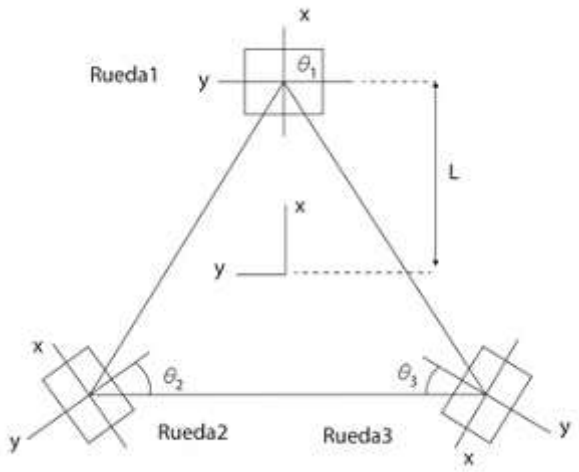


Ilustración 3: Disposición de ruedas

Con esta disposición, el vehículo será capaz de desplazarse a lo largo de los ejes X e Y, y rotar sobre sí mismo en ambos sentidos, además de realizar distintas combinaciones entre estos tipos de movimientos. Para poder llevar a cabo lo anterior, será necesario que las ruedas puedan ser gobernadas de forma individual, adaptando su velocidad al caso específico en el que se encuentren.

A continuación se adjuntan los giros que deben hacer las ruedas para posibilitar todos los tipos de desplazamiento posibles.

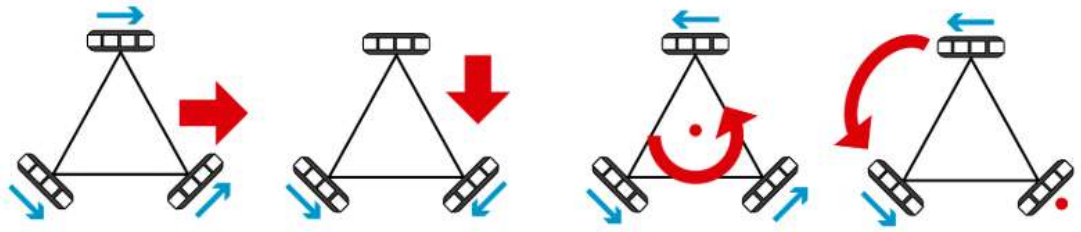


Ilustración 4: Tipos de movimiento

Es importante puntualizar que las ruedas deben moverse a una determinada velocidad para realizar cada movimiento, siendo en ocasiones de valor diferente entre ellas.

Para ello, se procede al estudio que relaciona las velocidades generales del robot y lineales en el plano cartesiano, con las velocidades angulares independientes para cada rueda.

El primer paso es realizar un esquema del robot, situando los distintos ejes de coordenadas y demás datos de interés.

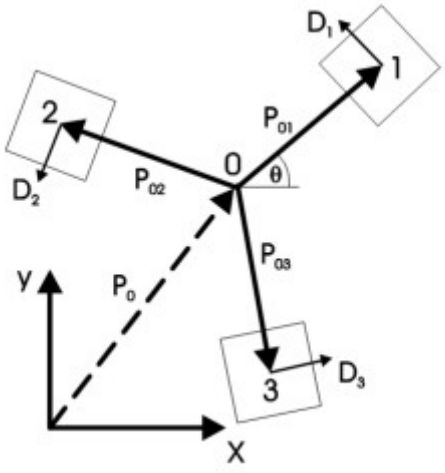


Ilustración 5: Esquema vehículo

Donde O es el centro de masa del robot, y θ el ángulo de giro medido de forma antihoraria entre el eje X y el eje que va desde O hasta la primera rueda.

Se introducirán primero las variables que se utilizarán en las ecuaciones:

θ	Ángulo en tiempo real
ϕ	Velocidad angular rueda
$\dot{x}$	Velocidad lineal en eje x
$\dot{y}$	Velocidad lineal en eje y
$\dot{\theta}$	Velocidad angular de la rueda
L	Longitud desde O hasta la rueda
r	Radio de la rueda

Sabiendo esto y teniendo en cuenta que utilizamos el vector $\overrightarrow{P_{01}}$ como eje $\overrightarrow{OX}$, podemos obtener la matriz de rotación.

$$R(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$$

A partir de esta matriz, considerando que L es la distancia desde O hasta la rueda, y que los vectores D indican la dirección de avance de su correspondiente rueda:

$$P_{01} = L\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad P_{02} = R\left(\frac{2\pi}{3}\right)P_{01} = \frac{L}{2}\begin{pmatrix} -1 \\ \sqrt{3} \end{pmatrix}$$

$$P_{03} = R\left(\frac{4\pi}{3}\right)P_{01} = \frac{L}{2}\begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}$$

$$D_i = \frac{1}{L}R\left(\frac{\pi}{2}\right)P_{0i}$$

$$D_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad D_2 = \frac{1}{2}\begin{pmatrix} \sqrt{3} \\ 1 \end{pmatrix} \quad D_3 = \frac{1}{2}\begin{pmatrix} \sqrt{3} \\ -1 \end{pmatrix}$$

El vector Po es la posición del centro de masa representado en la figura anterior, a partir de esto, la posición y velocidad de cada rueda será:

$$r_i = P_0 + R(\theta)P_{0i}$$

$$\mathrm{v}_i = \dot{P}_0 + \dot{R}(\theta)P_{oi}$$

Mientras que la velocidad de cada rueda en función de su correspondiente vector D será:

$$v_i = \mathrm{v}_i^T(R(\theta)D_i)$$

Sustituyendo en la ecuación anterior:

$$v_i = \dot{P}_0^T R(\theta)D_i + P_{oi}^T \dot{R}^T(\theta)R(\theta)D_i$$

Se simplifica que el segundo término es la velocidad tangencial del robot:

$$P_{oi}^{T}\dot{R}^{T}(\theta)\,R(\theta)\,D_i = L\dot{\theta}$$

Realizando los cálculos correspondientes, se obtiene la velocidad angular de las ruedas, que se divide por el radio *r* para obtener las velocidades lineales de las ruedas:

$$\begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{pmatrix} = \frac{1}{r}\begin{pmatrix} -\sin(\theta) & \cos(\theta) & L \\ -\sin(\frac{\pi}{3}-\theta) & -\cos(\frac{\pi}{3}-\theta) & L \\ \sin(\frac{\pi}{3}+\theta) & -\cos(\frac{\pi}{3}+\theta) & L \end{pmatrix}\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix}$$

Ilustración 6: Ecuaciones de la cinemática

Gracias a estas ecuaciones, podrán obtenerse las velocidades de las ruedas de forma independiente, a partir de las velocidades lineales que debe llevar el centro de gravedad del vehículo en cada instante.

4. Vehículo simulado

Para poder realizar la simulación, será necesario el diseño de una serie de piezas extra para llevar a cabo el montaje simulado del prototipo.

El software que se utilizará para dicha tarea será Solidworks, mediante el cual se diseñarán las distintas piezas necesarias para su uso en la simulación y su posterior montaje físico.

Para la simulación por ordenador del modelo, usaremos el software Simulink de Matlab. Para ello se creará un entorno en el que posteriormente se incluirán las distintas partes del vehículo y sobre el que se ensayarán distintos tipos de movimientos.

4.1 Diseño

4.1.1 Chasis

Para poder realizar el montaje de los motores en disposición triangular, tal como se ha visto en el apartado de estudio teórico, será necesario tener una serie de factores en cuenta a la hora de hacer el diseño.

El chasis tendrá forma de prisma triangular, con una serie de orificios para facilitar el montaje de los motores. Del mismo modo, se recortarán las esquinas mediante chaflanes para evitar que haya rozamiento entre el chasis y las ruedas en movimiento. Del mismo modo, necesitará una altura suficiente para incluir el equipo electrónico.

El resultado será el siguiente:

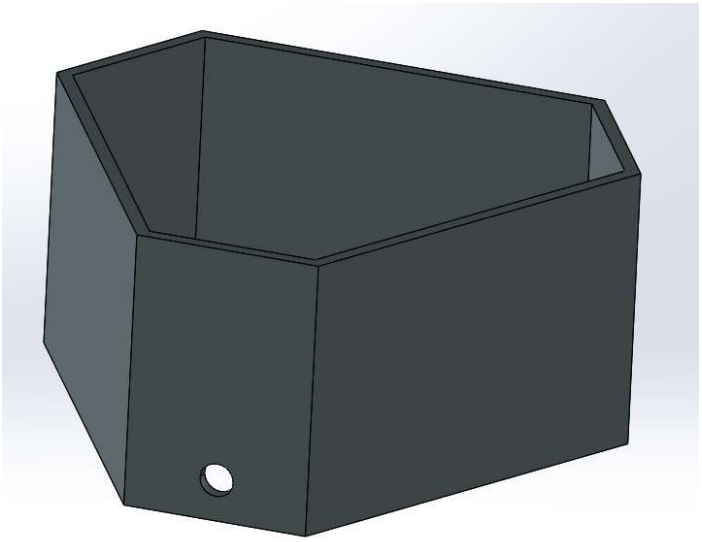


Ilustración 7: Chasis vista en ángulo

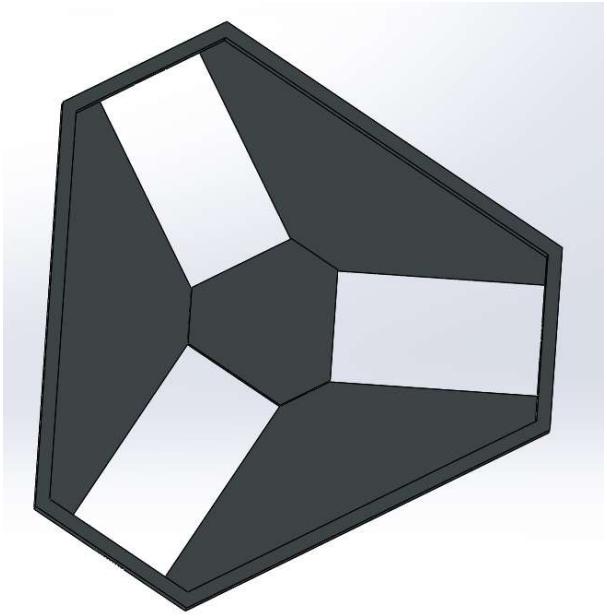


Ilustración 8: Chasis vista superior

4.1.2 Ruedas

Para las ruedas se usará el modelo comercial Vex EDR, por lo que obtendremos los diseños necesarios a partir de la página web del distribuidor. [2]

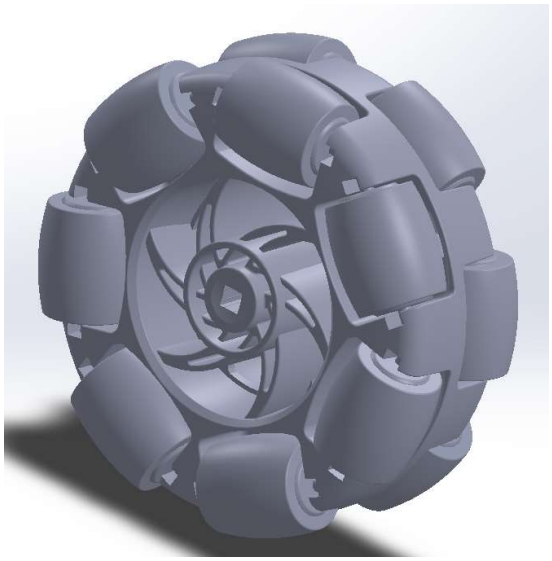


Ilustración 9: Modelo rueda

4.1.3 Motores

Del mismo modo, como los motores utilizados para mover las ruedas serán los 37d mm Metal Gearmotor, se usarán los modelos proporcionados por el fabricante para las simulaciones correspondientes.

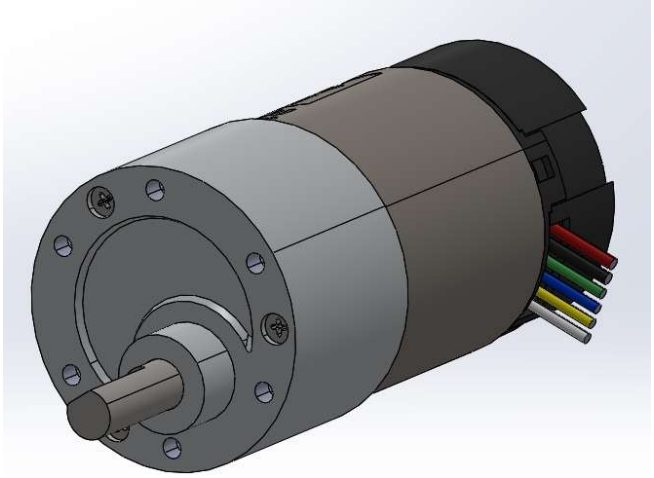


Ilustración 10: Modelo motor

4.1.4 Acople

Finalmente, al tener el motor un eje circular, y la rueda un hueco para eje de forma cuadrada, será necesario diseñar un acople para la correcta conexión entre ambos.

El acople estará conformado por dos piezas distintas. La primera de ellas tendrá una forma cilíndrica, con una parte hueca interior a lo largo de su eje, en la que se introducirá el eje del motor para su correcta sujeción.

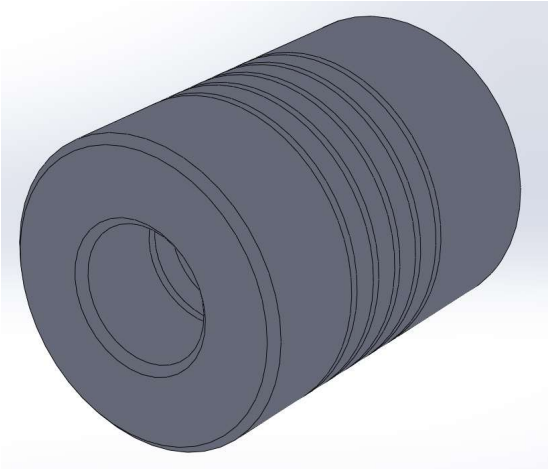


Ilustración 11: Modelo Acople Pieza 1

La segunda pieza tendrá una forma prismática. Una parte será también cilíndrica, para que pueda introducirse en la pieza anterior, del mismo modo que con el motor. La otra mitad de la pieza, sin embargo, será un prisma de base cuadrada que se introducirá en la sección hueca de la rueda.

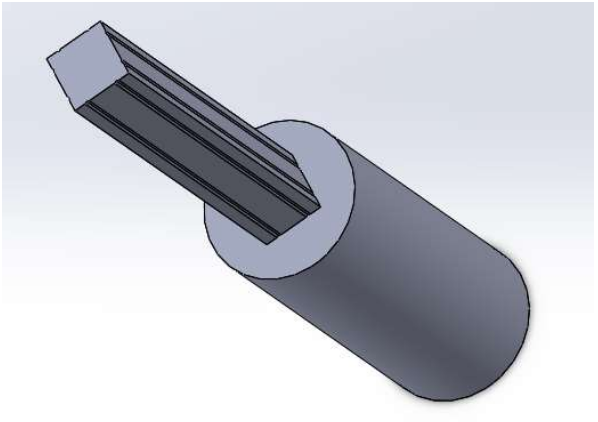


Ilustración 12: Modelo Acople Pieza 2

Con ambas piezas diseñadas correctamente, la rueda girará solidaria al eje y sin deslizamientos indeseados.

4.2 Simulación

Para la simulación por ordenador del modelo, se usará el software Simulink de Matlab. Para ello se creará un entorno en el que posteriormente se incluirán las distintas partes del vehículo y sobre el que se ensayarán distintos tipos de movimientos.

4.2.1 Simulación del vehículo por partes

El primer paso en la simulación es crear un espacio de trabajo e ir anexionando las distintas piezas del vehículo.

4.2.1.1 Entorno

Esta parte estará conformada por una superficie rectangular que hará de suelo, en el que se configurará la gravedad para la simulación. Además se incluirán los distintos grados de libertad deseados para el vehículo, para ello se definirá una articulación prismática entre el suelo y el chasis para habilitar la traslación en los ejes “x” e “y”, y una articulación de revolución para la rotación en el eje “z”.

4.2.1.2 Chasis

Esta pieza es la principal para el ensamblaje del vehículo. En ella se definirán una serie de ejes secundarios que servirán como nexo para el posterior ensamblaje con las ruedas y motores. Además, se ha incluido una esfera de color verde para marcar cuál es la rueda delantera, y dos de color rojo para las traseras.

4.2.1.3 Ruedas

Las ruedas estarán conformadas por dos piezas distintas: el buje y los rodillos.

Del mismo modo que en el chasis, es necesario añadir ejes secundarios al buje para el montaje. Los distintos rodillos se conectarán mediante articulaciones de revolución, para posibilitar que el vehículo pueda deslizarse en la misma dirección que el eje de las ruedas. A los rodillos de las ruedas, se les añadirá una serie de esferas que simularán la zona del rodillo sobre la que hay rozamiento con la superficie. El resultado será el siguiente:

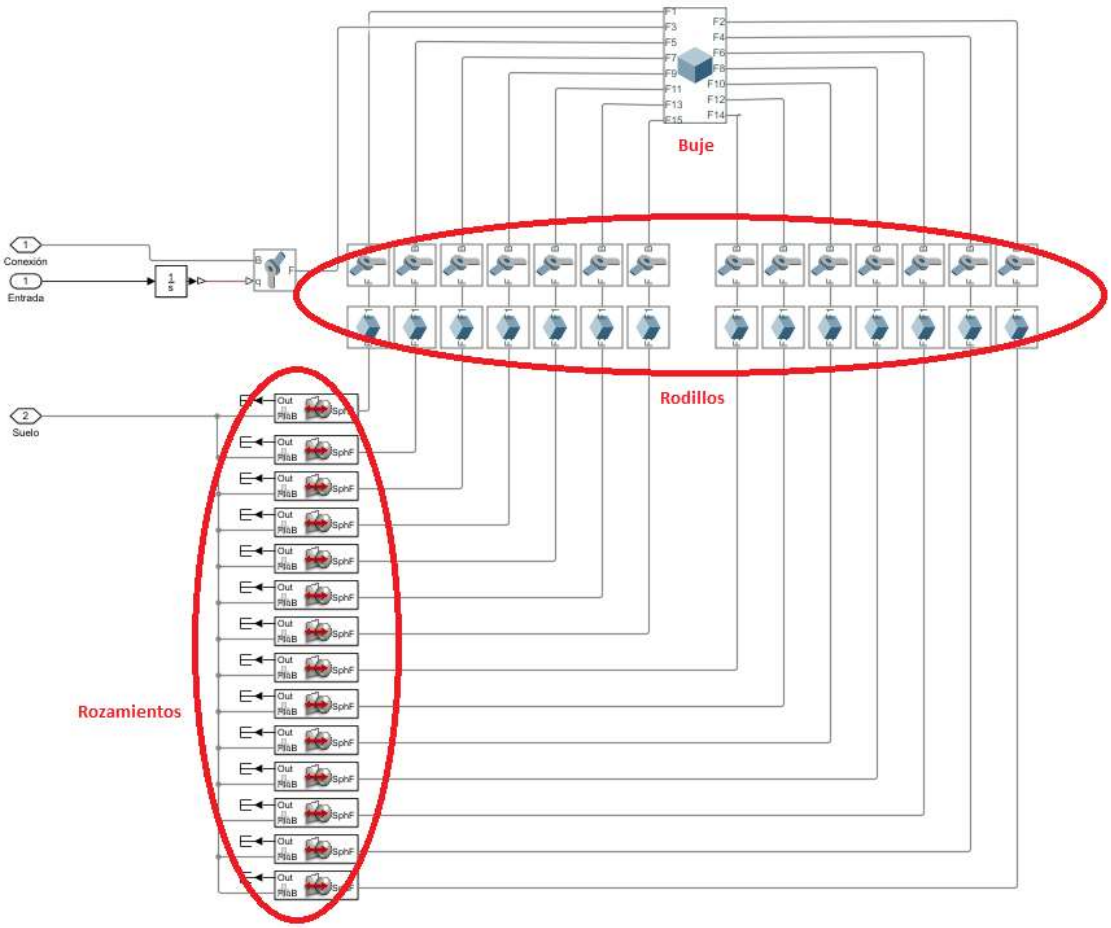


Ilustración 13: Esquema rueda simulación

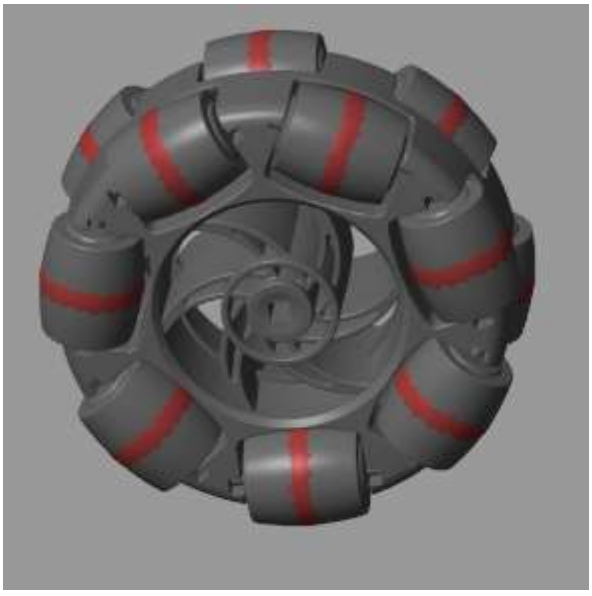


Ilustración 14: Superficies con rozamiento

Al simular la rueda en movimiento, puede observarse la trayectoria que describiría al no estar sujeta al resto del vehículo, puede observarse que se desvía debido al rozamiento con el suelo.

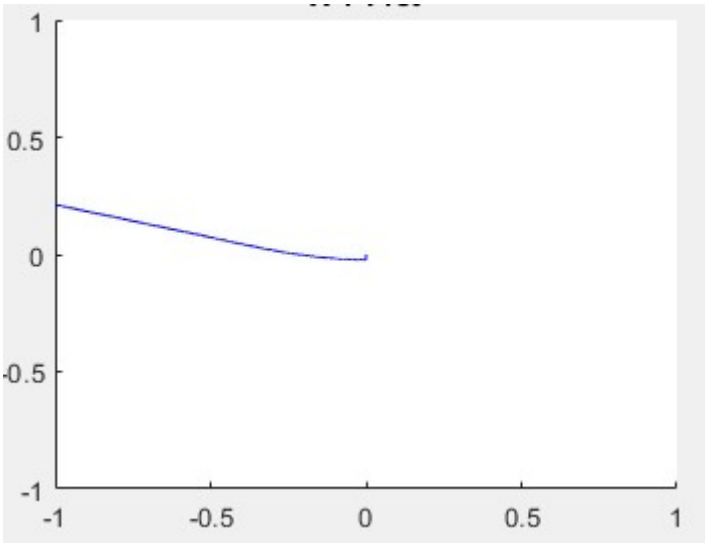


Ilustración 15: Movimiento descrito rueda plano XY

Por otra parte, para la conexión de la rueda completa al chasis se usará otra articulación de revolución, permitiendo así que la rueda gire sobre el eje del motor.

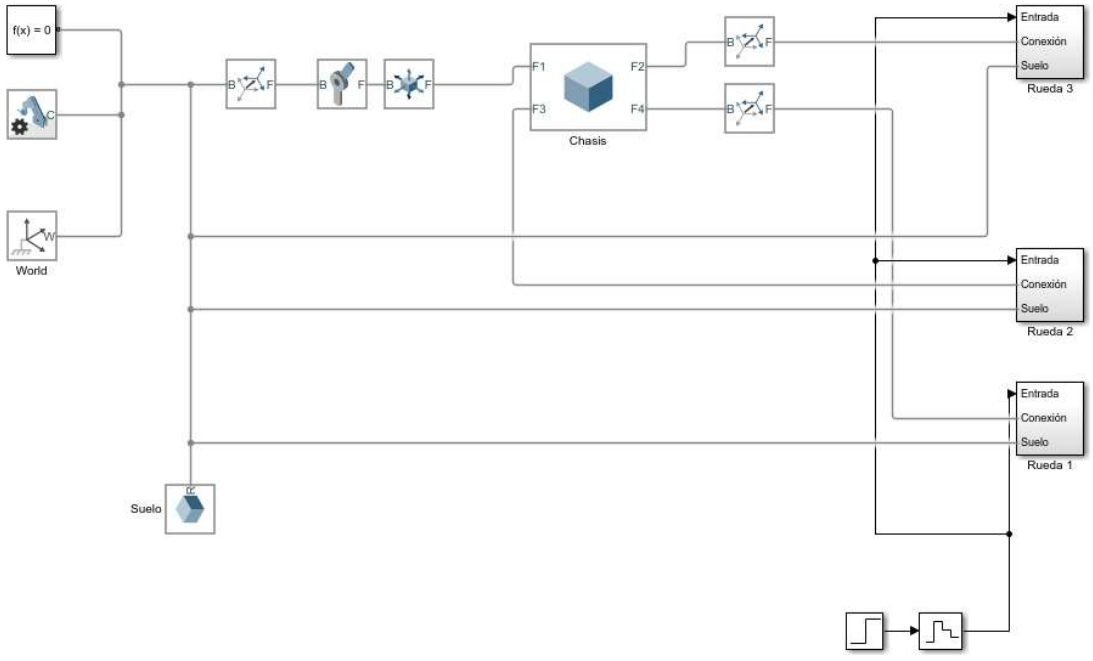


Ilustración 16: Esquema simulación vehículo completo

La representación del vehículo completo dentro de su entorno es la siguiente:

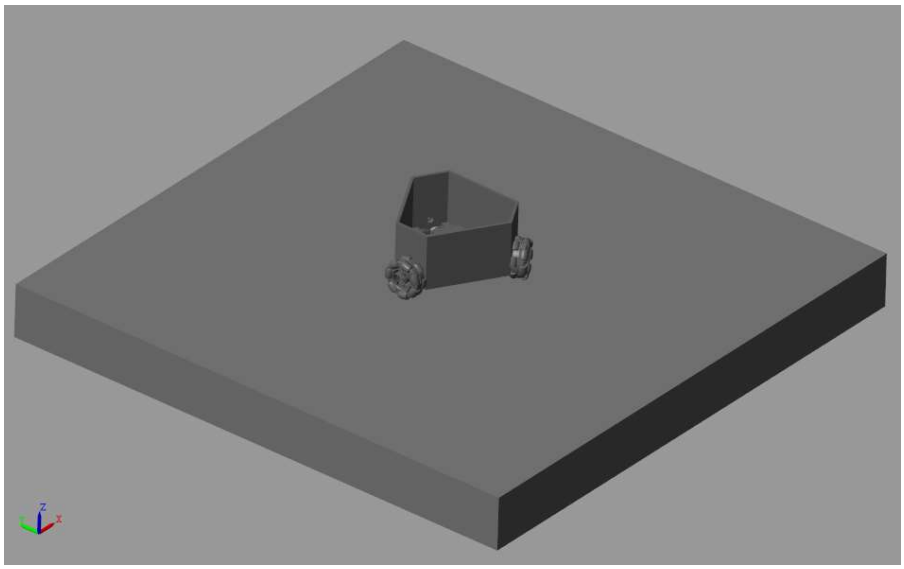


Ilustración 17: Vehículo completo en entorno

4.2.2 Simulación de movimiento simple

Una vez se ha comprobado que el montaje es correcto de forma visual, se realizará una prueba con un movimiento de rotación para comprobar que las restricciones están bien definidas y el vehículo responde tal y como se desea.

Para ello se comprobará que al recibir las tres ruedas una referencia positiva (de características idénticas para todas ellas), éstas se mueven en el mismo sentido y a la misma velocidad, produciendo así que el vehículo gire sobre sí mismo, sin desplazarse de forma vertical ni horizontal.

Al constatar que el movimiento es correcto, se podrá empezar a simular distintos tipos más complejos de movimientos de forma aislada, que serán utilizados posteriormente.

4.2.3 Simulación de movimiento horizontal y vertical

Este tipo de movimiento es más complejo que el realizado anteriormente, ya que se actuará sobre cada rueda de forma independiente.

Para realizar el movimiento horizontal, la rueda delantera deberá girar hacia la dirección a la que queremos desplazarnos al 100% de la velocidad deseada, mientras que las otras dos se moverán en la misma dirección, pero a un 50% de la velocidad de la otra rueda.

En el caso del movimiento vertical, la rueda delantera permanecerá inmóvil, mientras que las dos ruedas traseras giraran al 100% de la velocidad deseada y con sentido contrario entre ellas (en función del sentido, el vehículo se moverá de forma vertical hacia arriba o hacia abajo).

4.2.4 Simulación de trayectorias

Una vez comprobados todos los tipos de movimientos por separado (movimiento rotatorio, vertical y horizontal), se procederá a realizar una trayectoria usando todos ellos, comprobando así la fidelidad a la hora de seguir una ruta establecida.

Para la trayectoria se realizará primero un movimiento vertical tanto en sentido ascendente como en descendente y posteriormente se realizará algo similar pero para movimientos de carácter horizontal. Para la simulación se le indicará directamente a las ruedas la velocidad a la que deben ir, posteriormente, en el caso físico, sí que se obtendrán las velocidades a partir de una trayectoria preprogramada.

El resultado, fijándonos en las coordenadas X e Y del centro de masas del vehículo es el siguiente:

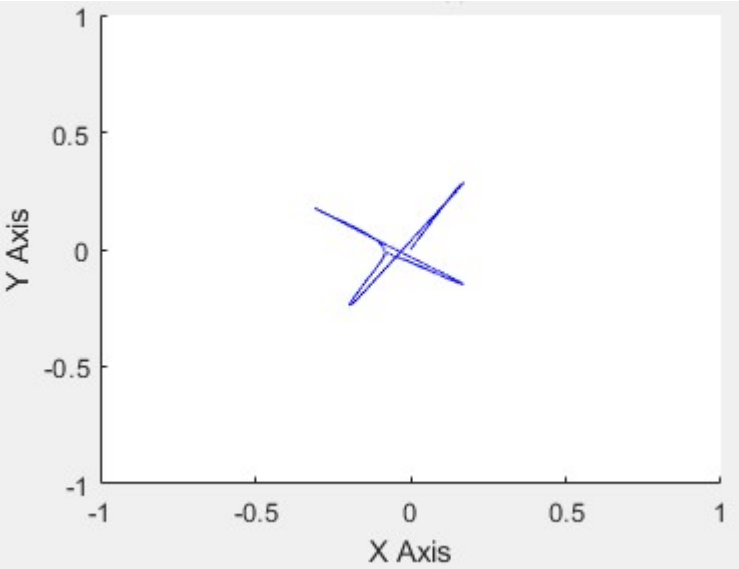


Ilustración 18: Trayectoria plano XY

Como puede observarse, la realización de la trayectoria es bastante buena, sin embargo hay algunos casos en los que el vehículo se desvía. Para saber por qué puede estar produciéndose esto, será necesario fijarse en las velocidades de las ruedas y del centro de masas:

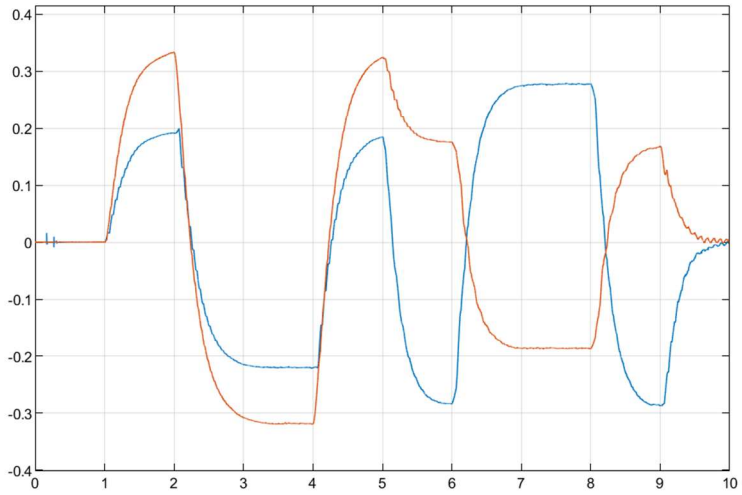


Ilustración 19: Velocidades lineales centro de masas

Donde la línea marrón es la velocidad lineal en el eje Y, y la línea azul es la del eje X del centro de masas del vehículo.

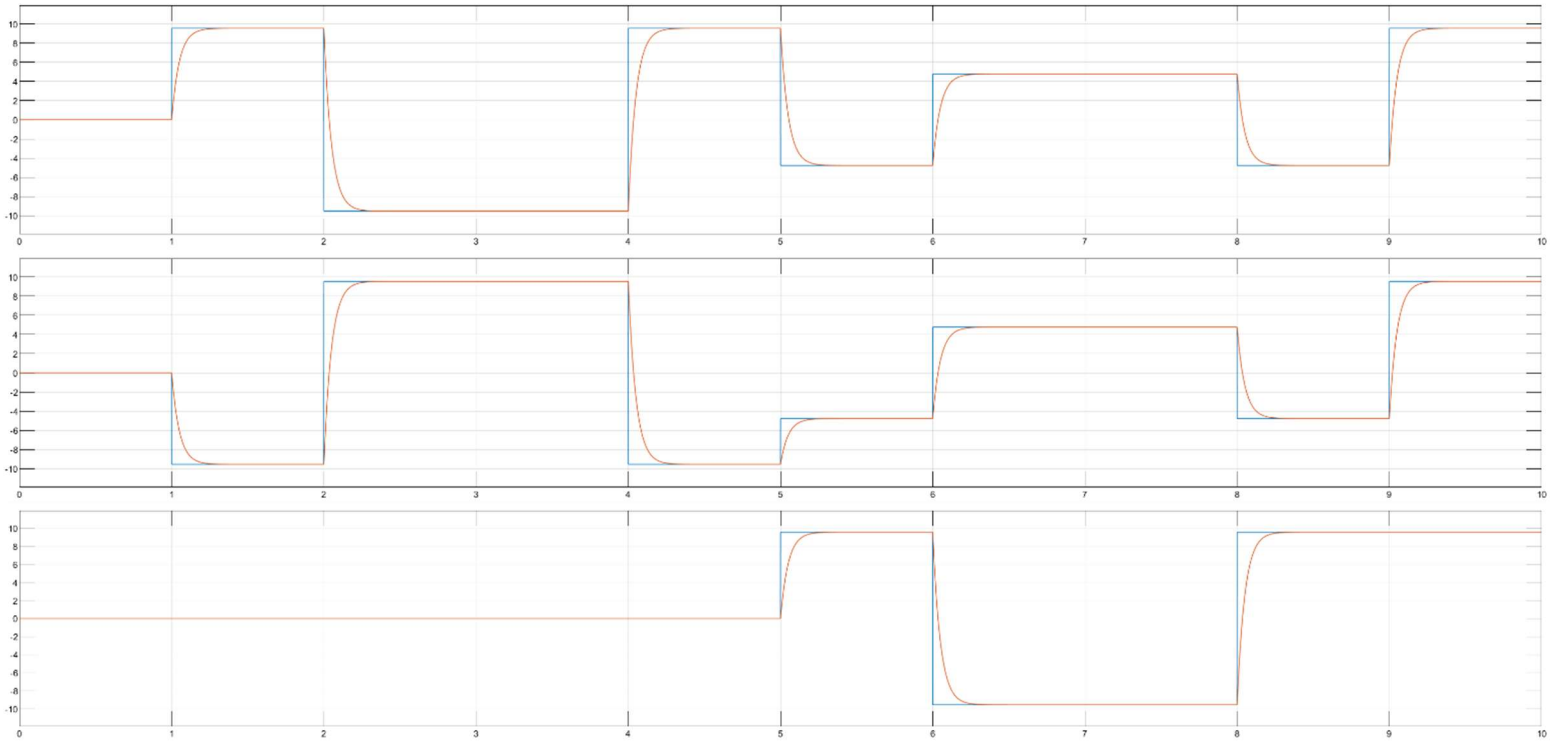


Ilustración 20: Velocidades ruedas

En las figuras anteriores puede verse que las esquinas de la trayectoria coinciden con los cambios bruscos de la velocidad (coinciden con los segundos 2, 4 ,6 y 8), por lo que se produce un deslizamiento con el suelo. Otra posibilidad, sería que durante el pequeño transitorio de los motores se produjera un giro indeseado, alterando así el resto de la trayectoria. Fijándonos en este detalle, se tendrá en cuenta que para las siguientes trayectorias deberán suavizarse los cambios de velocidad en la medida de lo posible.

4.2.5 Simulación de movimiento manual

El siguiente paso será introducir un método para poder modificar en tiempo real el movimiento del robot. Para ello se introducirán tres potenciómetros, uno para la velocidad y sentido del movimiento vertical, otro para velocidad y sentido del movimiento horizontal y otro para velocidad y sentido de rotación.

Además, las señales procedentes de éstos se gestionarán conjuntamente, permitiendo así movimientos compuestos más complejos, a partir de la suma y/o resta de las distintas señales.

Ilustración 21: Diales Modo Manual

5. Vehículo físico

5.1 Componentes a utilizar

A continuación se mostrarán los distintos elementos que se utilizarán para el montaje físico del vehículo.

5.1.1 Componentes electrónicos

5.1.1.1 Microcontrolador Arduino Due

El microcontrolador Arduino Due está basado en la CPU Atmel SAM3X8E ARM Cortex-M3. Tiene 54 pines de entrada/salida (12 de los cuales pueden ser usados como salidas PWM), 12 entradas analógicas, 4 UARTs (puertos serie de hardware), un reloj de 84 MHz, una conexión USB OTG, 2 DAC (convertidores digital a analógico), 2 TWI, una conexión de alimentación, un botón de reset y un botón de borrado. [1]

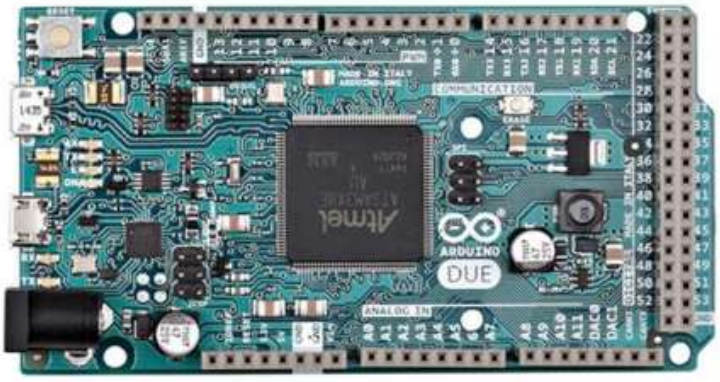


Ilustración 22: Microcontrolador Arduino

5.1.1.2 Módulo Bluetooth HC-06

Módulo compatible con Arduino, capaz de realizar la comunicación entre éste y otro dispositivo, como puede ser un PC, un dispositivo móvil o una tablet.

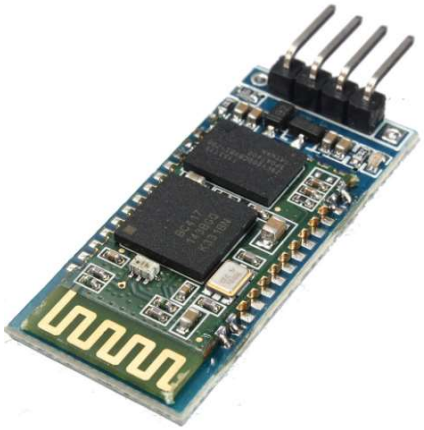


Ilustración 23: Módulo Bluetooth

5.1.1.3 Motor Shield V1

Para poder controlar los motores, es necesario usar un complemento extra montando junto al Arduino. El Motor Shield V1 Adafruit permite alimentar con corriente suficiente los 3 motores a utilizar, así como gobernarlos de forma precisa.

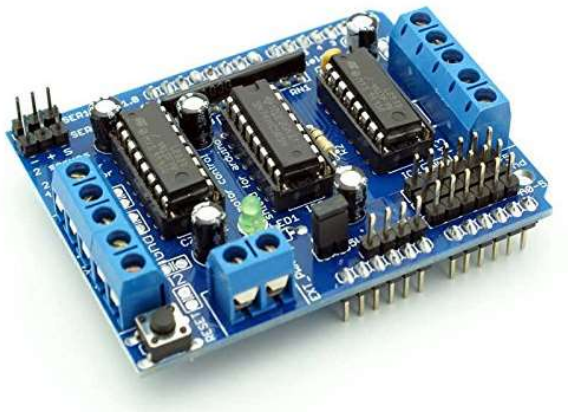


Ilustración 24: Motor Shield V1

5.1.2Componentes mecánicos

5.1.2.1 Pololu 37d mm Metal Gearmotor

Motor reductor de corriente continua con escobillas de 12V. Incluye un encoder cuya resolución es de 64 cuentas por movimiento del eje. Dicho eje tiene una forma de “D” a la salida.

Ilustración 25: Motor Pololu

5.1.2.2 Rueda omnidireccional Vex EDR

Ruedas omnidireccionales que además de los movimientos convencionales, permiten un desplazamiento lateral rápido debido a los rodillos de los que dispone en la parte exterior.

Ilustración 26: Rueda VEX EDR

5.1.2.3 Acoplador flexible 5mm - 8mm 18x25mm CNC Stepper Motor Shaft

Acoplador flexible ajustable al eje del motor, mediante una serie de tornillos puede actuar como una mordaza sobre el eje del motor, permitiendo el giro solidario entre ambos y evitando deslizamientos indeseados.

Ilustración 27: Acoplador

5.2 Montaje

Una vez comprobado que las simulaciones responden como deseamos, se procederá al montaje físico del vehículo. El esquema de montaje es el siguiente:

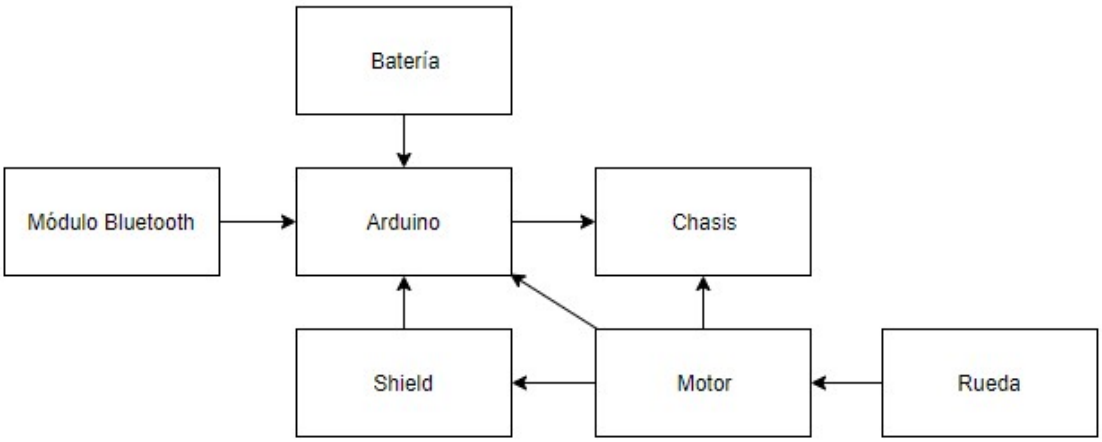


Ilustración 28: Diagrama montaje

Para ello, se exportará el chasis usado en el apartado de simulación para que sea creado mediante una impresora 3D. Para una mayor facilidad a la hora de imprimir, se ha dividido en dos partes que posteriormente se unirán. Una vez impreso, se usará un taladro para realizar una serie de perforaciones destinadas al correcto acople de los motores.

Con los motores fijados mediante tornillería, se procederá a conectar los motores al shield y al Arduino. Es importante fijarse en el código de colores en los cables del motor, detallado a continuación:

Color	Function
Red	motor power (connects to one motor terminal)
Black	motor power (connects to the other motor terminal)
Green	encoder GND
Blue	encoder Vcc (3.5 – 20 V)
Yellow	encoder A output
White	encoder B output

Ilustración 29: Código colores motor

Fijándonos en la tabla anterior, debemos conectar los cables rojo y negro al shield para la alimentación del motor, los cables verde y azul a la alimentación, y los cables amarillo y blanco a los pines del Arduino que definiremos posteriormente para la medición del encoder.

Para las pruebas iniciales, se incluirán también 3 potenciómetros conectados a los conversores DAC del Arduino, que serán de gran utilidad a la hora de implementar el control PI de velocidad.

Por otra parte, se sujetarán las ruedas a los motores mediante el uso de los acoples mencionados en apartados anteriores.

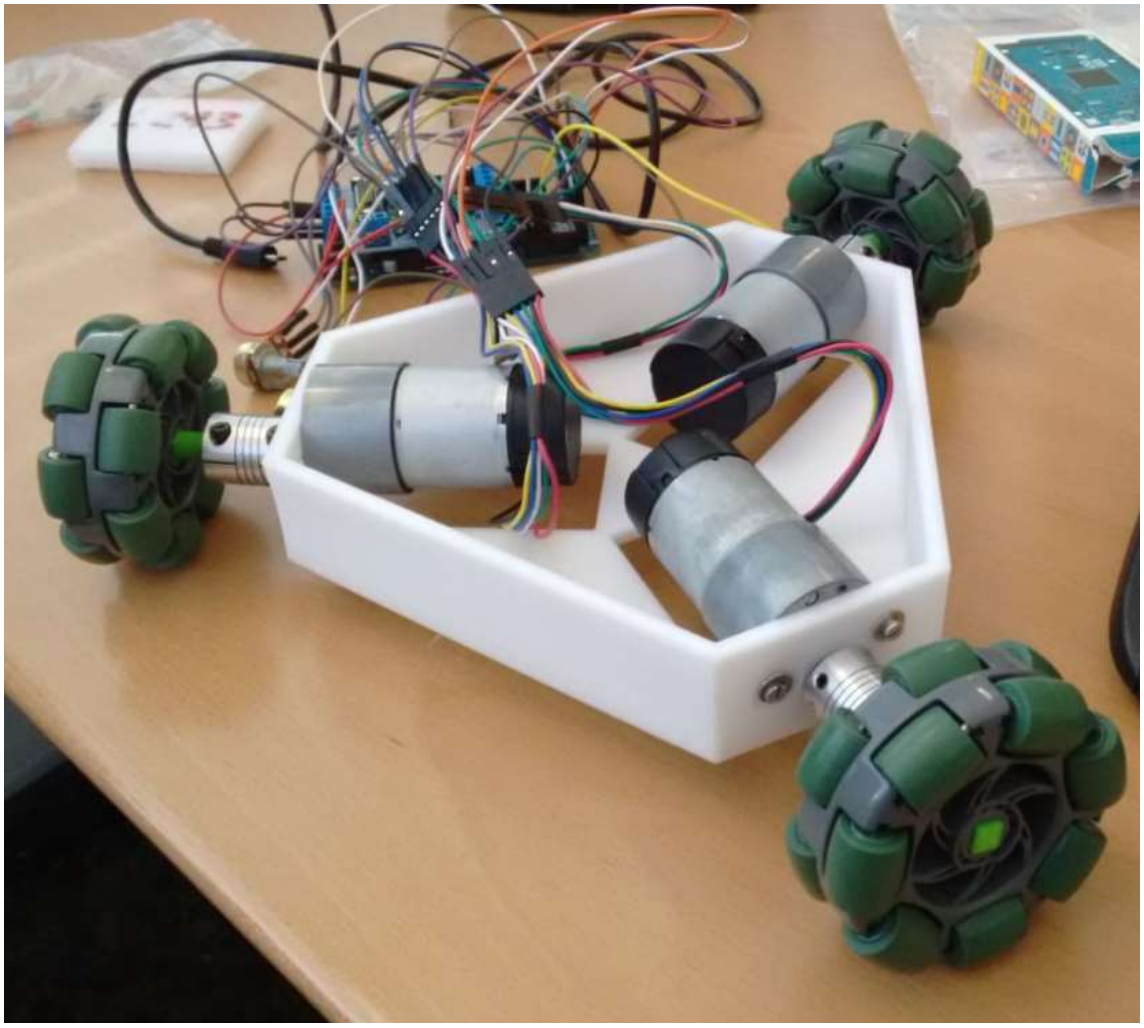


Ilustración 30: Montaje a medias

Finalmente se incluirá un módulo bluetooth, con el que se realizará la comunicación inalámbrica entre el vehículo y el PC, que se encargará de indicar la trayectoria y monitorizar las velocidades del mismo.

Una vez todo montado, se incluirá la tapa y el resultado es el siguiente:

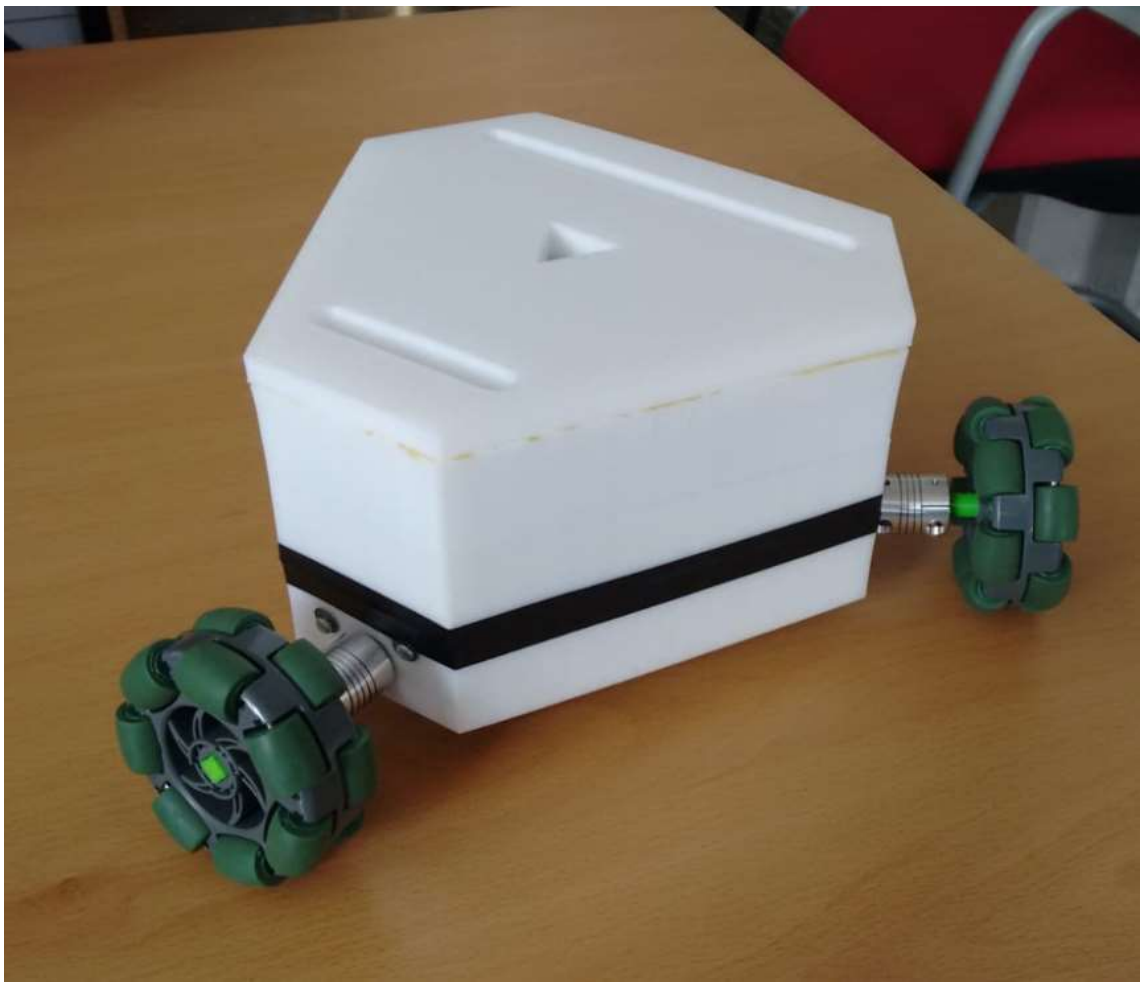


Ilustración 31: Montaje completo

6. Diseño del control, pruebas y correcciones

6.1 Pruebas iniciales

Para comprobar el correcto funcionamiento de los distintos componentes, se realizarán una serie de pruebas, empezando por pequeñas comprobaciones aisladas, sobre las que después se irá desarrollando el programa principal.

El primer paso será testear la comunicación, para ello se diseñará un programa muy simple usando un dispositivo Android. Dicho programa contará con dos joysticks, que informarán de su posición cartesiana. A partir del mensaje generado, el Arduino debe ser capaz de aislar cada dato de forma correcta, gestionarlo y mostrarlo por pantalla. En caso de no recibir ningún dato, se emitirá un mensaje de error. Se adjunta el código correspondiente en la sección de Anexos.

Por otra parte, para comprobar que los motores están bien conectados, se realizará un programa que convierta el valor medido mediante una serie de potenciómetros en valores estándar de acción de control. Estos valores se enviarán a los motores, que se moverán más o menos deprisa, en función de éste. Además, se mostrará por pantalla el valor de velocidad medido por los encoders. Código disponible en el apartado de Anexos.

Destacar que aunque estemos midiendo la velocidad, no es importante que para una misma acción de control los motores se muevan igual, ya que de esto se encargará el sistema de control (esto respecto a la velocidad, sí es importante que el sentido de giro sea el mismo).

6.2 Control

Para que el vehículo funcione de forma correcta, intervendrán varios tipos de control, que funcionarán tanto de forma independiente como interconectados entre ellos para el correcto desarrollo de la aplicación. Una parte del control se realizará en el propio vehículo mediante el uso de Arduino, y la otra parte se encontrará en el PC, estando ambos conectados gracias a una comunicación Bluetooth.

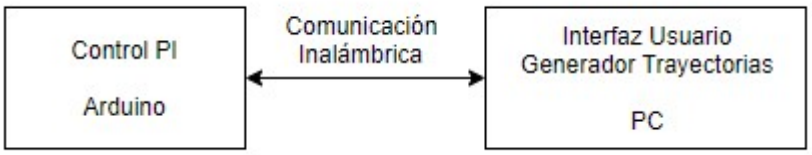


Ilustración 32: Esquema general de control

6.2.1 Control de Velocidad en Arduino

En este tipo de vehículos es muy importante que las ruedas se muevan siempre a la velocidad deseada, ya que cuanto más difiera la real de la deseada peor será el seguimiento de la trayectoria. Además, por la disposición triangular de las ruedas, es de vital importancia que una rueda no vaya más o menos deprisa de lo deseado, ya que podría significar una rotación del vehículo, incrementando de manera significativa el error.

El control adecuado para este sistema será un controlador PI, con el que conseguiremos una respuesta rápida por parte de los motores, nulificando además el error y garantizando la precisión en la velocidad a la que se moverá el motor.

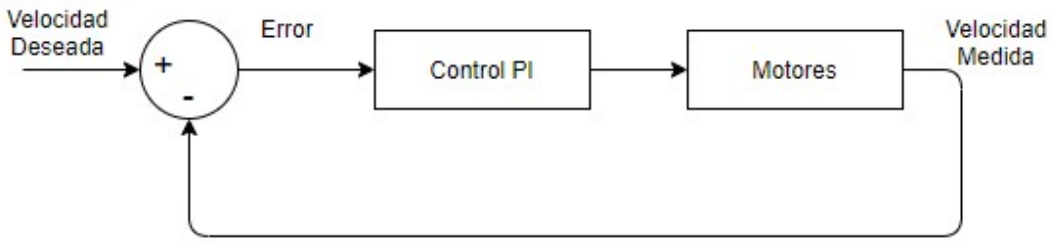


Ilustración 33: Esquema de control PI

Para que el control sea efectivo y confiable, es necesario calcular las constantes proporcional e integral de la forma más correcta posible. Para llevar a cabo esta tarea se han conectado dos potenciómetros a los conversores analógico a digital, con los que se variarán las constantes, comprobando la respuesta de velocidad obtenida (la visualización se hará en la interfaz gráfica, que se detallará más adelante).

Este ajuste se seguirá realizando hasta que se anule el error de posición, y el tiempo de establecimiento sea lo mínimo posible para no interferir en nuestra tarea. Estas comprobaciones se repetirán para cada uno de los tres motores usados, ya que aunque sean el mismo modelo y de la misma marca, puede haber algún pequeño defecto en alguno de ellos que haga necesario algún retoque en las variables de control.

Los valores resultantes de las variables son los siguientes:

	Motor 1	Motor 2	Motor 3
Kp	0.09	0.09	0.09
Ki	0.35	0.35	0.35

El código encargado de realizar esta parte seguirá el siguiente esquema:

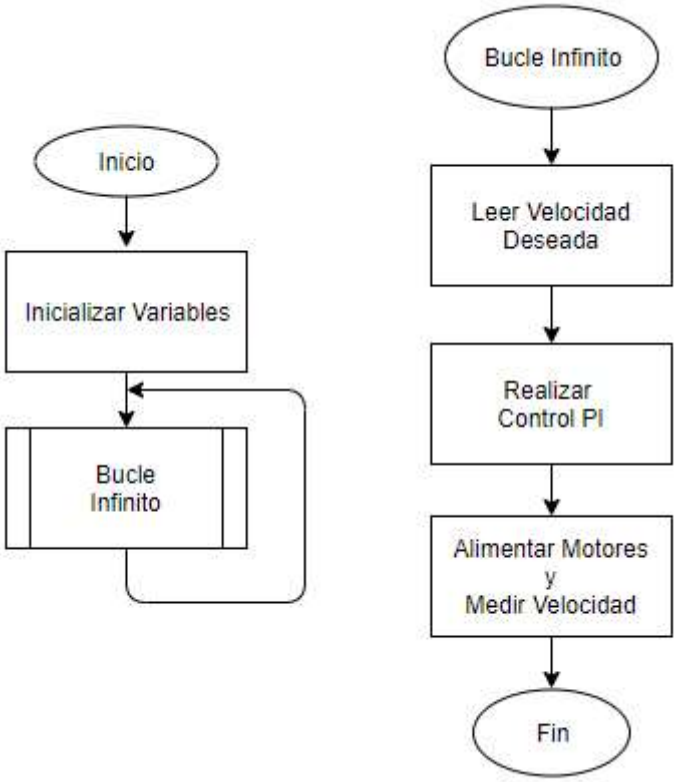


Ilustración 34: Flujograma del código

Una vez realizada la parte del microcontrolador, se pasará al diseño del bucle realizado en el ordenador para la monitorizar y realizar los cálculos correspondientes.

6.2.2 Generación de trayectorias y monitorización de resultados

Este bucle tendrá dos objetivos principales, uno de ellos será la generación y seguimiento de la trayectoria, y el otro será la visualización de las distintas velocidades y la comparación entre la trayectoria deseada y la medida.

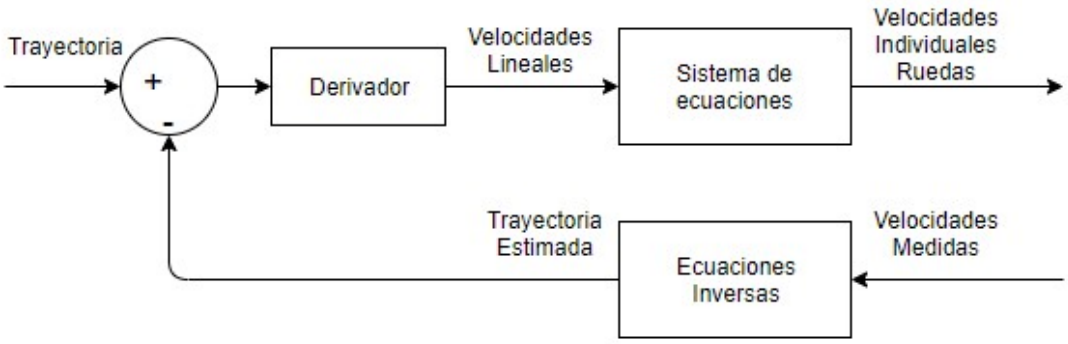


Ilustración 35: Bucle del PC

Para empezar, en este bucle se dispondrá de un mando estilo videoconsola, mediante el cual se podrá elegir entre realizar un control manual o moverse de acuerdo a una trayectoria previamente predefinida. Se entrará en más detalle sobre el control manual más adelante.

Independientemente del control elegido, al principio de nuestro bucle estará la posición cartesiana a la que queremos que se mueva. Esta posición se comparará con la que se medirá al final del bucle, en función de si hemos llegado o es necesario moverse a una dirección u otra, se enviará la velocidad lineal horizontal y vertical a la que queremos que se mueva el robot (estas velocidades serán fijas y estarán predefinidas).

A partir de estas velocidades lineales, mediante las ecuaciones explicadas en el apartado de estudio teórico, se calculará la velocidad y el sentido que debe llevar cada una de las ruedas para que el vehículo se mueva a la posición deseada.

Estos valores se enviarán al vehículo, el cual llevará a cabo el control de velocidad explicado anteriormente, y el vehículo mandará de vuelta las velocidades de los motores, medidos con sus correspondientes encoders. Al recibir el ordenador las velocidades, generará una serie de gráficas comparando la velocidad enviada por el PC al vehículo, con la medida mediante los encoders.

Por otra parte, se realizará el proceso inverso al de las ecuaciones usadas anteriormente. Se pasarán las velocidades de las ruedas a través de la inversa de la matriz, obteniendo así las velocidades lineales, que se integrarán para obtener la trayectoria medida que realiza el robot (importante saber que esta trayectoria no tiene por qué coincidir con la real, ya que se genera a partir de las velocidades, no de un sensor que mida la posición exacta).

Para finalizar la parte del PC, habrá una pequeña interfaz visual en la que se podrán visualizar varios aspectos de interés. Habrá un espacio cuadrado destinado a comparar la trayectoria generada y la medida, tanto en el caso del movimiento automático, como cuando se introduzcan los puntos de destino mediante el uso del modo manual. En el lado derecho de la pantalla, podrán observarse las velocidades de giro de cada una de las ruedas.

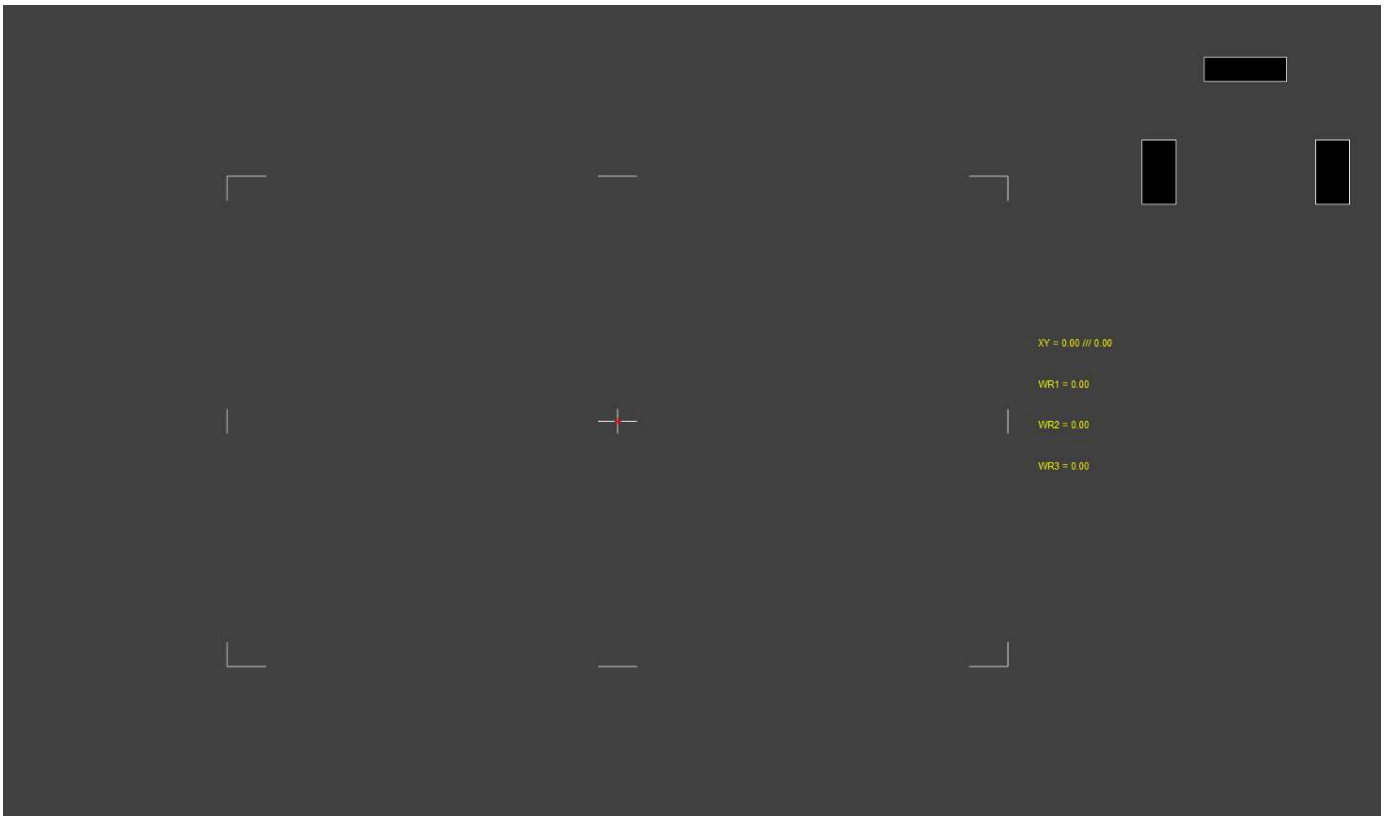


Ilustración 36: Interfaz visual

Encima de ellas, habrá una representación visual de las ruedas, que tomarán un color rojo o verde en función del sentido de giro en el que se encuentren, y haciéndose de tonos más claros cuando la velocidad sea más grande. A medida que el vehículo se vaya moviendo, dejará dibujada una línea roja a su paso, que podremos comparar con la verde generada por la propia trayectoria objetivo.

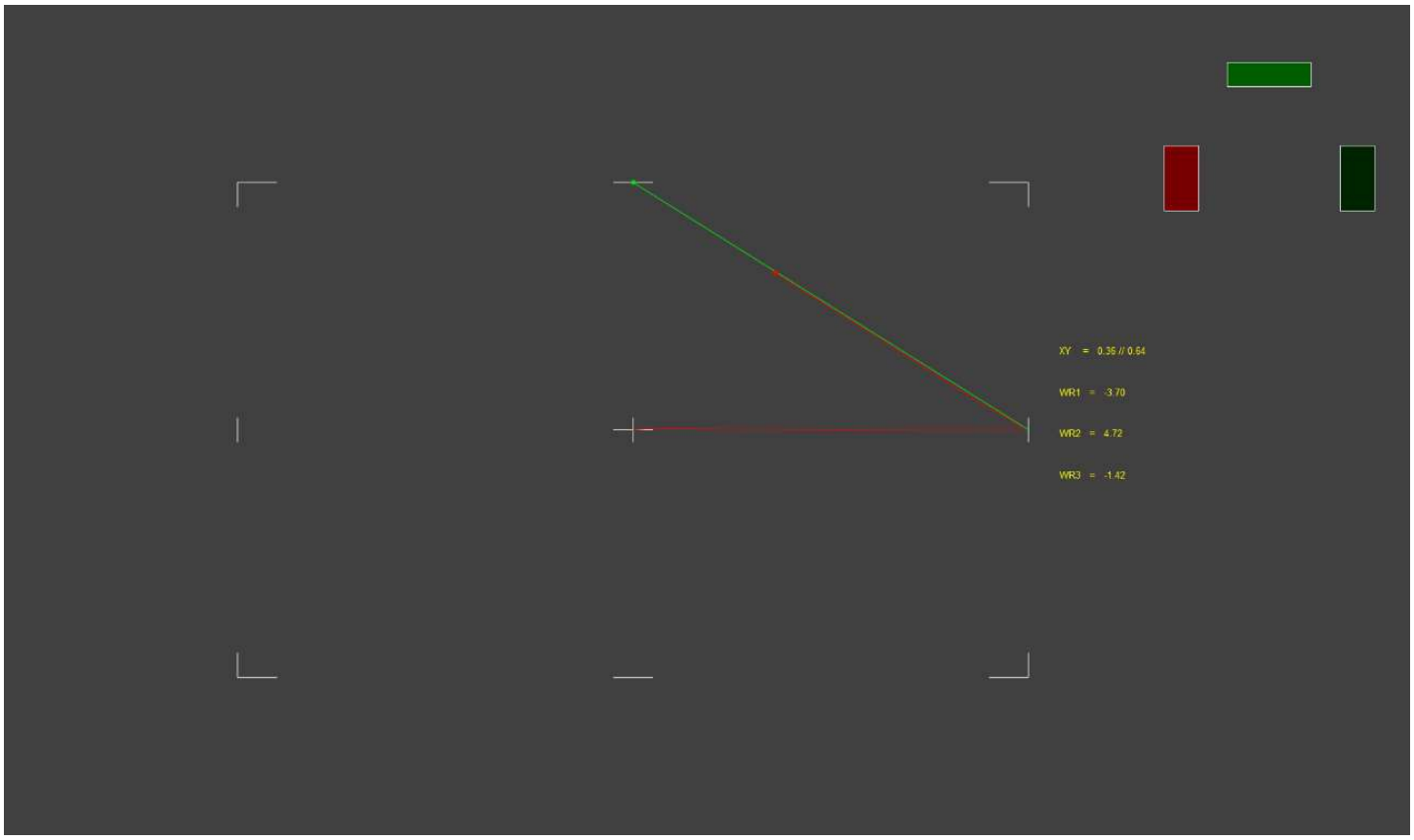


Ilustración 37: Interfaz visual ejemplo de funcionamiento

6.3 Análisis de resultados

En este apartado de incidirá en los resultados obtenidos a partir de los apartados anteriores, por si fuera necesario realizar algún ajuste final para mejorar la respuesta, o detectar algún fallo que haya pasado desapercibido.

Primero se visualizará la respuesta en velocidad de las ruedas para una trayectoria determinada:

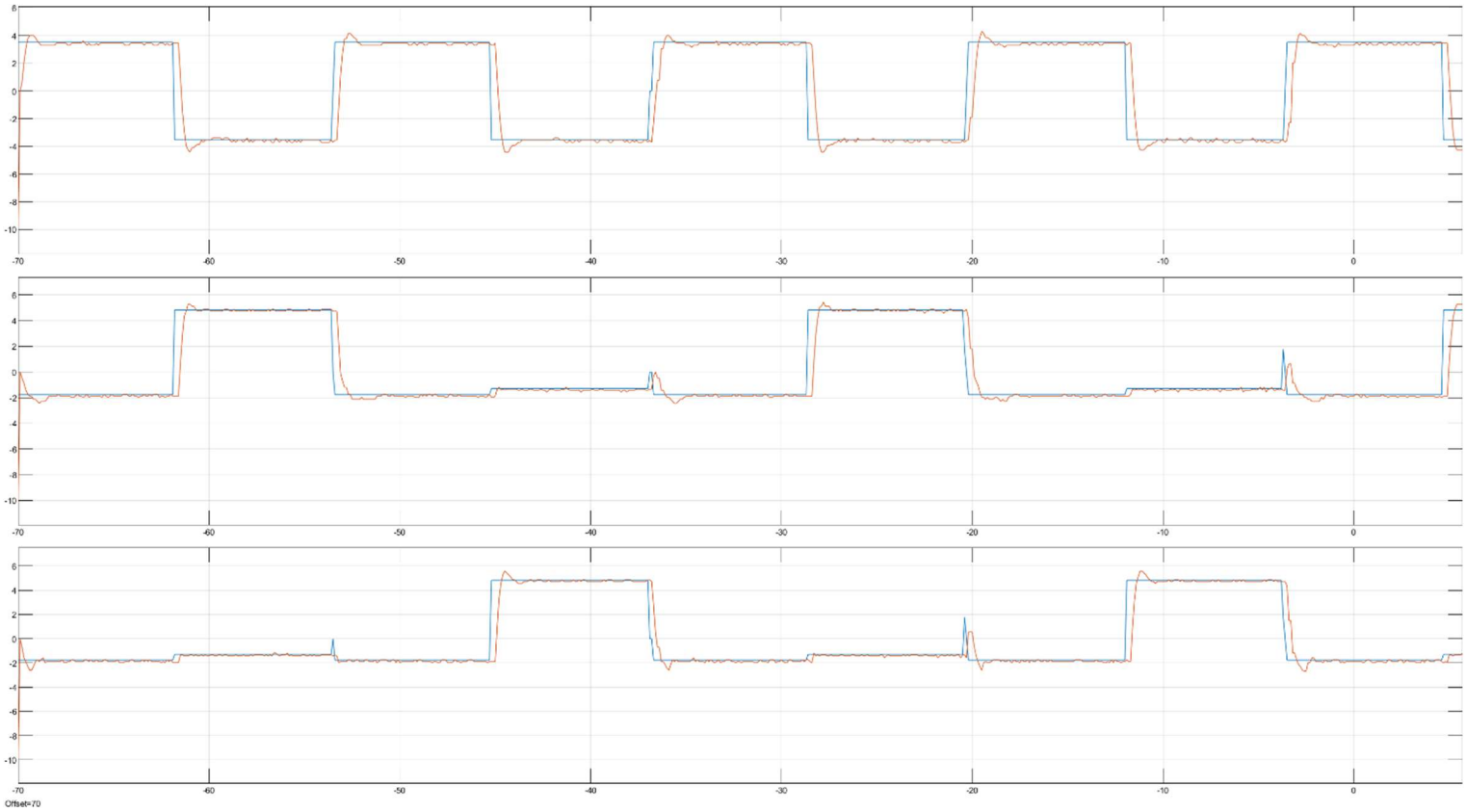


Ilustración 38: Velocidades ruedas para trayectoria ejemplo

Lo primero que se puede observar es un pequeño retraso en la señal de la velocidad respecto a la de referencia. Esto se debe a que la comunicación Bluetooth tarda 100ms en llegar desde el PC al Arduino, y otros 100ms para devolver la respuesta al PC. En nuestro caso esto no tiene por qué suponer un problema, ya que al ser una trayectoria preprogramada y no ser un valor de tiempo demasiado grande, puede que afecte muy poco al seguimiento, esto se comprobará más adelante.

Por otra parte puede verse que aunque haya un poco de sobreoscilación no es demasiado grande, y además el transitorio se produce bastante rápido, por lo que en un principio no parece necesario el reajuste de las constantes del controlador.

Pasamos ahora a ver el seguimiento de la trayectoria de la cual se han obtenido los resultados de velocidad anteriores:

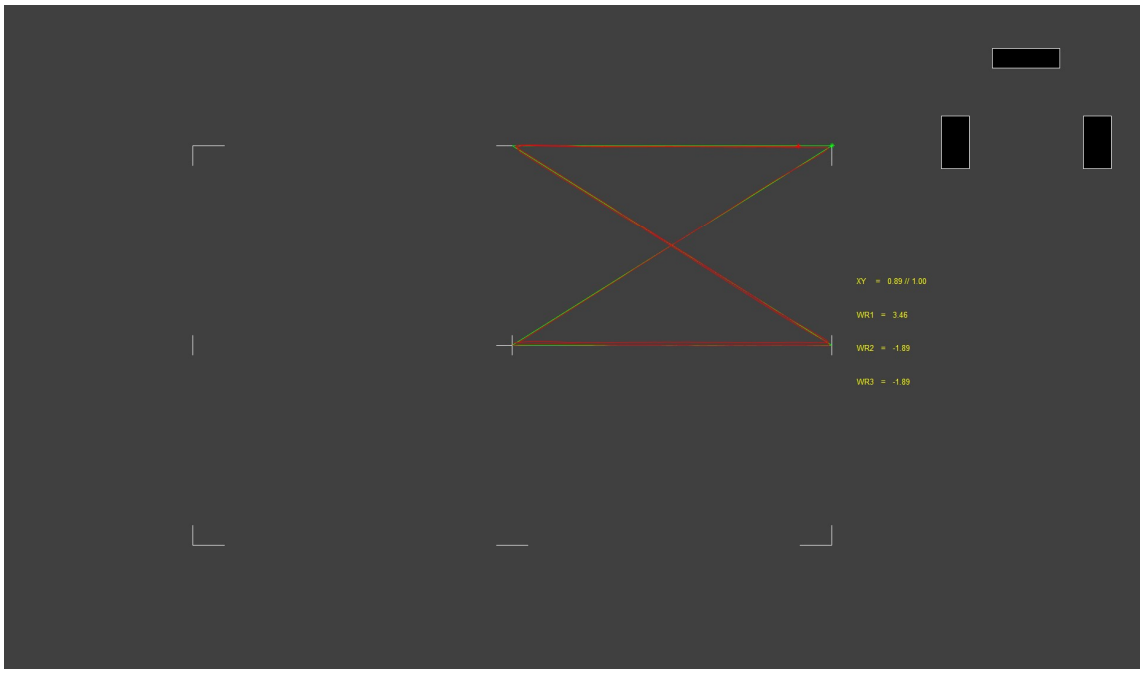


Ilustración 39: Trayectoria ejemplo en la Interfaz

Como puede verse en la fotografía, el seguimiento es prácticamente perfecto, por lo que no es necesario buscar una mejor respuesta de velocidad.

Destacar que esta comparación es entre la trayectoria generada y la medida a través de las velocidades, por lo que no se tienen en cuenta factores como el rozamiento, algún deslizamiento ocasional, o pequeñas faltas de sincronización entre los motores. Esto puede provocar que la trayectoria real difiera en mayor o menor medida, pero teniendo en cuenta que no se dispone de medios para medir y corregir la posición en tiempo real, este resultado indica que el control es lo suficientemente correcto para esta aplicación.

Finalmente, se visualizará el comportamiento real llevado por el vehículo a lo largo de toda la trayectoria.

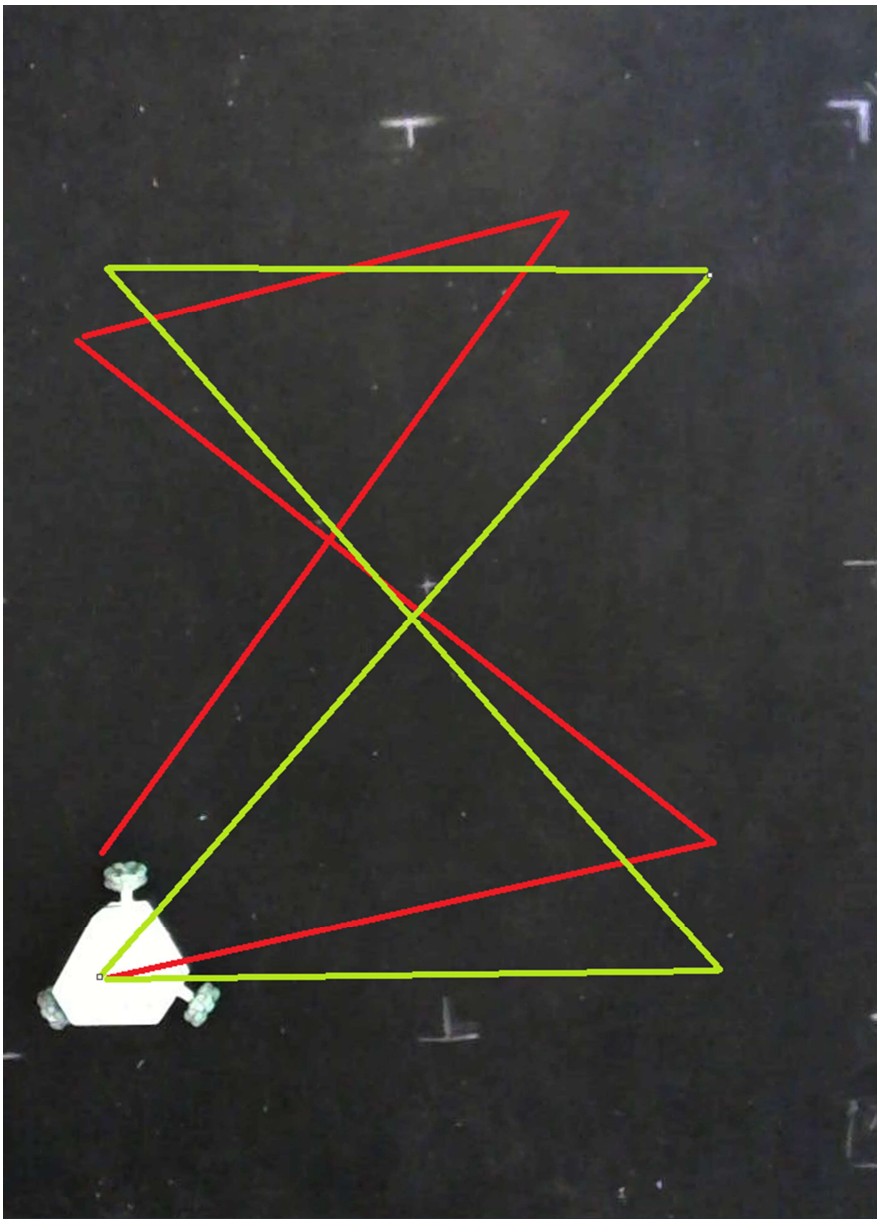


Ilustración 40: Trayectoria ejemplo en el mundo real

Como puede observarse, y tal como se ha comentado antes, aunque en las simulaciones el seguimiento ha sido casi perfecto, en el caso real se han producido desvíos. Esto se debe principalmente a que al producirse los cambios de velocidad en los motores, se producen pequeños deslizamientos que alteran la trayectoria. Como en las simulaciones el comportamiento era perfecto, y hay un buen agarre con el suelo, no hay nada que se pueda retocar para mejorar la respuesta, sin embargo, sí hay cosas que se puedan añadir.

Para esto, se detallarán en el siguiente apartado algunas medidas útiles para mejorar el proyecto.

Por otra parte, va a observarse la respuesta de velocidad usando el movimiento mediante modo manual, haciendo cambios de forma más rápida que en los casos anteriores:

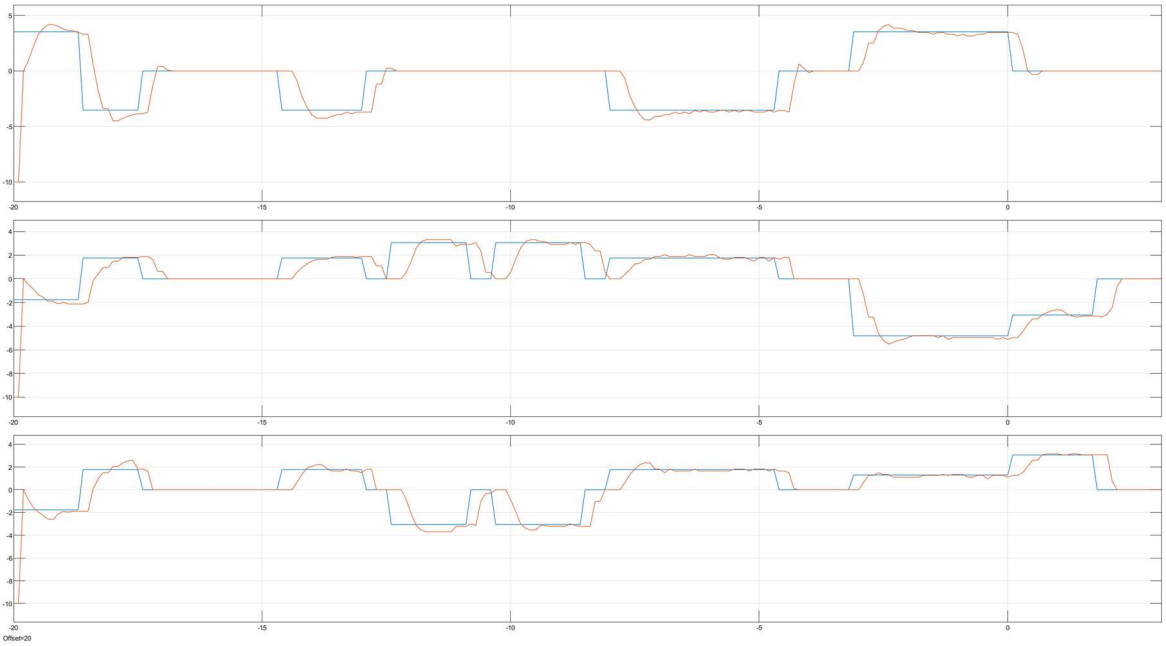


Ilustración 41: Velocidades ruedas en modo manual

Como puede observarse, cuando se producen varios cambios rápidos seguidos la respuesta es peor que en el caso de la trayectoria. De esto podemos deducir que para un correcto funcionamiento el robot debe tener suficiente tiempo para llegar al destino actual, antes de indicarle uno nuevo, o puede haber errores grandes en el seguimiento. Es importante tener esto en cuenta a la hora de diseñar las trayectorias, para que el vehículo pueda realizarlas de forma correcta.

7 Propuestas de mejora

7.1 Visualización del entorno mediante cámara

La primera propuesta de mejora presentada, es posiblemente la que mayor impacto puede tener de todas las propuestas. Tal como se ha explicado anteriormente, la trayectoria medida para la realimentación se genera a partir de las velocidades de las ruedas, por lo que no podemos asegurar que sea fiel a la real (puede serlo, o puede haber diferencias). Esto puede deberse a factores como el deslizamiento rueda-suelo, suelo irregular que provoque momentos donde una rueda esté levantada, y otra serie de factores que aunque no alteren la velocidad de las ruedas, hagan que el vehículo no se mueva como debería.

Para evitar esto, se propone que se conecte una cámara para visualizar el entorno de trabajo, y que mediante una serie de sensores (por ejemplo balizas) se pueda identificar la posición exacta del vehículo, lo que permitiría corregir de forma precisa la trayectoria a tiempo real.

7.2 Giróscopo

Tal como está diseñado el control, el vehículo solo usa los movimientos lineales para el seguimiento de trayectorias. Esto es debido a que un pequeño error a la hora de realizar los giros puede alterar la trayectoria de forma bastante grande, para evitar esto, se propone la incorporación de un giróscopo. Con su ayuda, podrá medirse en tiempo real el ángulo de giro, por lo que se corregirían las pequeñas rotaciones que puedan producirse, además de permitir trayectorias más complejas, que usen este tipo de movimiento, o bien para su uso en el propio entorno de trabajo, por ejemplo si el robot llevara acoplada una herramienta.

7.3 Selección de trayectoria

Para el prototipo desarrollado, solo se ha incluido el modo manual y una trayectoria preprogramada. Teniendo en cuenta la metodología usada para el desarrollo del programa, podrían añadirse más trayectorias, por las que además se podría navegar de forma sencilla gracias al control mediante gamepad.

8 Pliego de condiciones

8.1 Definición y alcance del pliego de condiciones

8.1.1 Objeto del pliego

A continuación se agruparán las condiciones técnicas de obligado cumplimiento para conseguir desarrollar de forma exitosa el proyecto descrito.

El objetivo es desarrollar un vehículo capaz de moverse siguiendo una trayectoria preprogramada, además de un modo manual. Por otra parte se preparará una interfaz de usuario que permita la monitorización de las velocidades y seguimiento de la trayectoria.

Las soluciones mostradas pueden estar sujetas a variaciones en función del caso específico, así como del entorno en el que se realice el trabajo.

8.1.2 Descripción general del procedimiento

Para realizar el procedimiento de forma correcta, el orden seguido ha sido el siguiente:

1. Teorización del vehículo a desarrollar
2. Obtención y/o diseño gráfico de las piezas necesarias
3. Simulación del sistema
4. Impresión 3D de las piezas
5. Montaje físico de los componentes
6. Testeo de piezas por separado
7. Diseño del bucle de control PI
8. Establecimiento de comunicación Bluetooth Arduino-PC
9. Diseño del control PC
10. Implementación de la interfaz visual
11. Depuración de errores

Estas tareas deben ser realizadas por un profesional capacitado para ello, garantizando de este modo que pueda resolver cualquier problema producido durante el desarrollo, el montaje o la puesta a punto del vehículo.

8.2 Condiciones y normas de carácter general

Será de obligado cumplimiento en todo momento la normativa vigente relativa a este tipo de proyectos.

Cualquier modificación debidamente justificada y por pequeña que sea será siempre realizada bajo la supervisión del realizador del proyecto. En caso contrario, la responsabilidad recaerá sobre quien haya realizado la modificación, con las consecuencias penales que ello pueda ocasionar.

8.3 Estudios de legislación

Dada la naturaleza del proyecto, el prototipo presentado en sí mismo no requiere el cumplimiento de ninguna normativa específica. En este caso, el estudio debería realizarse sobre la tarea en la que se realizará el trabajo, como por ejemplo, las normativas con respecto a contaminación o medio ambiente. Por lo que no sería necesario añadir normativa de forma expresa por la incorporación del vehículo.

9 Presupuesto

9.1 Introducción

Con la premisa de estimar económicamente el coste de realización del proyecto, se elaborará un documento con dicho objetivo.

Los presupuestos se dividirán en varios apartados en función de su naturaleza dentro del propio proyecto. Estos serán: montaje físico,

9.2 Montaje físico

A continuación se indica el precio de cada tipo de componente utilizado, así como las cantidades necesarias de cada uno.

Hardware	Coste por unidad	Unidades necesarias	Total (€)
Ruedas Vex EDR	30 € (cada par)	2	60,00 €
Arduino Due	29,65 €	1	29,65 €
Módulo Bluetooth	2,64 €	1	2,64 €
Acoplador	1,32 €	3	3,96 €
Motor Pololu	48,28 €	3	144,84 €
Motor Shield V1	9,90 €	1	9,90 €
Coste total			250,99 €

9.3 Recursos de personal

En este apartado se valorará la participación humana. Para este proyecto solo será necesaria la intervención de una persona, que será el ingeniero al que se le ha encargado la solución del problema. Su salario estará preestablecido antes de comenzar con el desarrollo.

Se realizará un desglose por horas, identificando el tiempo que ha sido necesario para cada una de las actividades llevadas a cabo en el desarrollo del proyecto.

Actividad	Coste/hora (€/h)	Horas empleadas (h)	Total por actividad
Análisis del problema	20,00 €	4	80,00 €
Diseño de la solución:			
Simulación	20,00 €	20	400,00 €
Montaje	20,00 €	5	100,00 €
Diseño de control	20,00 €	40	800,00 €
Depuración de errores	20,00 €	10	200,00 €
Coste total			1.580,00 €

9.4 Gastos de software

A continuación se listará el software utilizado y el coste de cada licencia:

Software	Coste por unidad	Unidades	Total (€)
Licencia Matlab	2.000,00 €	1	2.000,00 €
Licencia Arduino	Gratuito	1	0,00 €
Coste total			2.000,00 €

9.5 Presupuesto final

Finalmente, se suman todos los gastos mencionados anteriormente para obtener el coste total que tiene el proyecto.

Concepto	Coste
Hardware	250,99 €
Software	2.000,00 €
Personal	1.580,00 €
Total	3.830,99 €

10 Conclusiones

A lo largo del proyecto descrito, se ha diseñado un vehículo utilizando un sistema de tres ruedas omnidireccionales, que colocadas de forma concreta, posibilitan una muy buena movilidad en un espacio reducido.

Para realizar la teorización y comprobación del correcto diseño del vehículo, se ha realizado una simulación partiendo desde partes pequeñas como pueden ser las ruedas, hasta el vehículo montado completamente. Para verificar el buen funcionamiento, se han realizado una serie de pruebas de movimiento mientras se monitorizaban las velocidades y posiciones cartesianas de distintos componentes, como pueden ser las ruedas o el centro de masa del robot completo.

Teniendo en cuenta los resultados de las simulaciones, se ha realizado un montaje físico del conjunto, sobre el que se ha diseñado un control de velocidad llevado a cabo en un microcontrolador, y un generador de trayectorias en el PC. Este control, pese a las limitaciones presentadas por los motores usados, ha producido una buena respuesta a las distintas pruebas presentadas.

Los resultados obtenidos en las simulaciones han sido completamente satisfactorios, mientras que los medidos a partir de los encoders también han resultado gratamente positivos. Sin embargo, estos resultados difieren al fijarnos en la trayectoria real descrita por el vehículo, debido a las limitaciones físicas del modelo (como rozamientos, deslizamientos o los propios motores). Los resultados reales pueden ser mejorados gracias a las alternativas comentadas en apartados anteriores.

A partir de todos los resultados obtenidos, y teniendo en cuenta las posibles mejoras, se concluye que el proyecto se ha realizado de forma satisfactoria y se han cumplido todos los objetivos planteados inicialmente.

En este proyecto se han aplicado conocimientos de distintos campos dentro de la ingeniería, como pueden ser conocimientos de cinemática para la resolución teórica del movimiento, conocimientos de diseño y mecánica para el desarrollo del prototipo, y conocimientos de electrónica e informática para el apartado de control y programación del movimiento.

Finalmente, agradecer a mi tutor D. Vicente Fermín Casanova Calvo, quien me ha proporcionado una gran cantidad de conocimientos sobre el mundo del control automático, y me ha guiado a lo largo de este proyecto, resolviendo dudas y problemas que han ido surgiendo en el desarrollo, y sugiriendo ideas para su mejora.

Bibliografía

[1] https://store.arduino.cc/due

[2] https://www.cosues.com/65414-vex-edr-ruedas-omnidireccionales

http://wikitronica.labc.usb.ve/index.php/Robot_omnidireccional

Vehículo compacto omnidireccional con ruedas no convencionales. Diseño de grupos motrices y chasis de configuración adaptable a diversas aplicaciones – Francesc Ros Cerro:

https://upcommons.upc.edu/bitstream/handle/2099.1/2646/34562-1.pdf

Control y comportamiento de Robots Omnidireccionales – Santiago Martínez, Rafael Sisto:

https://www.fing.edu.uy/inco/grupos/mina/pGrado/easyrobots/doc/SOA.pdf

https://www.pololu.com/product/2825/resources

https://riunet.upv.es/bitstream/handle/10251/87465/LACASA%20-%20Control%20y%20Supervisi%C3%B3n%20de%20procesos%20mediante%20red%20industrial%20Profibus%20y%20Sistemas%20Scada.pdf?sequence=1

Anexo 1:

Datasheets

Arduino due

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

Pololu 37d mm Metal Gearmotor

Dimensions

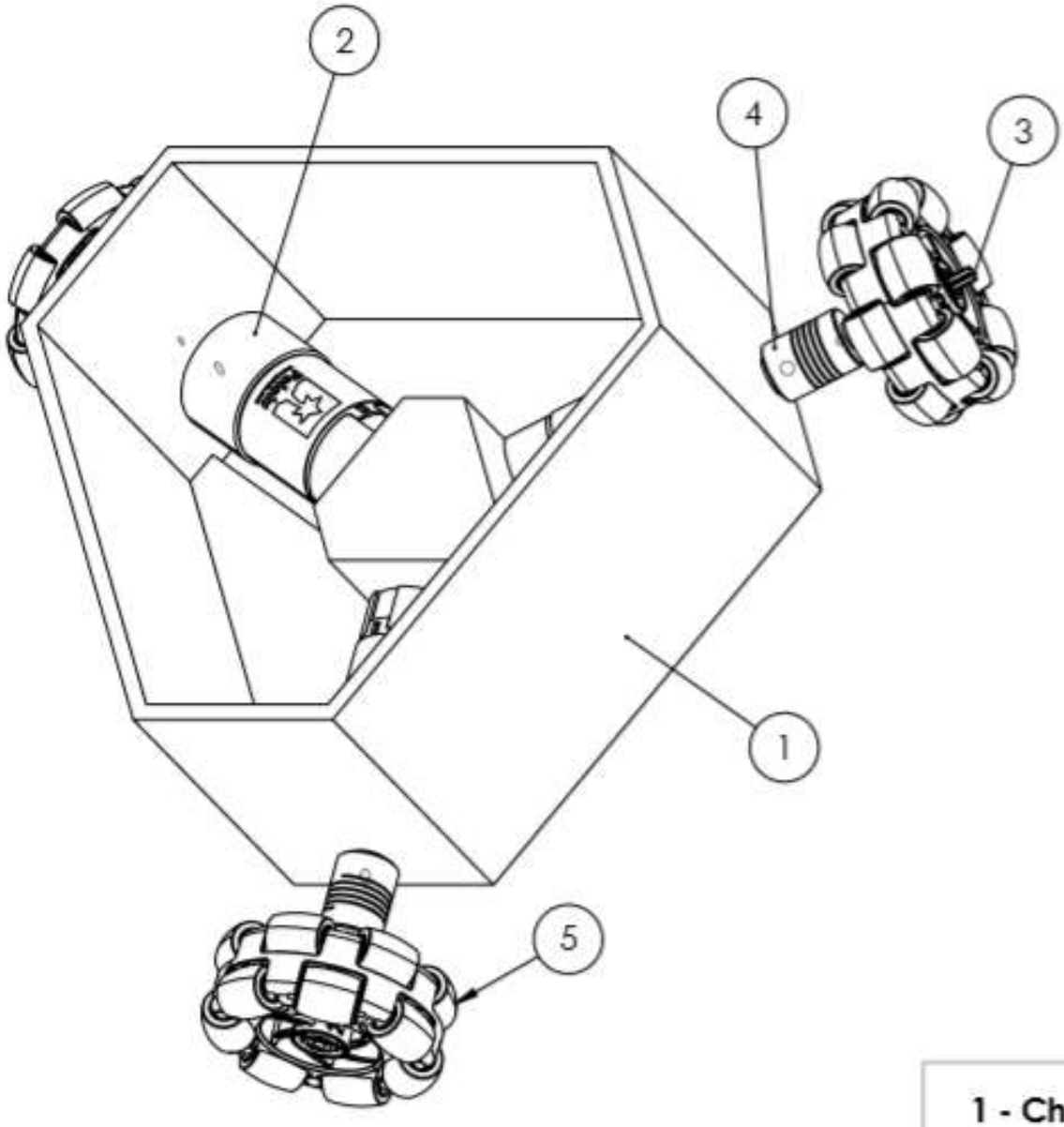
Size:	37D x 70L mm[1]
Weight:	225 g
Shaft diameter:	6 mm

General specifications

Gear ratio:	50:1
No-load speed @ 12V:	200 rpm
No-load current @ 12V:	300 mA
Stall current @ 12V:	5000 mA
Stall torque @ 12V:	170 oz·in
No-load speed @ 6V:	100 rpm[2]
No-load current @ 6V:	250 mA[2]
Stall current @ 6V:	2500 mA[2]
Stall torque @ 6V:	85 oz·in[2]
Lead length:	8 in[3]

Anexo 2:

Planos



1 - Chasis

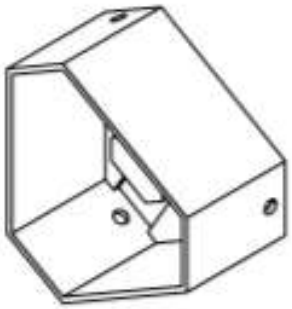
2 - Motor

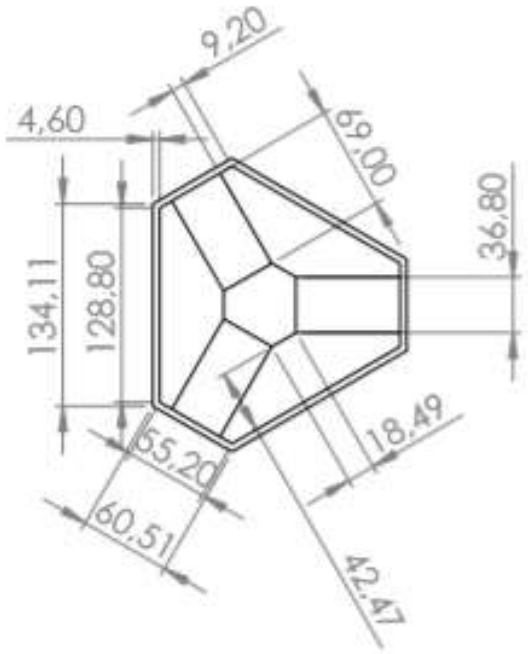
3 - Acople 1

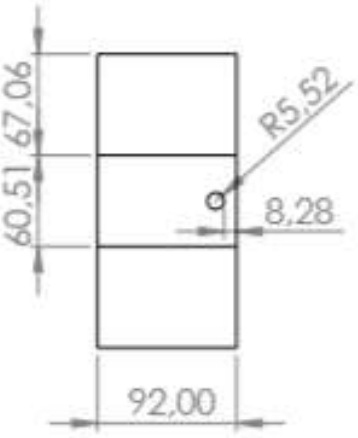
4 - Acople 2

5 - Ruedas

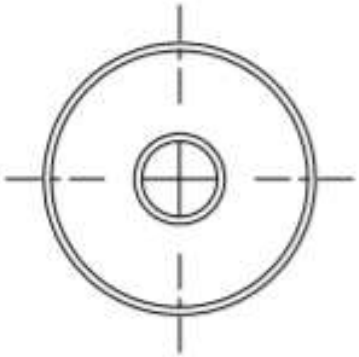
ETSID - UPV	Título: Vehículo Completo
Dibujado	
Domingo Lacasa Collado	
Fecha: 29/06/2019	Nº de dibujo: Dibujo 1 — A4
	Escala: 1:5

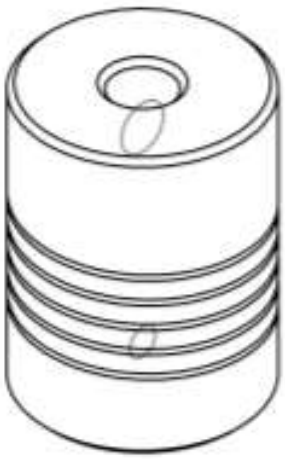


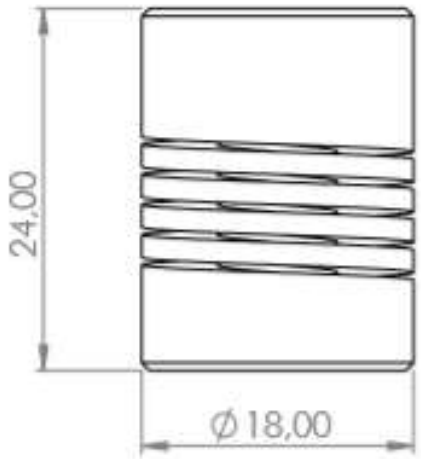


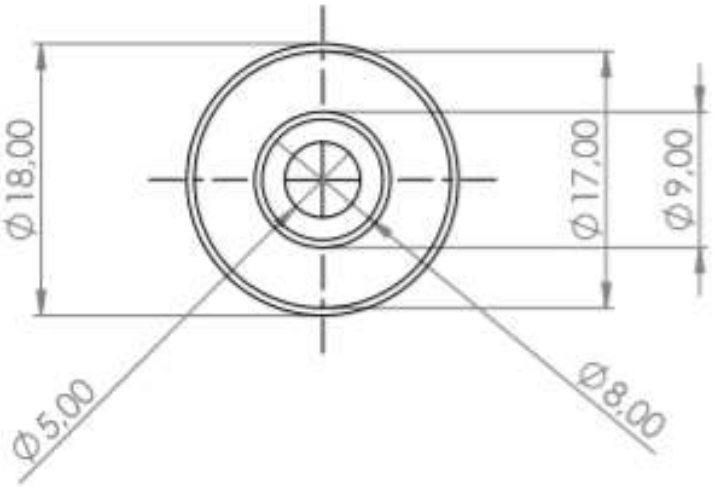


ETSID - UPV	TÍTULO: Chasis
Dibujado	
Domingo Lacasa Collado	
Fecha 29/06/2019	N.º DE DIBUJO Dibujo 2 — A4
	ESCALA:1:5

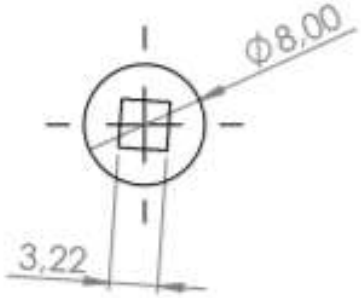


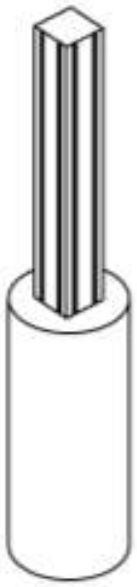


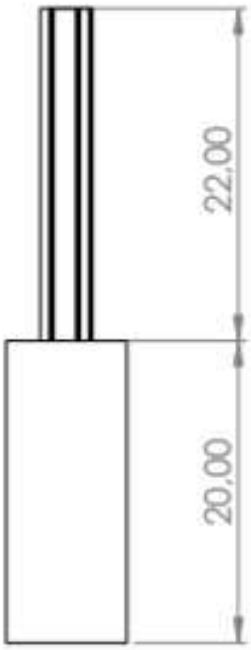




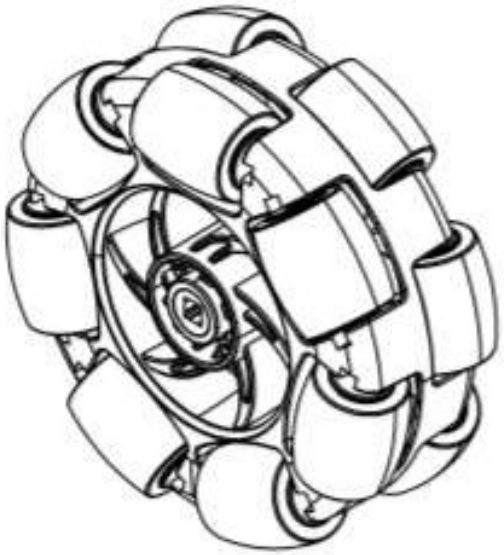
ETSID - UPV	TÍTULO: Acople
Dibujado	
Domingo Lacasa Collado	
Fecha 29/06/2019	N.º DE DIBUJO Dibujo 3
	A4
	ESCALA:2:1





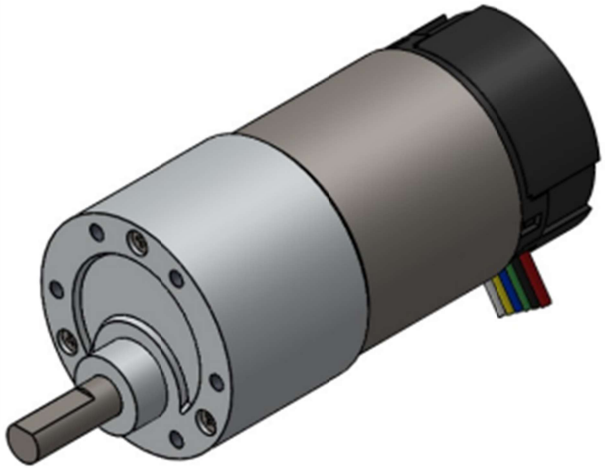


ETSID - UPV	TÍTULO: Conexión Rueda
Dibujado	
Domingo Lacasa Collado	
Fecha 29/06/2019	N.º DE DIBUJO Dibujo 4
	A4
	ESCALA:2:1



ETSID - UPV	TÍTULO: Rueda
Dibujado	
Domingo Lacasa Collado	
Fecha 29/06/2019	N.º DE DIBUJO Dibujo 5
	A4
	ESCALA:1:1

Gear ratio	L
19:1	22 mm [0.87 in]
30:1	22 mm [0.87 in]
50:1	24 mm [0.94 in]
70:1	24 mm [0.94 in]
100:1	26.5 mm [1.04 in]
131:1	26.5 mm [1.04 in]
150:1	26.5 mm [1.04 in]



1. To get the specified scale, select 100% in print settings.

Scale: 1:1

https://www.pololu.com/category/116/37d-mm-gearmotors

Name:	Item number:
37D mm Metal Gearmotors with 64 CPR Encoder	2822-2828
Drawing date: 02 April 2019 / Dev code: —	
Units: mm [in] / Material: Mixed	

Pololu Robotics & Electronics

Anexo 3:

Código

En este apartado se incluyen las partes más importantes de código.

- Código para testeo de comunicación Bluetooth

```
if (Serial1.available()>=7) {
   for (int ii=1 ; ii<=7 ; ii++) {
    inByte[ii]=Serial1.read() ;
   }
   if (inByte[1]=='A'){
      if (inByte[2]=='X' && inByte[5]=='Y') {
       x=(inByte[3]-48)*10+(inByte[4]-48) ;
       xf=map(x,10,99,-100,100)/100.0 ;
       if(xf<=0.05 && xf>=-0.05){xf=0;}
       dato1=xf;
       //x=Serial1.parseInt() ;
       Serial.print("X=") ; Serial.print(xf) ; Serial.print(" ; ") ;
       //y=Serial1.parseInt() ;
       y=(inByte[6]-48)*10+(inByte[7]-48) ;
       yf=map(y,10,99,-100,100)/100.0 ;
       if(yf<=0.05 && yf>=-0.05){yf=0;}
       dato2=(-1)*yf;
       Serial.print("Y=") ; Serial.print(yf) ; Serial.println(" ; ") ;
      }
   }
   if (inByte[1]=='B'){
    if (inByte[2]=='X' && inByte[5]=='Y') {
         w=(inByte[3]-48)*10+(inByte[4]-48) ;
         wf=map(w,10,99,-100,100)/100.0 ;
         if(wf<=0.05 && wf>=-0.05){wf=0;}
         dato3=wf;
         //x=Serial1.parseInt() ;
         Serial.print("W=") ; Serial.print(wf) ; Serial.print(" ") ;
         //y=Serial1.parseInt() ;

      }
   }
 } else {
    na++ ;
    Serial.println("---XXX---") ;
    if (na==10) {stp=1 ; na=0 ;}
   }
 }
```

- Bucle de control PI con compensación de zona muerta

```
EkA=veldesA-velA;
  UpkA=EkA*KpA;
  Ik_actu_A=EkA*Tm+Ik_ante_A;
  if(Ik_actu_A>satI){
    Ik_actu_A=satI;}
  Ik_ante_A=Ik_actu_A;
  UikA=Ik_actu_A*KiA;
  accA=UpkA+UikA;

  if(accA<0.35 && accA>0.05){accA=0.35;}
  else if(accA>=0 && accA<=0.05){accA=0;}
  else if(accA<0 && accA>=-0.05){accA=0;}
  else if(accA<-0.05 && accA>-0.35){accA=-0.35;};
```

- Escritura en los motores

```
if (abs(accA)>SAT) {accA=(accA/abs(accA))*SAT ;  Serial.println("SATURACIÓN MOTOR A") ;}
  if (abs(accB)>SAT) {accB=(accB/abs(accB))*SAT ;  Serial.println("SATURACIÓN MOTOR B") ;}
  if (abs(accC)>SAT) {accC=(accC/abs(accC))*SAT ;  Serial.println("SATURACIÓN MOTOR C") ;}

  if (accA>0) {motor(1,FORWARD,(byte)(accA*255.0/SAT)) ;}
  else if (accA<0) {motor(1,BACKWARD,(byte)(-accA*255.0/SAT)) ;}
  else {motor(1,BRAKE,0) ;}
  if (accB>0) {motor(2,FORWARD,(byte)(accB*255.0/SAT)) ;}
  else if (accB<0) {motor(2,BACKWARD,(byte)(-accB*255.0/SAT)) ;}
  else {motor(2,BRAKE,0) ;}
  if (accC>0) {motor(3,FORWARD,(byte)(accC*255.0/SAT)) ;}
  else if (accC<0) {motor(3,BACKWARD,(byte)(-accC*255.0/SAT)) ;}
  else {motor(3,BRAKE,0);}

void motor(int nMotor, int command, int speed) {
  int motorA, motorB;
  if (nMotor >= 1 && nMotor <= 4) {
    switch (nMotor) {
      case 1: motorA = MOTOR1_A; motorB = MOTOR1_B; break;
      case 2: motorA = MOTOR2_A; motorB = MOTOR2_B; break;
      case 3: motorA = MOTOR3_A; motorB = MOTOR3_B; break;
      default: break;
    }

    switch (command) {
      case FORWARD: motor_output (motorA, HIGH, speed); motor_output (motorB, LOW, -1); break;
      case BACKWARD: motor_output (motorA, LOW, speed); motor_output (motorB, HIGH, -1); break;
      case BRAKE: motor_output (motorA, LOW, 255);  motor_output (motorB, LOW, -1); break;
      case RELEASE: motor_output (motorA, LOW, 0); motor_output (motorB, LOW, -1); break;
      default: break;
    }
  }
}
```

```
void motor_output (int output, int high_low, int speed) {
  int motorPWM;
  switch (output) {
    case MOTOR1_A:
    case MOTOR1_B: motorPWM = MOTOR1_PWM; break;
    case MOTOR2_A:
    case MOTOR2_B: motorPWM = MOTOR2_PWM; break;
    case MOTOR3_A:
    case MOTOR3_B: motorPWM = MOTOR3_PWM; break;
    default: speed = -3333; break;
  }
  if (speed != -3333) {
    shiftWrite(output, high_low);
    if (speed >= 0 && speed <= 255) {analogWrite(motorPWM, speed); }
  }
}
```

- Lectura encoders

```
void encoderA_chA(){if(digitalRead(pinENCA_chA)==digitalRead(pinENCA_chB)) quadA--; else quadA++;}
void encoderA_chB(){if(digitalRead(pinENCA_chA)==digitalRead(pinENCA_chB)) quadA++; else quadA--;}
void encoderB_chA(){if(digitalRead(pinENCB_chA)==digitalRead(pinENCB_chB)) quadB--; else quadB++;}
void encoderB_chB(){if(digitalRead(pinENCB_chA)==digitalRead(pinENCB_chB)) quadB++; else quadB--;}
void encoderC_chA(){if(digitalRead(pinENCC_chA)==digitalRead(pinENCC_chB)) quadC--; else quadC++;}
void encoderC_chB(){if(digitalRead(pinENCC_chA)==digitalRead(pinENCC_chB)) quadC++; else quadC--;}
```

- Configuración pines importantes

```
void setup() {

  Serial.begin(115200) ;
  pinMode(RT,OUTPUT) ;
  Serial1.begin(115200) ;

  analogReadResolution(12) ;                    // Entrada analógica 12 bits (4096)

  pinMode(salida1alto, OUTPUT);
  pinMode(salida1bajo, OUTPUT);
  pinMode(salida2alto, OUTPUT);
  pinMode(salida2bajo, OUTPUT);
  pinMode(salida3alto, OUTPUT);
  pinMode(salida3bajo, OUTPUT);


  attachInterrupt(pinENCA_chA,encoderA_chA,CHANGE) ;
  attachInterrupt(pinENCA_chB,encoderA_chB,CHANGE) ;
  attachInterrupt(pinENCB_chA,encoderB_chA,CHANGE) ;
  attachInterrupt(pinENCB_chB,encoderB_chB,CHANGE) ;
  attachInterrupt(pinENCC_chA,encoderC_chA,CHANGE) ;
  attachInterrupt(pinENCC_chB,encoderC_chB,CHANGE) ;
}
```