Trabajo de Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Diseño, fabricación, montaje, estudio dinámico, control y teleoperación de un vehículo tipo péndulo invertido sobre dos ruedas

Autor: Nicolás Cortés Fernández
Tutor: Ignacio Alvarado Aldea

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo de Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Diseño, fabricación, montaje, estudio dinámico, control y teleoperación de un vehículo tipo péndulo invertido sobre dos ruedas

Autor:
Nicolás Cortés Fernández

Tutor:
Ignacio Alvarado Aldea
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Diseño, fabricación, montaje, estudio dinámico, control y teleoperación de un vehículo tipo péndulo invertido sobre dos ruedas

Autor: Nicolás Cortés Fernández

Tutor: Ignacio Alvarado Aldea

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mis padres

A mis hermanos

A mis abuelos

A los amigos que me llevo conmigo

Agradecimientos

En primer lugar, agradecerle a mi padres y hermanos la paciencia que han tenido conmigo. Pocas personas hubiesen aguantado los ruidos de las caídas y pitidos a las 12 de la noche del vehículo balancín del que trata este trabajo. Sin esa paciencia no hubiera sido posible la consecución del objetivo final, pues el vehículo habría sido estampado contra la pared a las primeras de cambio. Si más bromas, gracias a todos ellos.

Gracias a mi abuelo Santi, a mi abuela Concha, a mi abuela Pía y, en especial, a mi abuelo Marce, de quién seguí los pasos para estudiar esta carrera.

Dar las gracias también al profesor Dº. Daniel Limón, quíen me ayudo mucho con todo el tema de la tramitación de la beca de colaboración.

Por último, pero no por menos importante, dar las gracias a mi tutor Dº. Ignacio Alvarado por toda la ayuda recibida, por todo su tiempo dedicado hacia mi persona, por toda su implicación con el proyecto, por su buen trato tanto fisícamente en su despacho como por email… En resumen, ha sido un placer desarrollar este trabajo a su lado.

Nicolás Cortés Fernández

Sevilla, 2019

Resumen

Se han diseñado en 2d las diferentes piezas que, una vez fabricadas con la cortadora láser del departamento a partir de paneles de madera, se emsamblarán y conformarán el chásis del vehículo.

Se ha montado todo el sistema de control (motores, batería, microcontrolador, etc) sobre el chásis, que ha sido especialmente diseñado para poder albergar a cada uno de los diferentes elementos.

Se ha diseñado una PCB que se monta sobre el Arduino_Uno y en donde se halla soldada toda la electrónica (drivers, imu, led, etc). De esta manera se hace innecesario el uso de cables, ya que son las pistas de la placa las que unen eléctricamente los diferentes pines.

Se ha realizado un estudio dinámico del sistema a través de los postulados de la mecánica vectorial, *El Teorema de la Cantidad de Movimiento* y *El Teorema del Momento Cinético.*

Se ha realizado una identificación de los parámetros del modelo más difícilmente medibles, como las inercias, a través de un problema de optimización resuelto con Matlab.

Se ha diseñado el programa de control que se ejecuta en Arduino, que simula la concurrencia temporal entre las diferentes tareas que debe realizar el microcontrolador (lectura de sensores, aplicación de pasos a los motores, etc), de tal manera que se cumplan los tiempos de repetición de las tareas cíclicas y se respeten las prioridades entre tareas.

Se han diseñado e implementado diferentes estrategias de control sobre el sistema. Se ha realizado una comparativa de los mismos bajo multitud de situaciones (perturbaciones tipo pulso a favor y en contra del movimiento, fuerzas constantes, etc.)

Se ha llevado a cabo la teleoperacion vía Bluetooch del vehículo a través del mando Numshuk de la consola Wii.

Abstract

The different pieces have been designed in 2d that, once manufactured with the laser cutter of the department from wooden panels, will be assembled and will conform the chassis of the vehicle.

The entire control system (motors, battery, microcontroller, etc.) has been mounted on the chassis, which has been specially designed to house each of the different elements.

A PCB has been designed that is mounted on the Arduino_Uno and where all the electronics have been welded (drivers, imu, led, etc). In this way the use of cables is unnecessary, since it is the tracks of the plate that connect the different pins electrically.

A dynamic study of the system has been carried out through the postulates of vector mechanics, The Theorem of the Amount of Movement and The Theorem of the Kinetic Moment.

An identification of the parameters of the most difficultly measurable model, such as inertias, was made through an optimization problem solved with Matlab.

The control program that is executed in Arduino has been designed, which simulates the temporal concurrence between the different tasks that the microcontroller must perform (reading of sensors, application of steps to motors, etc.), in such a way that the times are met of repetition of the cyclical tasks and the priorities between tasks are respected.

Different control strategies on the system have been designed and implemented. A comparison has been made of them under a multitude of situations (pulse-type disturbances in favor and against movement, constant forces, etc.)

The vehicle has been teleoperated via Bluetooch via the Numshuk control of the Wii console.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

$\overrightarrow{r_{21}^{B}}$	Vector de posición de un punto B perteneciente al solido 2 visto desde el observador 1
$\overrightarrow{v_{21}^{B}}$	Velocidad de un punto B perteneciente al solido 2 visto desde el observador 1
$\overrightarrow{a_{21}^{B}}$	Aceleración de un punto B perteneciente al solido 2 visto desde el observador 1
G_{TT}	Centro de masas de sistema 2U3U4
G_2	Centro de masas del sólido 2
U	Unión
$\overrightarrow{\phi_{ij}}$	Fuerza vincular sobre el sólido i ejercida por el sólido j

1 INTRODUCCIÓN

En cuestiones de ciencia, la autoridad de mil no vale el razonamiento humilde de un individuo.

Galileo Galilei

En 1561, Galileo Galilei, con tan solo 17 años, asistío a misa en la catedral de Pisa. Allí, una lámpara llena de velas oscilaba de un lado para otro, colgada del techo mediante una cadena. El joven y curioso Galileo midió el periodo de oscilación con las pulsaciones de su muñeca, llegando a un resultado asombroso: aunque la amplitud del movimiento disminuía, el período del movimiento permanecía constante. Desde entonces, y hasta la actualidad, la dinámica del péndulo ha sido ampliamente estudiada generación tras generación. Su movimiento, conocido como *Movimiento Armónico Simple*, esta presente en infinidad de sistemas de la naturaleza, desde las vibraciones atómicas dentro de una red cristalina hasta las vibraciones de la cuerda de una guitarra.

Dentro del campo de la Ingeniería de control, el control del péndulo invertido es uno de los más clásicos y desafiantes. Su dinámica inestable, no lineal, y la presencia de un cero de fase no mínima, hacen de su control un autentico reto. Existen diferentes variantes del problema del control del péndulo invertido, tales como el péndulo de Furuta, el péndulo sobre un carro, el péndulo sobre dos ruedas, etc.

Figura 1-1. Péndulo de Furuta

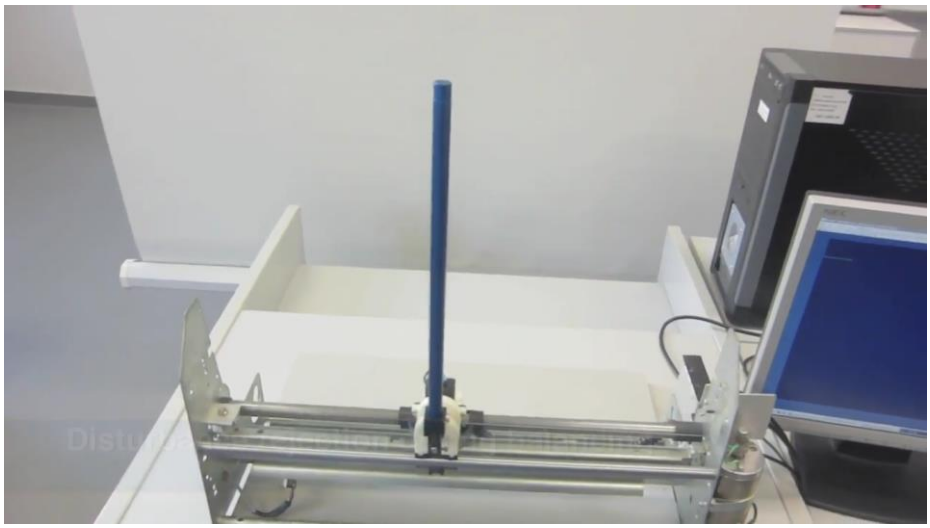


Figura 1-2. Péndulo sobre carro.

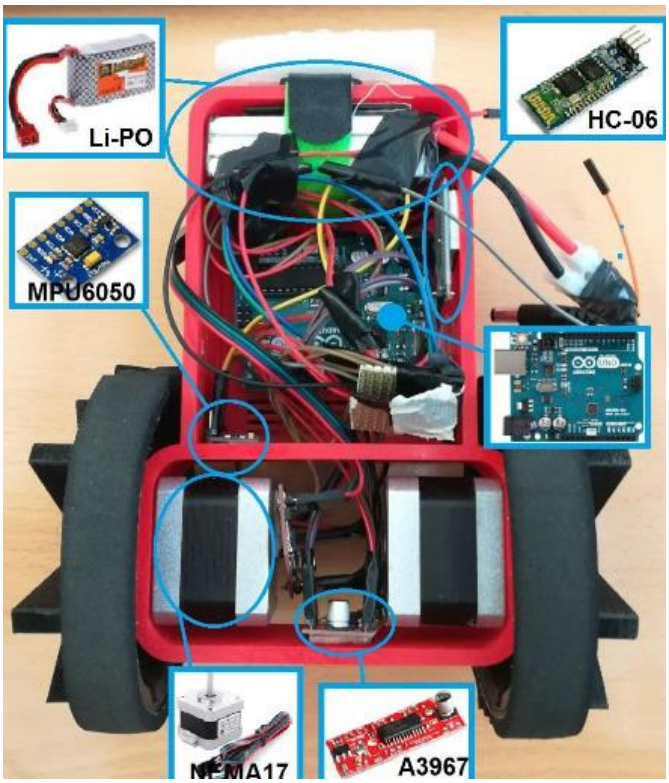


Figura 1-3. Péndulo invertido sobre dos ruedas del departamento.

1.1. Objetivos

La Figura 1-3 muestra el actual péndulo invertido sobre dos ruedas del Departamento de Ingeniería de Sistemas y Automática, de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Fue diseñado por D. Daniel Aparicio. Aunque su diseño mejora notablemente el anterior péndulo invertido sobre dos ruedas del departamento, basado en aluminio y madera, admite algunos puntos de mejora.

- Su fabricación mediante impresión 3d es lenta, lo que hace inviable su fabricación en serie dentro del departamento.
- Componentes electrónicos sueltos y cableado engorroso.

Por tanto, uno de los objetivos del proyecto será el diseño estructural en 2d de las piezas de un nuevo *Two-Wheeled Inverted Pendulum*, como se le denomina en la lengua anglosajona a este tipo de péndulo. Una vez obtenidas las piezas mediante la cortadora láser del departamento, se emsamblarán y formarán el chasis del vehículo. Su emsanblaje con chavetas aportará rígidez absoluta al conjunto, lo que hará innecesario el uso de pegamentos.

El problema del cableado ha sido resuelto a través del diseño y encargo de una placa de circuito impreso. El diseño de una PCB, que se monta sobre el microcontrolador ArduinoUno y en donde se sueldan los diferentes elementos electrónicos, soluciona el problema. Son las pistas metálicas del escudo las que unen los diferentes componentes de la electrónica del sistema de control, haciendo innecesario el uso de cables.

Su rápida fabricación y bajo coste animan enormemente a su fabricación en serie con diferentes fines:

- Renovación del equipamiento de prácticas. Para un alumno resulta mucho más motivamente el control de un sistema como el péndulo invertido que el control de la velocidad de un motor de continua, por ejemplo.
- Servir de base para otros proyectos del departamento como, por ejemplo, la comunicación y cooperación entre varios de estos vehículos para conseguir un fin determinado.
- Su venta al público. Por ejemplo, departamentos de Ingenieria de Control de otras universidades seguramente estarían dispuestos en obtener algunas unidades.
- Trabajos de fin de grado/máster de otros estudiantes.

1.2. Estructura del documento

Una vez obtenido el nuevo vehículo, el objetivo es poner aprueba todo lo aprendido en la carrera para lograr el control y la teleoperación del mismo. Para conseguirlo, se llevarán a cabo los siguientes pasos:

1. En el capítulo 2 de la memoria se estudiará la dinámica del sistema aplicando todo lo aprendido (*Teoremas de la Cantidad de Movimiento y del Momento Cinético)* en la asignatura de *Ampliación de Física*, de 2º del grado en Tecnologías Industriales, y se compararán los resultados con los obtenidos mediante la aplicación de las ecuaciones de Lagrange.
2. Una vez obtenido el modelo dinámico, se realizará en el capítulo 3 de la memoria una identificación de los parámetros más difíciles de medir, como las inercias o la distancia del centro de gravedad al eje de giro de las ruedas. Se elegirá el par de parámetros que minimice el error cuadrático entre la predicción del modelo matemático y el sistema real.
3. En el capítulo 4 se presentará todo el diseño electrónico y estructural. Se mostrarán los diferentes dispositivos, sus características mas importantes y su conexionado. Se hablará también de los requisitos estructurales del chasis, de cómo ha sido diseñado, y se mostrarán numerosas imágenes del vehículo acabado.
4. En el capítulo 5 se diseñará el programa de control que se ejecuta en Arduino, que simula la concurrencia temporal entre las diferentes tareas que debe realizar el microcontrolador (lectura de sensores, aplicación de pasos a los motores, etc), de tal manera que se cumplan los tiempos de repetición de las tareas cíclicas y se respeten las prioridades entre tareas.
5. En el capítulo 6 se diseñarán diferentes controladores tales como LQR o LQR+Efecto_integral con Matlab. Se implementarán sobre el sistema real, y se compararán sus resultados para diferentes situaciones como seguimiento de referencias en velocidad, perturbaciones, etc.
6. En el capítulo 7 se teleoperará via Bluetooch el vehículo, haciendo uso del mando Numshuk de la consola Wii.

Aquí concluye la *Introducción* del trabajo.

2 ESTUDIO DINÁMICO

Un hombre puede imaginar cosas que son falsas, pero
sólo puede entender cosas que son ciertas

Isaac Newton

Isaac Newton (1643-1727) es conocido, entre muchas cosas, por ser el padre de la Mecánica Clásica. En su obra *Philosophiæ naturalis principia mathematica*, publicada el 5 de Julio de 1687, se recogen sus dos grandes teoremas que explicán con una simplicidad matemática asombrosa el por qué los cuerpos se mueven de la forma en que lo hacen. Como bien se sabe, se trata de los teoremas de *La Cantidad de Movimiento* y del *Momento Cinético.*

Hoy en día existen técnicas más potentes que la Mecánica Newtoniana para el estudio dinámico de sistemas, como las ecuaciones de Newton-Euler (Leonhard Euler, 1707-1783) o la Mecánica Analítica Lagrangiana (Joseph-Louis Lagrange, 1736-1813), que permiten obtener las leyes del movimiento de un determinado sistema eximiendo del tratamiento de las fuerzas de reacción vinculares. Aún así, en este capítulo se obtendrán las ecuaciones de movimiento del péndulo invertido sobre dos ruedas de dos maneras diferentes, aplicando los teoremas de Newton que se enseñan en la asignatura de *Ampliación de Física* de 2º del grado en Tecnologías Industriales. Se mostrará también la obtención de las ecuaciones de la dinámica mediante los postulados de Lagrange. Se linealizarán las ecuaciones en torno al punto de equilibrio inestable, y se obtendrá el modelo linealizado en el espacio de estados.

2.1. Descripción del Sistema

La figura 2-1 muestra uno de los primeros prototipos realizados del péndulo invertido sobre dos ruedas en madera. El cuerpo del vehículo se asemeja a un péndulo invertido que se situa en posición de equilibrio inestable, lo que significa que, ante una pequeña perturbación que lo separe de la vertical, el sistema caerá movido por la gravedad, y realizará un *Movimiento Armónico Simple* en torno a su punto de equilibrio estable, a menos que se aplique la correcta velocidad/aceleración en los motores de las ruedas para mantenerlo en equilibrio.

El vehículo cuenta con dos motores independientes (uno en cada rueda) y, por tanto, permiten la aplicación de velocidades y aceleraciones diferentes en cada rueda. La pregunta ahora es ¿De qué manera depende el ángulo con la vertical del péndulo de la velocidad/aceleracíon en las ruedas?

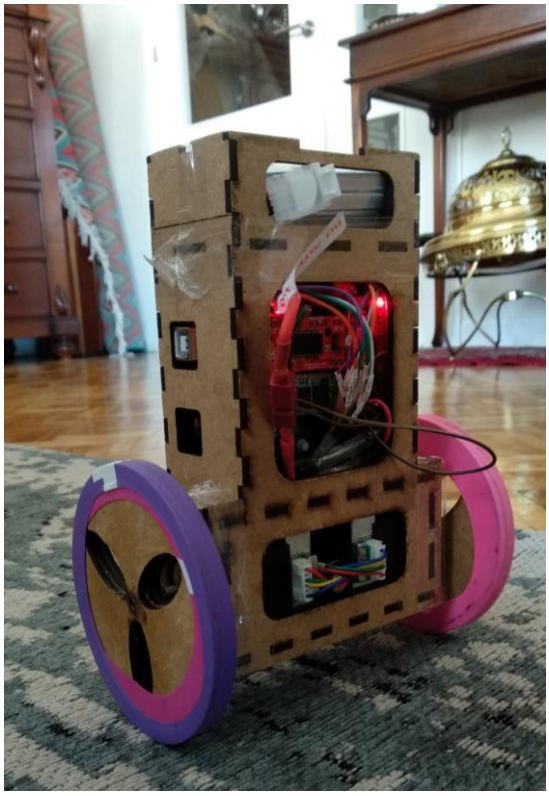


Figura 2-1. Primer prototipo del péndulo invertido sobre dos ruedas en madera

El estudio dinámico del sistema viene precisamente a mostrarnos la relación matemática, en forma de ecuaciones diferenciales, que hay entre el ángulo con la vertical y la velocidad/aceleracíon de cada una de las ruedas.

2.2 Suposiciones en el modelo

Por comodidad, se realizará el estudio dinámico para movimiento plano. Mediante experimentación con el sistema real, se estudiará la robustez del modelo para seguimiento de trayectorias curvas, viendo la dependencia con el radio de curvatura o con la celeridad de la base. Se estudiará hasta que punto es necesario la obtención de un modelo 3d del vehículo que tenga en cuenta las dinámicas no modeladas.

Otras suposiciones en el modelo son:

- No existe deslizamiento en el contacto suelo-rueda.
- El contacto suelo-rueda se produce a través de la recta tangente, es decir, no existe rodadura.
- Los diferentes solidos que forman el vehículo se consideran rígidos.
- Por consideraciones de simetría la inercia I_{YZ} expresada en los ejes del solido 2 se considera nula. Veanse los ejes en la figura 2-2.
- La distribución de masas en los diferentes solidos del sistema se considera uniforme.
- Se supone despreciable el ángulo de desviación del centro de masas con respecto del eje de simetría Z_2.
- El observador (sólido 1) se considera en reposo: sistema de referencia no inercial.

2.3 Aplicación del TCM y TMC basada en la desvinculación del cuerpo

Antes de proceder con la aplicación de los teoremas se enunciarán los mismos con objeto de tenerlos como referencia. Note el lector que sus expresiones matemáticas están particularizadas para un sólido rígido y no para un sistema de masas puntuales sin ligaduras de rigidez. Nótese también que los teoremas están particularizados para un sistema de referencia no inercial, por lo que se excluyen de la ecuación las fuerzas de inercia.

2.3.1 Enunciados de los dos teoremas

Teorema de la Cantidad de Movimiento TCM

La derivada respecto al tiempo de la cantidad de movimiento de un sistema de masas es igual al sumatorio de las fuerzas externas actuando sobre él. En otras palabras, si sobre un sistema de partículas no actúa ninguna fuerza externa, su cantidad de movimiento permanece constante.

Matemáticamente se expresa como:

$$\frac{d\vec{C}}{dt} = \frac{d(\iiint \rho\vec{v}\, dV)}{dt} = \frac{d(M_2\overrightarrow{v_{21}^G})}{dt} = \sum_{i=1}^{n} \overrightarrow{F_i^{ext}} \tag{2–1}$$

Siendo $\vec{C}$ la cantidad de movimiento, ρ la densidad de masa, dV un diferencial de volumen, M_2la masa total del sólido, $\overrightarrow{v_{21}^G}$ la velocidad del centro de masas del sólido 2 respecto del observador 1, y $\sum_{i=1}^{n} \overrightarrow{F_i^{ext}}$ el sumatorio de fuerzas externas actuando sobre el sólido 2.

Teorema del Momento Cinético TMC

La derivada con respecto al tiempo del momento cinético de un sistema de masas respecto de un punto geométrico A, es igual al sumatorio de los momentos provocados por las fuerzas externas al sistema respecto de A, más el producto vectorial entre la cantidad de movimiento y la velocidad del punto sobre el que se toman los momentos A.

Matemáticamente se expresa como:

$$\frac{d\overrightarrow{L_A}}{dt} = \sum_{i=1}^{n} \overrightarrow{M_{Ai}^{ext}} + \vec{C}x\dot{\overrightarrow{O_1A}} \tag{2–2}$$

$$\overrightarrow{L_A} = \overleftrightarrow{I_A}\overrightarrow{\omega_{21}} + M_2\overrightarrow{AG_2}x\overrightarrow{v_{21}^A} \tag{2–3}$$

Siendo $\overrightarrow{L_A}$ el momento cinético respecto del punto geométrico A, $\dot{\overrightarrow{O_1A}}$ la velocidad de ese punto geométrico respecto del observador 1, $\sum_{i=1}^{n} \overrightarrow{M_{Ai}^{ext}}$el sumatorio de momentos externos respecto del punto A actuando sobre el sistema, y $\overleftrightarrow{I_A}$ la matriz de inercia calculada en el punto A.

Estos teoremas acaban de ser comentados de una forma muy escueta, mostrando simplemente los resultados matemáticos finales. Por si el lector desea entender y comprender mejor como se llega a los anteriores resultados, se añadirá al final del presente documento un anexo que muestre cómo se deducen los anteriores enunciados. Para ello, haremos uso de los apuntes de Ampliación de Física de los profesores Dº. Enrique Drake Moyano y Dº. Manuel Toscano Jiménez.

2.3.2 Cinemática del movimiento

Antes de aplicar los teoremas, se hace necesario estudiar la cinemática del sistema, ya que posteriormente será necesario el conocimiento de diversas velocidades y aceleraciones. Así pues, procedemos con el estudio de la cinemática del movimiento ayudados de la figura 2-2.

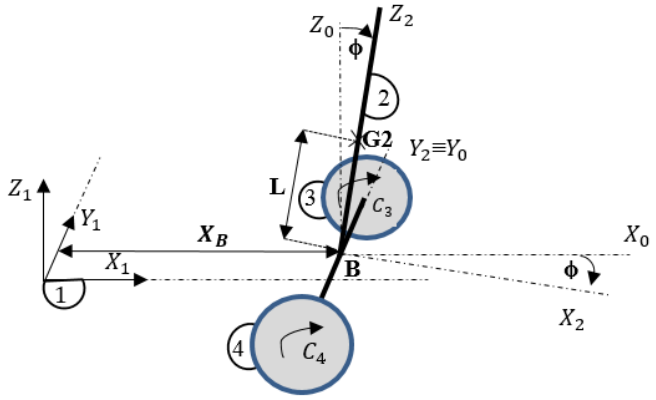


Figura 2-2. Ilustración gráfica de los parámetros del modelo dinámico.

El vector de posición del centro de masas del solido 2 respecto del observador 1 es:

$$\overrightarrow{r_{21}^{G}} = (X_B + L\sin\phi)\overrightarrow{\imath_1} + L\cos\phi\,\overrightarrow{J_1} \tag{2–4}$$

La derivada hecha por el observador 1 con respecto al tiempo una y otra vez nos da la velocidad y aceleración del centro de masas:

$$\overrightarrow{v_{21}^{G}} = \left(\dot{X}_B + L\dot{\phi}\cos\phi\right)\overrightarrow{\imath_1} - L\,\dot{\phi}\sin\phi\,\overrightarrow{J_1} \tag{2–5}$$

$$\overrightarrow{a_{21}^{G}} = \left(\ddot{X}_B + L\ddot{\phi}\cos\phi - L\dot{\phi}^2\sin\phi\right)\overrightarrow{\imath_1} - (L\ddot{\phi}\sin\phi + L\dot{\phi}^2\cos\phi)\overrightarrow{J_1} \tag{2–6}$$

Nótese que para el observador 1, los vectores de su base son constantes, por lo que su derivación da como resultado el vector nulo.

Para el punto B de la base se tiene que:

$$\overrightarrow{r_{21}^{B}} = X_B\overrightarrow{\imath_1} \tag{2–7}$$

$$\overrightarrow{v_{21}^{B}} = \dot{X}_B \overrightarrow{\imath_1} \tag{2–8}$$

$$\overrightarrow{a_{21}^{B}} = \ddot{X}_B \overrightarrow{\imath_1} \tag{2–9}$$

Para los centros de las ruedas:

$$\overrightarrow{v_{21}^{C4}} = \overrightarrow{v_{41}^{C4}} = \omega_{41} R \overrightarrow{\imath_1} \tag{2–10}$$

$$\overrightarrow{v_{21}^{C3}} = \overrightarrow{v_{31}^{C3}} = \omega_{31} R \overrightarrow{\imath_1} \tag{2–11}$$

Véase que los centros de las ruedas C3 y C4, pertenecientes a los solidos 3 y 4, poseen la misma velocidad que los puntos C3 y C4 pertenecientes al solido 2. Los sentidos de las velocidades angulares son los que se muestran en la figura 2-2.

Haciendo uso de la ecuación del campo de velocidades de un sólido rígido se deduce la expresión de la componente en Z de la velocidad angular del sólido 2 respecto a 1:

$$\overrightarrow{v_{21}^{C3}} = \overrightarrow{v_{21}^{C4}} + \overrightarrow{\omega_{21}} x \overrightarrow{C_4 C_3} \tag{2–12}$$

$$\overrightarrow{\omega_{21}} . \overrightarrow{k_1} = \omega_{21}^{Z_1} = \frac{(\omega_{41} - \omega_{31}) R}{b} \tag{2–13}$$

Siendo b la distancia entre C3 y C4.

Como se verá posteriormente, $\omega_{21}^{Z_1}$ no se utiliza en la aplicación de los teoremas del TCM y TMC, sin embargo, su conocimiento puede ser muy útil si se quieren utilizar técnicas de estimación de posición como la conocida Odometría[1].

2.3.3 Desvinculación del solido 2

Principio de Liberación

Todo sistema material o sistema de puntos materiales sometido a vínculos puede ser tratado como si estuviese libre de los mismos si se sustituyen dichos vínculos por las denominadas fuerzas de reacción vincular $\overrightarrow{\phi_k}$

Atendiendo al principio de liberación de solidos sometidos a ligaduras, desvinculamos el solido 2 del resto de solidos añadiendo las correspondientes fuerzas de reacción vincular, las cuales cumplen las mismas funciones que los vínculos sustituidos.

[1] La Odometría es una técnica de estimación de posición ampliamente utilizada en el mundo de la robótica. Particularizando para nuestro vehículo, consiste en la integración de la velocidad angular de las ruedas con objeto de estimar la posición.

Estudiando el movimiento relativo de los solidos 3 y 4, con respecto al sólido 2, observamos que la velocidad relativa de los centros de las ruedas C3 y C4 son $\vec{0}$.

$$\overrightarrow{v_{32}^{C3}} = \vec{0} \tag{2–14}$$

$$\overrightarrow{v_{42}^{C4}} = \vec{0} \tag{2–15}$$

Por tanto, existen 2 fuerzas de reacción vincular $\overrightarrow{\phi_{32}}$ y $\overrightarrow{\phi_{42}}$, cada una con sus tres componentes desconocidas.

El uso del *Principio de Liberación* también es de aplicación para las restricciones en las rotaciones, añadiendo los correspondientes momentos incógnita en las direcciones correspondientes.

Estudiando las rotaciones relativas entre los sólidos 2-3 y 2-4, se tiene que:

$$\overrightarrow{\omega_{32}} = \begin{bmatrix} 0 \\ \omega_{32}^{y} \\ 0 \end{bmatrix} \tag{2–16}$$

$$\overrightarrow{\omega_{42}} = \begin{bmatrix} 0 \\ \omega_{42}^{y} \\ 0 \end{bmatrix} \tag{2–17}$$

Dado que la rotación de los solidos 3 y 4 relativas al solido 2 es nula en las direcciones X_0 y Z_0, hay que añadir los correspondientes momentos de reacción vincular.

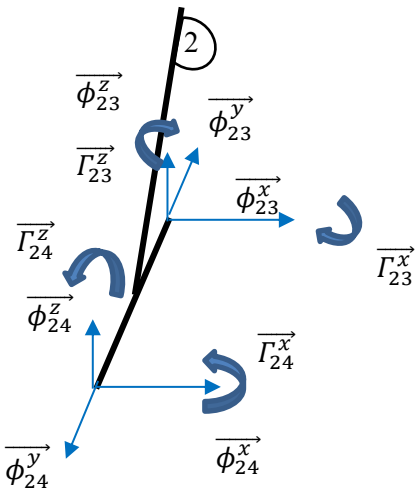


Figura 2-3. Desvinculación del solido 2.

2.3.4 Aplicación TMC a 2 en B y dirección k1

Una vez desvinculado el sólido 2, estamos en disposición de aplicar los teoremas.

$$\overrightarrow{L_B} = \overleftrightarrow{I_B}\overrightarrow{\omega_{21}} + M_2\overrightarrow{BG_2}x\overrightarrow{v_{21}^B} \tag{2–18}$$

$$\overrightarrow{L_B} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\phi} \\ 0 \end{bmatrix} + M_2 \begin{vmatrix} \overrightarrow{\imath_1} & \overrightarrow{J_1} & \overrightarrow{k_1} \\ L\sin\phi & 0 & l\cos\phi \\ \dot{X_B} & 0 & 0 \end{vmatrix} \tag{2–19}$$

Proyectando en la dirección $\overrightarrow{k_1}$:

$$\overrightarrow{L_B}.\overrightarrow{k_1} = \dot{\phi} I_{yz} \tag{2–20}$$

Como se comentó en las suposiones del modelo, la simetría del vehículo hace que la inercia I_{yz} sea cero, por tanto:

$$\overrightarrow{L_B}.\overrightarrow{k_1} = \dot{\phi} I_{yz} = 0 \tag{2–21}$$

Derivando respecto al tiempo y proyectando sobre $\overrightarrow{k_1}$ se tiene:

$$\dot{\overrightarrow{L_B}}.\overrightarrow{k_1} = 0 \tag{2–22}$$

Aplicando el TMC en la dirección $\overrightarrow{k_1}$ (véanse sentidos vectoriales en la figura 2-3) se obtiene:

$$\dot{\overrightarrow{L_B}}.\overrightarrow{k_1} = 0 = \underbrace{\Gamma_{24}^z - \Gamma_{23}^z}_{\text{=0, por simetría}} - \phi_{23}^x \frac{b}{2} + \phi_{24}^x \frac{b}{2} \tag{2–23}$$

Por tanto, de este razonamiento se deduce la igualdad de las componentes en X_1 de las fuerzas vinculares en los puntos C3 y C4.

$$\phi_{23=}^x \phi_{24}^x \tag{2–24}$$

2.3.5 TCM al solido 2

Ayudandonos de la expresión previamente cáculada de la aceleración del centro de masas (2–6), la aplicación del TCM en la dirección X_1 da como resultado:

$$M_2\left(\ddot{X_B} + L\ddot{\phi}\cos\phi - L\dot{\phi}^2\sin\phi\right) = \phi_{23}^x + \phi_{24}^x \tag{2–25}$$

De (2–24) y de (2–25) se deduce que:

$$\phi_{24}^{x} = \phi_{23}^{x} = \frac{M_2}{2}\left(\ddot{X}_B + L\ddot{\phi}\cos\phi - L\dot{\phi}^2\sin\phi\right) \tag{2–26}$$

2.3.6 Desvinculación de la rueda y aplicación TCM

Para las dos ruedas se llega al mismo resultado, por lo que, para no extendernos demasiado, trabajaremos únicamente con la rueda derecha (sólido 4).

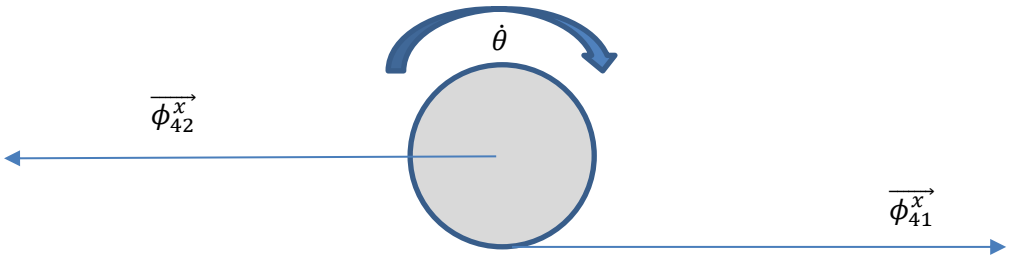


Figura 2-4. Desvinculación del sólido 4.

Aplicando el TCM se obtiene:

$$m_{rueda}\ddot{X}_B = \phi_{41}^{x} - \phi_{42}^{x} \tag{2–27}$$

Obsérvese que, dado que se trata de un movimiento unidireccional en dirección X_1, se tiene que la velocidad del punto B es la misma que la de los centros de las ruedas. Por tanto, y haciendo uso de (2–26), se tiene que:

$$\phi_{41}^{x} = \left(m_{rueda} + \frac{M_2}{2}\right)\ddot{X}_B + \frac{M_2}{2}\left(L\ddot{\phi}\cos\phi - L\dot{\phi}^2\sin\phi\right) \tag{2–28}$$

Como dijimos, por simetría tenemos:

$$\phi_{31}^{x} = \left(m_{rueda} + \frac{M_2}{2}\right)\ddot{X}_B + \frac{M_2}{2}\left(L\ddot{\phi}\cos\phi - L\dot{\phi}^2\sin\phi\right) \tag{2–29}$$

2.3.7 TMC aplicado a la rueda

De nuevo, y con objeto de no extendernos, aplicaremos el TMC solo al sólido 4. Los momentos se tomarán respecto al centro de la rueda C4.

Procediendo obtenemos:

$$I_r\ddot{\theta} = \Gamma_M - \phi_{41}^x R \tag{2–30}$$

Siendo Γ_M el par mecánico que ejerce el motor de la rueda.

Haciendo uso del resultado obtenido en (2–28) obtenemos:

$$\Gamma_M = \ddot{\theta}\left\{I_r + \left(m_{rueda} + \frac{M_2}{2}\right)R^2\right\} + \frac{M_2 R}{2}\left(L\ddot{\phi}\cos\phi - L\dot{\phi}^2\sin\phi\right) \tag{2–31}$$

2.3.8 TMC aplicado al sólido 2 respecto de B

Para acabar por fin con la obtención de la ecuación diferencial que muestra la dependencia del ángulo con la vertical con la aceleración angular de las ruedas, procedemos a aplicar el TMC al sólido 2 en el punto B.

Procediendo obtenemos:

$$I_{yy}\ddot{\phi} + M_2\ddot{X}_B L\cos\phi - M_2\dot{X}_B\dot{\phi}L\sin\phi = -2\Gamma_M + M_2 gL\sin\phi - M_2\dot{X}_B\dot{\phi}L\sin\phi \tag{2–32}$$

Obsérvese que el par motor Γ_M ha sido multiplicado por dos, ya que son 2 ruedas.

Haciendo uso de (2–31), operando y simplificando, obtenemos:

$$\left(I_{yy} + M_2 RL\cos\phi\right)\ddot{\phi} + \{R^2(3m_r + M_2) + M_2 RL\cos\phi\}\ddot{\theta} - M_2 RL\dot{\phi}^2\sin\phi - M_2 gL\sin\phi = 0$$

(2–33)

La ecuación (2–33) refleja la relación matemática existente entre la aceleración angular de las ruedas y el ángulo con la vertical, obtenidas a partir de dos principios matemáticos del siglo XVII. En el siguiente punto llegaremos al mismo resultado matemático mediante otro camino.

2.4 Aplicación del TCM y TMC basada en la desvinculación del vehículo completo

En este punto procederemos de la misma manera que el punto anterior, es decir, aplicaremos los teoremas para la obtención de las leyes del movimiento. La diferencia esta en que llegaremos al mismo resultado por otro camino, eligiendo otros sistemas de sólidos para aplicar el TCM y TMC.

2.4.1 TMC al sistema 2 U 3 U 4

Aplicaremos el *Teorema del Momento Cinético* respecto del punto B al sistema formado por la unión de los solidos 2,3 y 4.

$$\overrightarrow{L_B} = \sum_{\forall i} \overrightarrow{BP_i}\, xm_i \overrightarrow{v_i} = \sum_{\forall i}^{sólido2} \overrightarrow{BP_i} xm_i \overrightarrow{v_i} + \underbrace{\sum_{\forall i}^{sólido3} \overrightarrow{BP_i} xm_i \overrightarrow{v_i} + \sum_{\forall i}^{sólido3} \overrightarrow{BP_i} xm_i \overrightarrow{v_i}}_{\text{=0, por simetría}} \tag{2–34}$$

El momento cinético respecto de B del sólido 2 ya fue calculado en (2-20)

$$\overrightarrow{L_B}.\overrightarrow{k_1} = \dot{\phi} I_{yz} = 0 \tag{2–35}$$

Al igual que vimos en (2-21), la inercia I_{yz} es cero por la forma del vehículo.

$$\dot{\overrightarrow{L_B}}.\overrightarrow{k_1} = 0 \tag{2–36}$$

Como se puede ver en la siguiente figura, son las componentes en X_1 de las fuerzas de reacción vincular en los puntos de contacto rueda-suelo las que producen momento en la dirección $\overrightarrow{k_1}$. En este caso no existen momentos vinculares ejercidos por el sólido 1 que prohíban la rotación de las ruedas.

De la aplicación del teorema resulta:

$$\overrightarrow{\phi_{41}^{x}} = \overrightarrow{\phi_{31}^{x}} \tag{2–37}$$

Este resultado matemático nos dice algo totalmente lógico: para movimiento plano se hace necesario la igualdad de dichas componentes de tal manera que el sistema no tienda a rotar sobre el eje OZ_1.

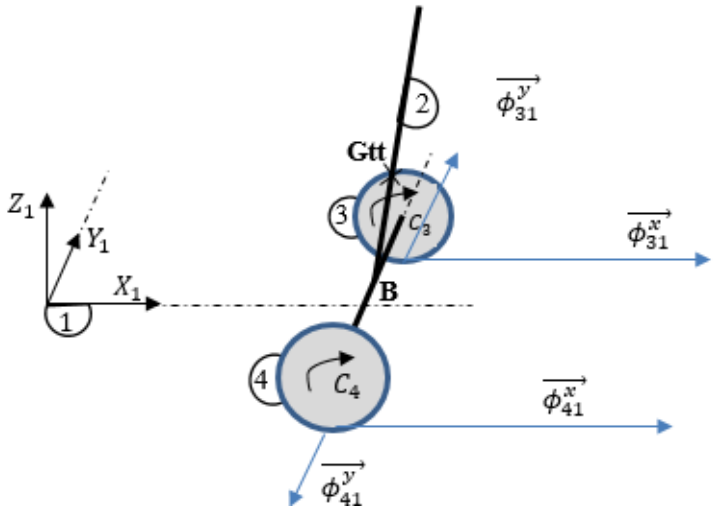


Figura 2-5. Desvinculación sistema 2U3U4

2.4.2 TCM al sistema 2 U 3 U 4

Antes de aplicar *el Teorema de la Cantidad de Movimiento* hay que mencionar algo sumamente importante: el centro de masas del sistema 2U3U4, al cual llamaremos G_{TT}, no tiene nada que ver con el centro de masas del sólido 2, G_2. El centro de masas G_2 se situaba a una distancia L de la base medida sobre el eje OZ_2 y, en este caso, el centro de masas G_{TT} se sitúa a una distancia L´. Los valores de L y L´no coinciden, aunque como se verá, sus valores están relacionados y cumplen con la desigualdad de que L´es menor que L.

La aplicación de TCM da como resultado:

$$(M_2 + 2m_{rueda})\left(\ddot{X}_B + L'\ddot{\phi}\cos\phi - L'\dot{\phi}^2\sin\phi\right) = \phi_{31}^{x} + \phi_{41}^{x} = 2\phi_x \tag{2–38}$$

Como se ve en (2-38), la expresión de la aceleración del centro de masas es análoga a la de G_2, sin más que sustituir L´ por L.

2.4.3 Aplicación TMC al sólido 4

Los resultados de la aplicación de el *Teorema del Momento Cinético* al sólido 4 son equivalentes a la aplicación de los mismos sobre el sólido 3, por lo que se aplicará únicamente al sólido 4.

Ayudandonos de la figura 2-4 obtenemos:

$$I_r\ddot{\theta} = \Gamma_M - \phi_{41}^{x}R \tag{2–39}$$

Combinando (2–39) y (2–38) se concluye que:

$$\Gamma_M = I_r\ddot{\theta} + \frac{R}{2}(M_2 + 2m_{rueda})\left(\ddot{X}_B + L'\ddot{\phi}\cos\phi - L'\dot{\phi}^2\sin\phi\right) \tag{2–40}$$

2.4.4 Aplicación TMC al sólido 2

Tal como se dedució en (2-32), la expresión de la aplicación del teorema da como resultado:

$$I_{yy}\ddot{\phi} + M_2\ddot{X}_B L'\cos\phi - M_2\dot{X}_B\dot{\phi}L'\sin\phi = -2\Gamma_M + M_2 g L'\sin\phi - M_2\dot{X}_B\dot{\phi}L'\sin\phi \tag{2–41}$$

Donde, sustitituyendo la expression de Γ_M de (2–40) se obtiene:

$$\left(I_{yy} + (M_2 + 2m_{rueda})RL'\cos\phi\right)\ddot{\phi} + \{2I_r + M_2RL'\cos\phi + R^2(M_2 + 2m_{rueda})\}\ddot{\theta} \ldots$$
$$\ldots - M_2RL'\dot{\phi}^2\sin\phi - M_2gL'\sin\phi = 0$$

(2–42)

Aproximando I_r por $\frac{1}{2}m_{rueda}R^2$ obtenemos:

$$\left(I_{yy} + (M_2 + 2m_{rueda})RL'\cos\phi\right)\ddot{\phi} + \{R^2(M_2 + 3m_{rueda}) + M_2RL'\cos\phi\}\ddot{\theta} \ldots$$
$$\ldots - M_2RL'\dot{\phi}^2\sin\phi - M_2gL'\sin\phi = 0$$

(2–43)

Atendiendo a la relación entre L y L´, obtenida de la definición de centro de masas de un sistema de sólidos rígidos, se obtiene:

$$(M_2 + 2m_{rueda})L' = 0m_3 + 0m_4 + LM_2 \tag{2–44}$$

Aplicando (2–44) en (2-43), obtenemos la ecuación diferencial que relaciona el ángulo con la vetical del sólido 2 con la aceleración angular de las ruedas. Se puede comprobar que es idéntica a la expresión (2-33) obtenida de la forma 1.

$$\left(I_{yy} + M_2RL\cos\phi\right)\ddot{\phi} + \{R^2(3m_r + M_2) + M_2RL\cos\phi\}\ddot{\theta} - M_2RL\dot{\phi}^2\sin\phi - M_2gL\sin\phi = 0$$

(2-45)

Resulta fascinante obtener iguales resultados matemáticos de diferentes maneras con una correcta aplicación de los Teoremas de Newton.

2.5 Aplicación de las ecuaciones de Lagrange

En este apartado se llevará a cabo la obtención de la ecuación (2-45) a través de la mecánica Lagrangiana.

Cabe decir que la siguiente deducción ha sido tomada del trabajo de Fin de Grado de Dª. Cecilia González Gonzalez, trabajo titulado *Mejora del Software de un vehículo autoequilibrado de tipo péndulo invertido de bajo coste*, publicado en 2016. Como nota, recordar que se supone una desviación nula entre el centro de masas y el eje de simetría. Para los lectores curiosos, decir que en el trabajo de Cecilia se muestra la deducción matemática suponiendo una desviación no nula.

2.5.1 Expresión de la Lagrangiana

Matemáticamente, la Lagrangiana de un sistema se define como

$$\mathcal{L} = T_{tras} + T_{rot} - V \tag{2–46}$$

Siendo V la energía potencial del sistema y T la energía cinética, descompuesta como la suma de las energías cinéticas de traslación y rotación.

Las expresiones de los anteriores términos, puestos en función de los parámetros del modelo de la figura 2-2, son las siguientes:

$$T_{tras} = \left(m_{rueda} + \frac{M_2}{2}\right) R^2\dot{\theta}^2 + \frac{M_2 L^2 \dot{\phi}^2}{2} + RLM_2\dot{\theta}\dot{\phi}\cos\phi \tag{2–47}$$

$$T_{rot} = 2\frac{1}{2} I_r \dot{\theta}^2 + \frac{1}{2} I_{yy}\dot{\phi}^2 \tag{2–48}$$

$$V = M_2 g L \cos\phi \tag{2–49}$$

2.5.2 Ecuaciones de Lagrange

Sin objeto de saber su deducción, las ecuaciones de Lagrange particularizadas para nuestro sistema adquieren la forma:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\phi}}\right) - \frac{\partial \mathcal{L}}{\partial \phi} = -\Gamma_M \tag{2–50}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = \Gamma_M \tag{2–51}$$

Calculando la expresión de los diferentes términos de las ecuaciones (2–50) y (2-51) obtenemos:

$$\frac{\partial \mathfrak{L}}{\partial \dot{\phi}} = 2M_2L^2\dot{\phi} + M_2RL\dot{\theta}\cos\phi \tag{2–52}$$

$$\frac{d}{dt}\left(\frac{\partial \mathfrak{L}}{\partial \dot{\phi}}\right) = 2M_2L^2\ddot{\phi} + M_2RL\ddot{\theta}\cos\phi - M_2RL\dot{\theta}\dot{\phi}\sin\phi \tag{2–53}$$

$$\frac{\partial \mathfrak{L}}{\partial \phi} = -M_2RL\dot{\theta}\dot{\phi}\sin\phi + M_2gL\sin\phi \tag{2–54}$$

$$\frac{\partial \mathfrak{L}}{\partial \dot{\theta}} = 2\left(m_{rueda} + \frac{M_2}{2}\right)R^2\dot{\theta} + m_{rueda}R^2\dot{\theta} + M_2RL\dot{\phi}\cos\phi \tag{2–55}$$

$$\frac{d}{dt}\left(\frac{\partial \mathfrak{L}}{\partial \dot{\phi}}\right) = 2\left(m_{rueda} + \frac{M_2}{2}\right)R^2\ddot{\theta} + m_{rueda}R^2\ddot{\theta} + M_2RL\ddot{\phi}\cos\phi - M_2RL\dot{\phi}^2\sin\phi \tag{2–56}$$

$$\frac{\partial \mathfrak{L}}{\partial \theta} = 0 \tag{2–57}$$

Donde se han sustituido los valores de I_{yy} e I_r por sus expresiones aproximadas:

$$I_r \simeq \frac{1}{2}m_{rueda}R^2 \tag{2–58}$$

$$I_{yy} \simeq 2M_2L^2 \tag{2–59}$$

Nótese que la inercia a la rotación de la rueda esta tomada respecto a unos ejes que pasan por B y no por el centro de masas, por lo que, por el *Teorema de Steiner,* se ha tenido que sumar M_2L^2, ocasionando que la expresión tenga un factor 2.

2.5.3 Obtención ecuación de movimiento

Sustituyendo las ecuaciones (2-52) (2-53) (2-54) (2-55) (2-56) (2-57) en (2–50) (2–51) obtenemos:

$$(2M_2L^2 + M_2RL\cos\phi)\ddot{\phi} + \{R^2(3m_r + M_2) + M_2RL\cos\phi\}\ddot{\theta} - M_2RL\dot{\phi}^2\sin\phi - M_2gL\sin\phi = 0 \tag{2–60}$$

La anterior expresión es completamente análoga a la ecuación (2-45)

$$\left(I_{yy} + M_2RL\cos\phi\right)\ddot{\phi} + \{R^2(3m_r + M_2) + M_2RL\cos\phi\}\ddot{\theta} - M_2RL\dot{\phi}^2\sin\phi - M_2gL\sin\phi = 0 \tag{2–61}$$

La expresión (2-61) constituye la ley por la que se rige el movimiento del péndulo. Se puede comprobar que se ha llegado al mismo resultado que mediante la Dinámica Vectorial de Newton.

2.6 Linealización del modelo

Para el desarrollo de controladores lineales en variables de estado se hace necesaria la obtención de un modelo lineal mediante una aproximación de Taylor en torno al punto de equilibrio inestable de la dinámica expresada por la ecuacíon (2-61).

Para ello, analizando la dinámica del sistema, se deduce que:

$$\ddot{\theta}_e = 0 \tag{2–62}$$

$$M_2 g L \sin\phi = 0 \tag{2–63}$$

La ecuación (2-63) nos dice que existen dos puntos de equilibrio, uno inestable y otro estable, con valores 0 y pi respectivamente.

$$\phi_e^1 = 0 \tag{2–64}$$

$$\phi_e^2 = \pi \tag{2–65}$$

Obviamente, es el punto de equilibrio inestable el que nos interesa.

Dado que, en el punto de equilibrio, la aceleración angular de las ruedas es nula, la velocidad de las mismas es igual a una constante. El valor de esa constante es un parámetro libre, por lo que es posible controlar al sistema de tal manera que siga referencias en velocidad. Por tanto, el punto de equilibrio esta definido por

$$\begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \\ \phi \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\theta}_e \\ 0 \end{bmatrix} \tag{2–66}$$

Una vez conocido el punto de equilibrio, procedemos con la linealización de la ecuacíon diferencial (2-61) en torno a dicho punto. La función de Taylor F tiene la forma mostrada en (2-67)

$$F(\phi, \dot{\phi}, \ddot{\phi}, \ddot{\theta}) = (I_{yy} + M_2 R L \cos\phi)\ddot{\phi} + \{R^2(3m_r + M_2) + M_2 R L \cos\phi\}\ddot{\theta} - M_2 R L \dot{\phi}^2 \sin\phi - M_2 g L \sin\phi = 0 \tag{2–67}$$

Su aproximación en torno al punto de equilibrio tiene la forma:

$$F(\phi, \dot{\phi}, \ddot{\phi}, \ddot{\theta}) \simeq F(0,0,0,0) + \frac{\partial F}{\partial \phi}(\phi - 0) + \frac{\partial F}{\partial \dot{\phi}}(\dot{\phi} - 0) + \frac{\partial F}{\partial \ddot{\phi}}(\ddot{\phi} - 0) + \frac{\partial F}{\partial \ddot{\theta}}(\ddot{\theta} - 0) \tag{2–68}$$

Estando las derivadas parciales evaluadas en el punto de equilibrio.

Calculando las expresiones de las anteriores derivadas parciales:

$$\frac{\partial F}{\partial \phi} = -M_2 R L \ddot{\phi} \sin\phi - M_2 R L \ddot{\theta} \sin\phi - M_2 R L \dot{\phi}^2 \cos\phi - M_2 g L \cos\phi \overset{p.e}{\Rightarrow} -M_2 g L \tag{2–69}$$

$$\frac{\partial F}{\partial \dot{\phi}} = -2M_2RL\dot{\phi}\sin\phi \overset{p.e}{\Rightarrow} 0 \tag{2–70}$$

$$\frac{\partial F}{\partial \ddot{\phi}} = \left(I_{yy} + M_2RL\cos\phi\right) \overset{p.e}{\Rightarrow} \left(I_{yy} + M_2RL\right) \tag{2–71}$$

$$\frac{\partial F}{\partial \ddot{\theta}} = \{R^2(3m_r + M_2) + M_2RL\cos\phi\} \overset{p.e}{\Rightarrow} \{R^2(3m_r + M_2) + M_2RL\} \tag{2–72}$$

La expresión final del modelo linealizado se muestra en (2-73)

$$\left(I_{yy} + M_2RL\right)\ddot{\phi} + \{R^2(3m_r + M_2) + M_2RL\}\ddot{\theta} - M_2gL\phi = 0 \tag{2–73}$$

2.7 Obtención del modelo linealizado en el espacio de estados

La expresión general de la dinámica de un sistema lineal mediante el espacio de estados en forma continua tiene la siguiente forma

$$\dot{x} = Ax + Bu + C\omega \tag{2–74}$$

donde $C\omega$ representa los errores de modelado y las perturbaciones sobre el sistema.

Particularizando la expresión (2-74) para nuestro modelo linealizado (2-73) obtenemos:

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{M_2gL}{I_{yy} + M_2RL} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{R^2(3m_r + M_2) + M_2RL}{I_{yy} + M_2RL} \\ 1 \end{bmatrix} u \tag{2–75}$$

Donde se ha despreciado el término $C\omega$. Observesé que la entrada de control u es la aceleración de las ruedas $\ddot{\theta}$.

Con esto queda concluido el *Estudio Dinámico* del sistema.

3 IDENTIFICACIÓN DE PARÁMETROS

No importa lo hermosa que sea tu teoría, no importa lo inteligente que seas. Si no está de acuerdo con el experimento, está mal. En esa simple afirmación está la clave de la ciencia.

Richard Phillips Feynman

La anterior frase del físico norteamericano Richard P. Feynman quién, entre otras cosas, participó en el proyecto Manhattan de la 2º guerra mundial, viene a que nos hagamos preguntar acerca de la validez de nuestro modelo obtenido en el punto anterior. ¿Es válido nuestro modelo? ¿De verdad se cumple?

El propio Feynman nos da la clave en su frase. Debemos de tomar datos reales del sistema y ver si la predicción del modelo matemático se asemeja a los valores reales obtenidos del sistema mediante la experimentación con este. En el momento en que nuestro modelo matemático se vea validado por datos empíricos tendremos la certeza de que el modelo es correcto y, por tanto, podremos proceder al diseño de controladores basándonos en él.

3.1 Preparación del experimento

Una vez obtenido el programa de control en el capítulo 5, el cual implementa cualquier ley de control sobre nuestro vehículo, se procede a la sintonización mediante técnicas heuristicas[2] de dos controladores PID, uno para ángulo y otro para velocidad, de tal manera que sean capaces de controlar al vehículo. Si, controlando al sistema en torno a una referencia nula en velocidad, aumentamos la ponderación del término integral del PID (de esta manera lo empeoraremos), conseguiremos que nuestro vehículo realice el movimiento correcto (grandes oscilaciones).

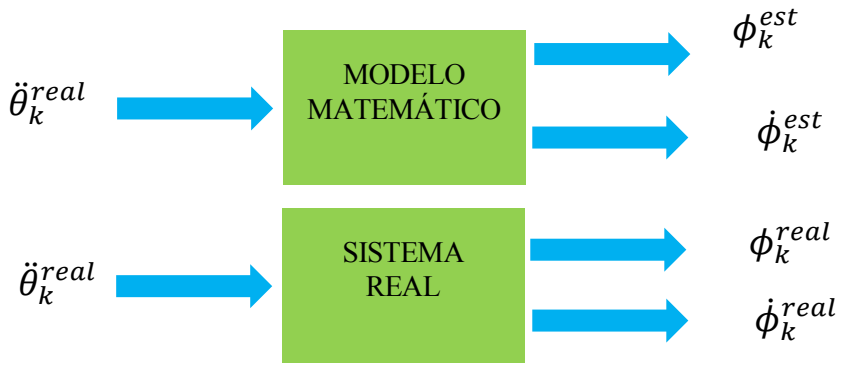


Figura 3-1. Diagrama esquemático de la estimación de parámetros.

[2] Con técnicas heurísticas se refiere a una sintonización por prueba y error, probando distintos valores para los parámetros de los PID y viendo cuales producen un mejor resultado en el control.

Para que la estimación de parámetros sea correcta, se hace necesario que el ruido intrinseco a la medida del ángulo con la vertical sea despreciable frente al propio ángulo ϕ. Por ello, como se dijo antes, se requieren oscilaciones en torno al punto de equilibrio con amplitudes grandes. Dicho de otra manera, si controlamos al sistema en torno a la vertical y obtenemos datos de dicho ángulo, sería muy difícil distinguir en la señal lo que es una pequeña oscilación de lo que es ruido en la medida, por lo que se hace necesario aumentar la amplitud de las oscilaciones.

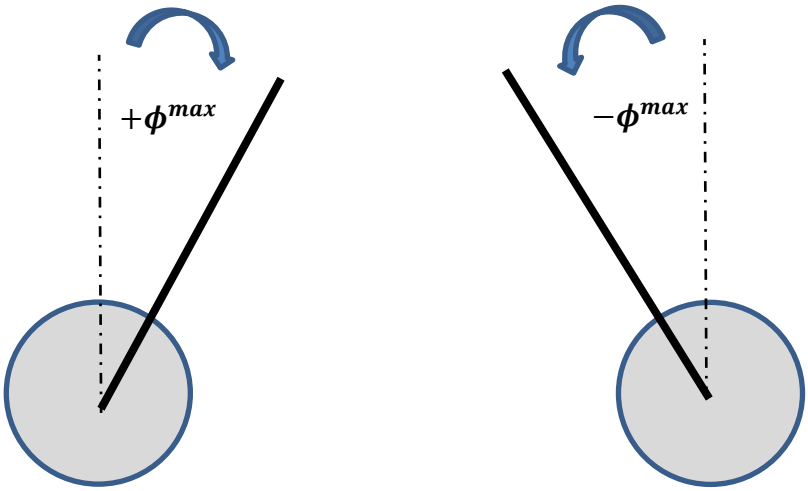


Figura 3-2. Esquema del movimiento del vehículo para la estimación.

La figura 3-1 muestra la forma de proceder en la estimación de parámetros que se llevará a cabo. Dejando al vehículo sobre el suelo, controlado con los PIDs que se comentarón anteriormente, se reciben los datos del movimiento, formados por las variables que se ven en el bloque *Sistema Real* de la figura.

Una vez hecho lo anterior, se excita a nuestro modelo matemático mediante la misma señal de aceleración angular de las ruedas que se obtuvo del sistema real y, mediante integración numérica, se reconstruyen las trayectorias del ángulo y velocidad angular.

Para medir la calidad de un modelo, y con objeto de establecer comparaciones entre varios de ellos, se define el funcional J como:

$$J = \sum_{k=0}^{N_{mues}-1} \lambda^k \left(\dot{\phi}_k^{real} - \dot{\phi}_k^{est}\right)^2 \tag{3-1}$$

El parámetro λ, configurable con valores en torno a 0.99, sirve para penalizar más los errores al principio de la integración que los errores finales. Dado que, la integración del modelo diverge conforme aumenta el tiempo, debido a la integración de señales con ruido, errores de modelado causados por las dinámicas no modeladas (suposiciones del modelo), errores en la medición de masas, etc, carece de sentido ponderar de la misma manera errores al principio que errores al final de la simulación.

En otras palabras, nuestro objetivo es encontrar el modelo matemático que prediga la evolución del sistema a corto plazo de la foma más óptima posible, entendiéndose por la forma más óptima la que minimice el valor del funcional J.

3.2 Estimación de parámetros

Como se recordará, la ecuación diferencial lineal del sistema tiene la forma dada en (3-2)

$$\left(I_{yy} + M_2RL\right)\ddot{\phi} + \{2I_r + R^2(2m_r + M_2) + M_2RL\}\ddot{\theta} - M_2gL\phi = 0 \tag{3-2}$$

Donde **no** se ha hecho la aproximación $I_r \simeq \frac{1}{2}m_{rueda}R^2$.

En ella, todos los parámetros son medibles fácilmente, excepto L , I_{yy} e I_r ,cuya obtención es más complicada.

Procediendo con la medida de los parámetros más facilmente medibles obtenemos, mediante el uso de pesos y reglas de medida, los siguientes valores:

Tabla 3-1. Valores de los parámetros del modelo

Parámetro	Valor
M_2	945g
R	5cm
m_r	65g
g	9.81ms^{-2}

3.2.1 Obtención de L

La figura 3-3 muestra cómo se ha obtenido la distancia *L* haciendo uso del *Teorema de la Momento Cinético* aplicado al vehículo completo. El experimento consiste en colgar al vehículo de un hilo muy fino agarrado a la parte superior (concretamente en una de las ranuras destinadas a la chavetas) y esperar a que se quede totalmente estático. Aplicando el TMC respecto a cualquier punto del hilo, se deduce que para que el vehículo se mantenga estático, es condición necesaria que el centro de masas se encuentre alineado con el hilo.

Cuando se observa el vehículo ya montado, se ve que está bastante balanceado, lo que quiere decir que la hipótesis de una desviación despreciable entre el centro de masas y el eje de simetría es correcta. Por tanto, tal y como se muestra en la figura 3-3, se obtiene la posición del centro de masas calculando la intersección entre ambas rectas.

Es muy importante decir que la distancia cálculada no es *L*, sino *L´*, dado que se trata del centro de masas del vehículo completo (Las ruedas están montadas en el chasis). Para obtener L se puede proceder de dos formas:

1. Quitar las ruedas al vehículo y volver a hacer el procedimiento de colgarlo y hallar el nuevo punto de corte entre las dos rectas.
2. Aplicar la relación vista en (2-44), la cual relaciona L y L´.

Si procedemos de la segunda forma, y teniendo en cuenta que la masa de las ruedas, por ser de madera, son totalmente despreciables frente a la masa del cuerpo, por lo que la aproximación $L \simeq L'$ es correcta.

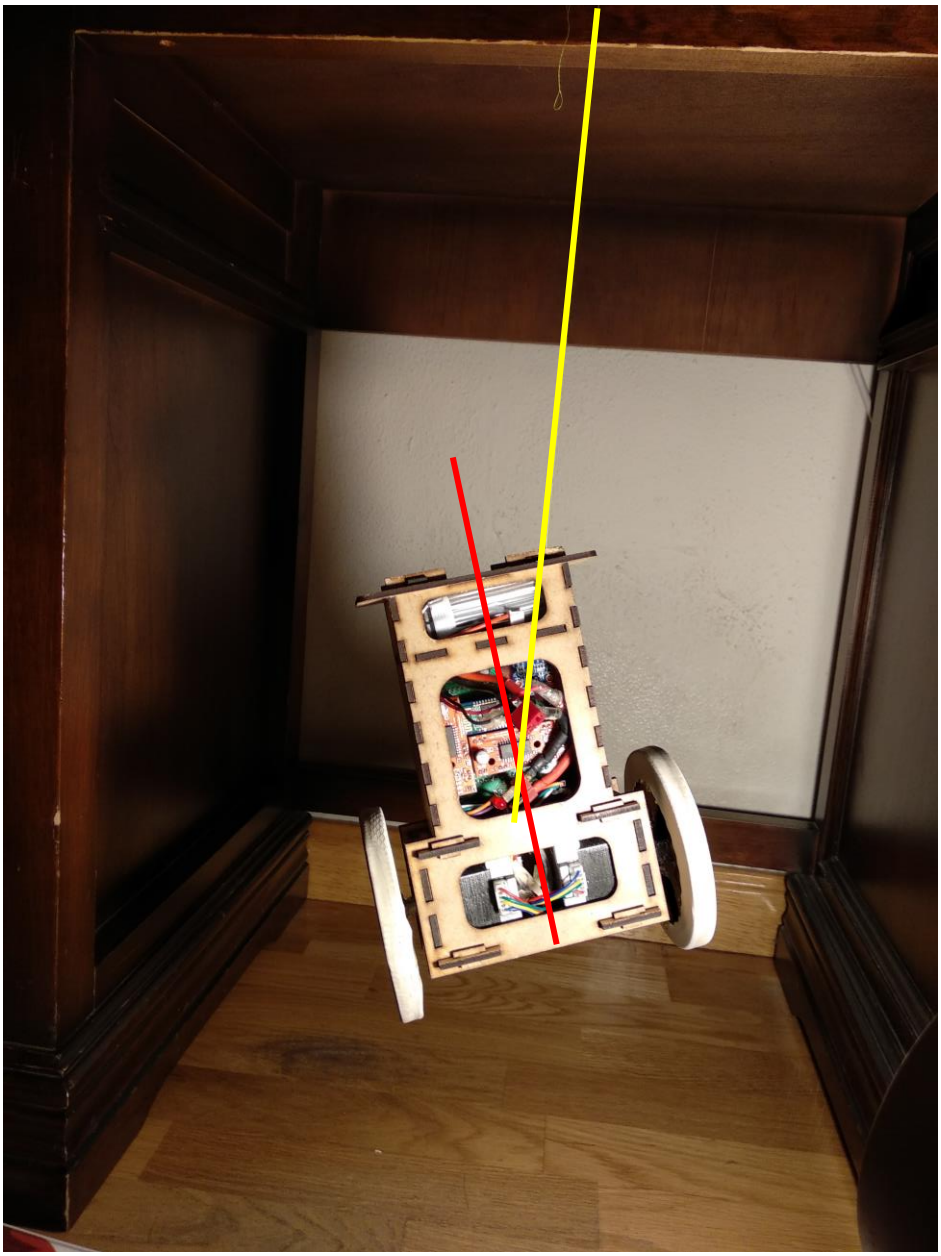


Figura 3-3. Centro de masas mediante estática de sólidos.

Nota: se ha superpuesto una linea amarilla sobre el hilo en la figura 3-3, por lo que el hilo se encuentra tras dicha linea.

Midiendo se obtiene que $L \simeq 5cm$.

3.2.2 Obtención de Iyy e Ir

El objetivo será encontrar el par de valores $\left(I_r^{opt}, I_{yy}^{opt}\right)$ que minimicen el valor del funcional J de la expresión (3-1). En otras palabras, se llevará a cabo la resolución de un problema de optimización que calcule el par de parámetros $\left(I_r^{opt}, I_{yy}^{opt}\right)$ que, incorporados al modelo dado por la ecuación diferencial (3-2), y tras la integración numérica de la misma, hagan que el valor del funcional J sea mínimo. Se trata de un problema de optimización con restricciones que tienen en cuenta la naturaleza acotada de los parámetros. Por ejemplo, como se verá más adelante, una restricción sobre los parámetros es que deben tomar valores positivos, en concordancia con la definición de los mismos.

En términos gráficos, el objetivo es encontrar los valores óptimos de $\left(I_r^{opt}, I_{yy}^{opt}\right)$ que hagan que las gráficas de $\dot{\phi}_k^{est}(t) = \dot{\phi}_k^{real}(t)$ se parezcan lo máximo posible al principio de la simulación.

3.2.2.1 Obtencion de datos del experimento

En la figura 3-4 que se muestra a continuación se puede observar la evolución temporal de las diferentes variables del modelo durante aproximadamente 8s. Nótese la gran amplitud de las oscilaciones del ángulo con la vertical, en torno a $\pm 25º$. Será con estos datos con los que trabajaremos para la obtención de los parámetros óptimos.

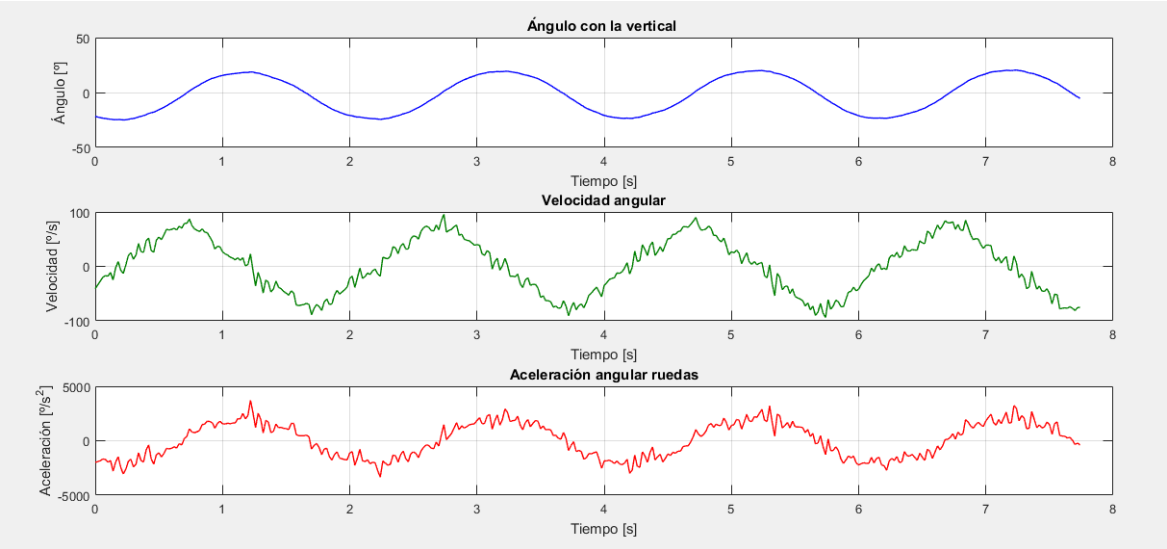


Figura 3-4. Datos obtenidos del experimento.

3.2.2.2 Programación del problema de optimización en Matlab

Para entender el código en Matlab, se explicarán brevemente las funciones programadas, de tal manera que se establezca una visión general del código. Para profundizar más en cómo han sido programadas, véanse los comentarios que se escriben dentro del código. Las funciones programadas se explicarán de abajo a arriba del código, ya que de esta manera es mucho más entendible. Dicho esto, procedemos.

1. *function [dX] = dinamic_two_wheeled_inverted_pendulum(~,x,u,params)*: En esta función esta programado el modelo dado por la ecuación (3-2) en el espacio de estados. Su tarea es devolver la derivada del vector de estados en un tiempo Tk. Recibe los parámetros a optimizar del modelo (I_r, I_{yy}), la acción de control y el valor del vector de estados en un tiempo Tk.

2. *function [y,T] = Runge_Kutta_orden_4(intervalo,u,xo,params)*: Su tarea es usar la función 1 para integrar la ecuación diferencial del modelo en **un solo** intervalo de tiempo *intervalo* de tamaño Tm=0.02s (Tiempo de muestreo). Utiliza el *Método Runge-Kutta de orden 4* programado manualmente. Una vez resuelta la integración, devuelve el valor del vector de estados en el tiempo final del intervalo de integración. Nótese que, dado que utilizamos un mantenedor de orden cero para la acción de control, el parámetro de entrada *u* es constante en el intervalo. Necesita como entradas el intervalo de tiempo *intervalo*, el valor del vector de estado *xo* en el extremo inferior del intervalo, la acción de control *u*, y el vector de parámetros a optimizar (I_r, I_{yy}).

3. *function [J]=calculo_funcional_J(parametros,aux):*Su tarea es usar la función 2 para integrar la ecuación diferencial del modelo en tantos tiempos de muestreo Tm como datos haya en nuestro conjunto de prueba obtenido en el punto 3.2.2.1 . Una vez realizada la integración, se aplica la ecuación (3-1) para el cálculo del funcional J, que mide cuán parecidas son las trayectorias reales y simuladas. Recibe como entradas el vector parámetros, con los valores de I_r e I_{yy}, y la variable auxiliar *aux*, que vale 1 para mostrar gráficas y 0 en caso contrario.

4. *function [Jmin]=optimizacion(ParametrosIniciales,modo)*: La tarea de esta función depende de sus argumento *modo.* Si *modo=='optimizar'*, se llama a la función de optimización *fmincon*, que utiliza la función 3 para el cálculo de los valores óptimos de los parámetros I_r y I_{yy}. La función *fmincon* necesita una estimación inicial de los parámetros *ParametrosIniciales* y las restricciones que deben cumplir dichos parámetros. Si *modo=='integracion'*, simplemente se integra el modelo para los parámetros *ParametrosIniciales*, y se muestra la gráfica comparativa.

Como se ha dicho, la explicación de dichas funciones ha sido muy breve. A continuación, y para mayor comprensión, se deja el código con comentarios a los lados.

Nota: la función principal es *optimización.m*, el esto son subfunciones asociadas a la primera.

```
function [Jmin]=optimizacion(ParametrosIniciales,modo)
%------------------------------------------------------------------------%
%------------------------------------------------------------------------%
%Explicación de la función
%
%El funcionamiento de esta función depende de su argumento modo:
%Si modo=='optimizar':
        %1. Se configuran las restricciones en los parámetros( deben ser
        %mayores que 0 ya que son inercias y por definición, positivas.
        %Tambien estan acotadas por arriba, ya que su valor no puede ser
        %muy diferente al que proporcionan las aproximaciones
        %Ir=0.5*mr*R^2 e Iyy=2ML^2).
        %
        %2.Llamada a la funcion fmincon, que nos devuelve el vector de
        %parámetros óptimos que minimizan el valor del funcional J. Se le
        %pasa como argumento ParametrosIniciales
        %
        %3.Muestra las gráficas comparativas real-simulacion tanto para Fi
        %como para dFi/dt.
        %
%Si modo=='integracion':
```

```
        %1. Integra el modelo para los parametros ParametrosIniciales
        %
        %2. Muestra las gráficas comparativas entre las señales reales y
        %simuladas.
%Si existe error en los argumentos, se muestra mensaje de error y se acaba
%con la ejecución del programa.
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
if nargin<2
    error('Faltan argumentos, vease la ayuda help optimizacion.m')
end

aux=0;
if strcmp(modo,'optimizar')==1

LB=[0 0]';
UB=[0.01 1]';

[ParamsOpt,Jmin]=fmincon(@(parametros)calculo_funcional_J(parametros,aux)...
    ,ParametrosIniciales,[],[],[],[],LB,UB);

Iyy=ParamsOpt(1);
Ir=ParamsOpt(2);

disp('La inercia del solido 2 vale:(Kgm^2)')
disp(Iyy)
disp('La inercia de la rueda vale:(Kgm^2)')
disp(Ir)
aux=1;
calculo_funcional_J(ParamsOpt,aux);

else
    if strcmp(modo,'integracion')==1
    aux=1;
    calculo_funcional_J(ParametrosIniciales,aux);
    else
        error('Argumento no valido\n');
    end
end
%-------------------------------------------------------------------------%
%----------------------------FIN FUNCIÓN----------------------------------%
%-------------------------------------------------------------------------%
end

function [J]=calculo_funcional_J(parametros,aux)
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Explicación de la función

%Esta funcion utiliza [y,T] = Runge_Kutta_orden_4(intervalo,u,xo,params)
%para integrar el modelo en tantos tiempo de muestreo dure la simulación
%sobre el sistema real,es decir, según el numero de muestras que posea el
%fichero datos1.m, el cual posee los datos del experimento. Recibe el
%vector de parámetros a optimizar parametros=[Iyy Ir], y una variable
%auxiliar aux. Si aux==1 : integra y muestra las graficas comparativas
%real-simulacion. Si aux==0 : se limita a integrar el modelo y a calcular
%el funcional J, que miden cuán parecida es la salida del modleo matemático
%y la trayectoria real.
%Esta funcion se utilizará como argumento para la funcion fmincon, la cual
```

```
%busca los parámetros óptimos que minimizan J.
%------------------------------------------------------------------------%
%------------------------------------------------------------------------%
%Fichero a modificar ¡¡RETOCAR AQUI!!
%
%Modifique datos1.m con los datos de la simulación
%propia. Dichos datos deben estar en GRADOS º. En caso contrario, modifique
%las lineas 99-100-101. Datos1 es un fichero con 3 columnas y n filas.
%Columna 1:Fi, Columna2:dFi/dt, Columna3:Uk
%Modifique el valor del tiempo de muestreo [s]. Juegue con el
%valor de landa en el cálculo del funcional J. Landa sirve para ponderar
%más los errores iniciales que los errores finales.

load datos1.m;
Tm=0.02;
landa=0.995;
%------------------------------------------------------------------------%
%------------------------------------------------------------------------%
Fi=(2*pi/360)*datos1(:,1);
dFi=(2*pi/360)*datos1(:,2);
Uk=(2*pi/360)*datos1(:,3);
clear datos1
n=length(Fi);
xo=[Fi(1) dFi(1)]';
Tf=n*Tm-Tm;
t=0:Tm:Tf;
Xk=zeros(2,n);
Xk(:,1)=xo;

for k=1:n-1
    [y,~] = Runge_Kutta_orden_4([t(k) t(k+1)],Uk(k),Xk(:,k),parametros);
    Xk(:,k+1)=y(:,2);
end

if aux==1
    subplot(2,1,1)
    plot(t,Xk(1,:),'r',t,Fi,'b'),shg
    subplot(2,1,2)
    plot(t,Xk(2,:),'r',t,dFi,'b'),shg
end

J=0;
for k=1:n
    J=J+(landa^k)*(dFi(k)-Xk(2,k))^2 ;
end

%------------------------------------------------------------------------%
%----------------------------FIN FUNCIÓN---------------------------------%
%------------------------------------------------------------------------%
end

function [y,T] = Runge_Kutta_orden_4(intervalo,u,xo,params)
%------------------------------------------------------------------------%
%------------------------------------------------------------------------%
%Explicación de la función
%
%Integra el modelo dado por
%[dX] = dinamic_two_wheeled_inverted_pendulum(~,x,u,params) durante un
%intervalo de tiempo intervalo=[To Tf] que verifica Tf-To=Tm=0.02s. Por
```

```
%tanto, integra la ecuacion diferencial en UN SOLO tiempo de muestreo.
%Recibe params=[Iyy Ir] con los parámetros a optimizar,el estado inicial en
%T=To, y la acción de control u, constante en el intervalo dado que usamos
%un mantenedor de orden cero para la acción de control
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Código fijo

h=0.0001;
T=intervalo;
time=intervalo(1):h:intervalo(2);
n=length(time);
Y=zeros(2,n);
Y(:,1)=xo;

for k=1:n-1

    k1=dinamic_two_wheeled_inverted_pendulum(0,Y(:,k),u,params);
    k2=dinamic_two_wheeled_inverted_pendulum(0,Y(:,k)+0.5*k1*h,u,params);
    k3=dinamic_two_wheeled_inverted_pendulum(0,Y(:,k)+0.5*k2*h,u,params);
    k4=dinamic_two_wheeled_inverted_pendulum(0,Y(:,k)+0.5*k3*h,u,params);

    Y(:,k+1)=Y(:,k)+(h/6)*(k1 + 2*k2 + 2*k3 + k4);
end
y=[Y(:,1) Y(:,end)];
%-------------------------------------------------------------------------%
%----------------------------FIN FUNCIÓN----------------------------------%
%-------------------------------------------------------------------------%
end
function [dX] = dinamic_two_wheeled_inverted_pendulum(~,x,u,params)
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Explicación de la función
%
%Dado el vector de parametros params=[Iyy Ir] con los parámetros a
%optimizar, y el estado x y la accion de control en un tiempo Tk, devuelve
%la derivada del vector de estados dX/dt en Tk.
%Sirve de base a function [y,T]=Runge_Kutta_orden_4(intervalo,u,xo,params)
%para que esta pueda integrar la ecuacion diferencia en un tiempo
%T=Tm=0.02s=Tiempo de muestreo.
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Parametros del modelo: ¡¡¡¡¡RETOCAR SEGUN NUESTRO PÉNDULO INVERTIDO!!!!!

M=0.94;             %Masa del solido 2 en Kg
mr=0.065;           %Masa de la rueda en Kg
R=0.05;             %Radio de las ruedas en m
L=0.05;             %Distancia desde la base ejes solido 2 hasta el
                    %centro de masas del solido 2 en m
g=9.81;             %Aceleración de la gravedad ms^-2
Iyy=params(1);      %Inercia del solido 2 respecto de los ejes 2 Kgm^2
Ir=params(2);       %Inercia del solido 3 y 4(ruedas) respecto del eje del
                    %motor Kgm^2
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Código fijo
%
%Notesé que según nuestras ecuaciones en el espacio de estados:
% x(1)=Fi
```

```
% x(2)=dFi/dt

dX=zeros(2,1);
dX(1)=x(2);
dX(2)=(M*g*L*x(1)-(2*Ir+2*mr*R^2+M*R^2+M*R*L)*u)/(Iyy + M*R*L);
%-------------------------------------------------------------------------%
%----------------------------FIN FUNCION----------------------------------%
%-------------------------------------------------------------------------%
end
```

Figura 3-5. Código de la función optimización.m.

3.3 Resultados de la estimación

La figura 3-6 nos muestra la comparativa gráfica entre el modelo obtenido mediante la optimización de parámetros y el sistema real. Se puede observar como el módelo predice muy bien a corto plazo el estado del sistema, pero va divergiendo lentamente debido a factores como:

- Errores de modelado: se trata de un modelo linealizado de un modelo no lineal obtenido con numerosas suposiciones.
- Ruidos en los sensores.
- Los modelos de los sistemas inestables son difíciles de identificar.

Aun así, la calidad del modelo es notable, siendo capaz de describir la evolución del sistema sin errores apreciables durante dos segundos.

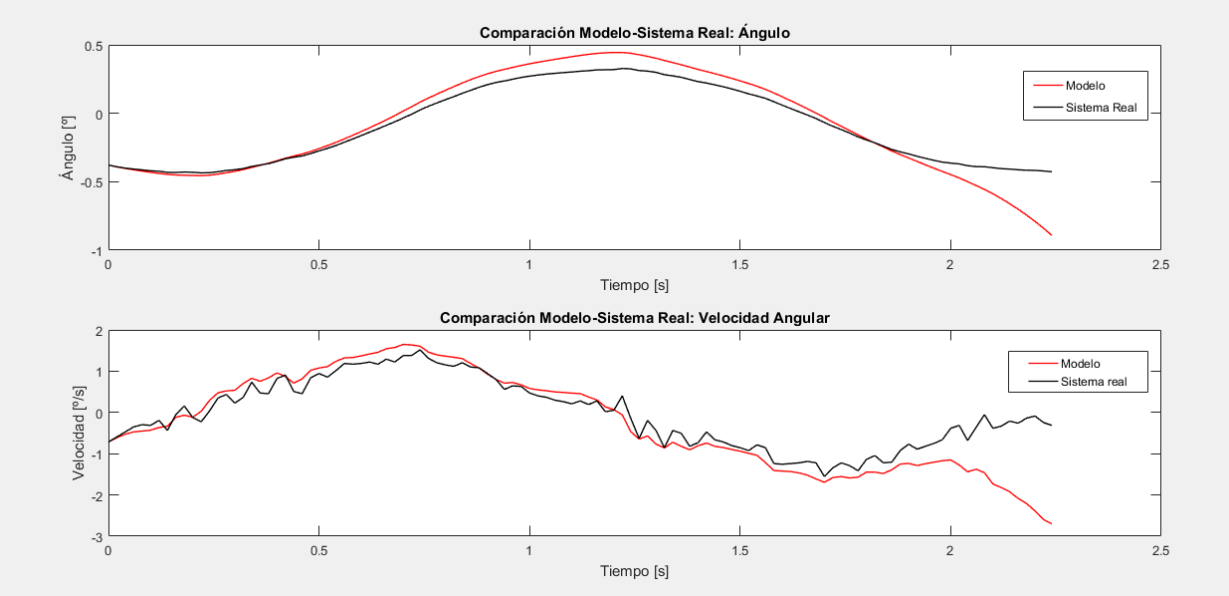


Figura 3-6. Comparativa entre el modelo y el sistema real.

Por tanto, el añadiendo los valores de L, I_r e I_{yy} a la tabla 3-1, obtenemos la tabla resumen con todos los

parámetros del modelo dinámico.

Tabla 3-2. Valores de los parámetros del modelo

Parámetro	Valor
M_2	945g
R	5cm
m_r	65g
g	9.81ms^{-2}
L	5cm
I_r	0.00098421Kgm^2
I_{yy}	0.00994356Kgm^2

Como nota, se deja una comparativa entre los valores que predecían las ecuaciones (2-57) y (2-58) y los valores obtenidos mediante la optimización.

$$I_r \simeq 0.00118125\ Kgm^2 \tag{3-3}$$

$$I_{yy} \simeq 0.004725\ Kgm^2 \tag{3-4}$$

Se puede observar como los ordenes de magnitud se asemejan a los dados por la tabla 3-2.

Si calculamos los errores relativos de las aproximaciones respecto a los estimados (consideramos los valores estimados como los reales):

$$\frac{\left|I_r - I_r^{opt}\right|}{I_r^{opt}} 100 \simeq 20\% \tag{3-5}$$

$$\frac{\left|I_{yy} - I_{yy}^{opt}\right|}{I_{yy}^{opt}} 100 \simeq 50\% \tag{3-6}$$

Dichas diferencias podrían explicarse como:

- Las ruedas poseen 3 agujeros de tamaño no despreciable respecto al área de un circulo.
- El motor no solo mueve la rueda, sino que también mueve el eje metálico. La inercia del eje no ha sido añadida al modelo por la dificultad de su medida.
- La hipótesis de $I_{yy} \simeq 2ML^2$ requiere una distribución de masas uniforme, lo cual no se cumple.

Con esto concluye la *Identificación de Parámetros.*

4 DISEÑO ELECTRÓNICO Y ESTRUCTURAL

En este punto de la memoria mostraremos, en primer lugar, la electrónica que formará el sistema de control. Se mostrarán los diferentes dispositivos que forman el mismo, las características principales y las conexiones eléctricas. El cómo usar los diferentes dispositivos se mostrará en el capítulo 5 del presente documento, donde diseñaremos el programa de control.

En segundo lugar, se mostrará brevemente cómo se ha diseñado la estructura del vehículo haciendo uso del programa *CorelDRAW X8*. Se justificará la elección del programa, se hablará de los requisitos estructurales del chásis del vehículo y se mostrarán diferentes fotografías del mismo desde diferentes puntos de vista.

En tercer y último lugar, se realizará un presupuesto aproximado del coste de fabricación del vehículo, excluyendo los costes de mano de obra.

Se dejará un anexo al final de la presente memoria, que servirá de manual para todo aquel que desee usar la impresora láser del Dpto. de Ingeniería de Sistemas y Automática. Se mostrarán los pasos a seguir para cortar piezas en madera a partir de un archivo *CorelDraw*. También se dejarán en el anexo las acotaciones de las diferentes piezas diseñadas.

4.1 Electrónica del vehículo

En la figura 4-1, que se muestra a continuación, se presenta el esquema general del sistema de control que se montará sobre el vehículo.

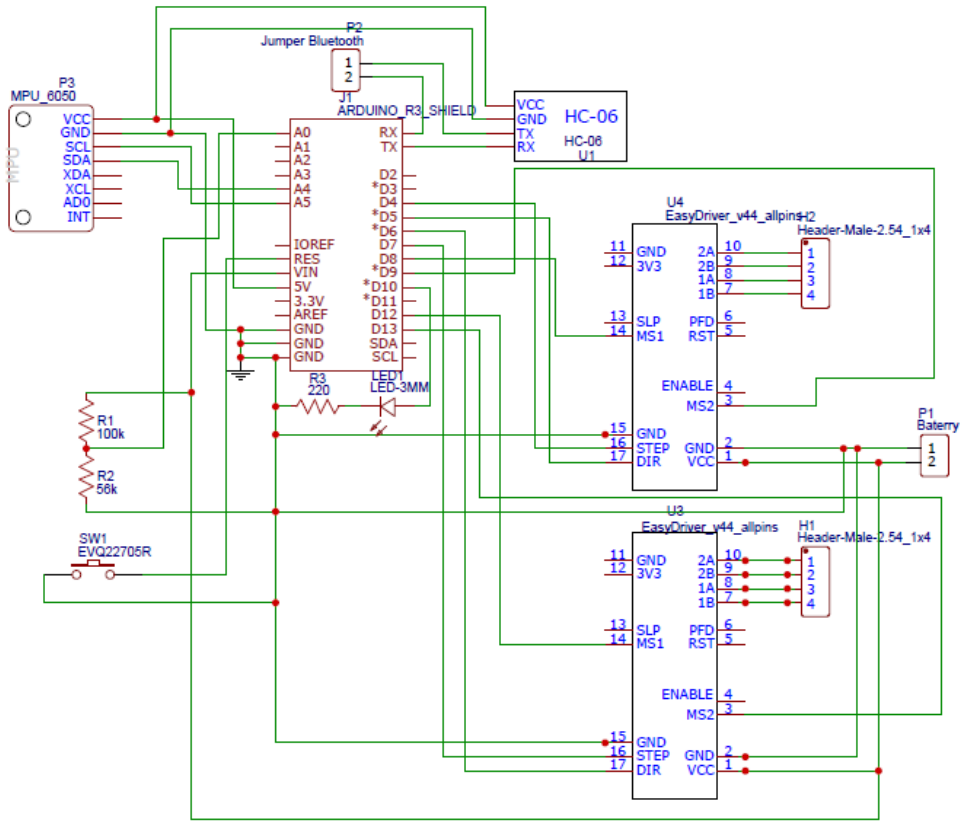


Figura 4-1. Esquema general del sistema de control.

Dentro del esquema se pueden distinguir los diferentes dispositivos:

- Un microcontrolador *Arduino-UNO*.
- Dos drivers de motores paso a paso *EasyDriver_v44 A3967*.
- Una unidad de medición inercial IMU *MPU-6050*.
- Un modulo bluetooch *HC-06*.
- Una batería LIPO de 11.1v (3S).
- Dos motores paso a paso *NEMA-17*, mostrados en la figura con la etiqueta Header-Male.
- Un botón de *Reset* para el microcontrolador.
- Un *divisor de tensión* para medir carga de la batería.
- Un *Jumper* para conexión/ desconexión del modulo bluetooch.

Las conexiones entre los diferentes dispositivos también se muestran en la anterior figura por lo que, cuando procedamos a la programación del microcontrolador en el punto 5 de la memoria, tendremos que tener muy presente a qué pines del *Arduino-Uno* están conectados el resto de los dispositivos.

A continuación, se verán los dispositivos anteriores uno a uno, comentando sus características principales. Nos ayudaremos para ello del trabajo de fin de grado titulado *Desarrollo de un Robot autoequilibrado basado en Arduino con motores paso a paso,* de Dº José Antonio Borja Conde. Para más información, se recomienda al lector la lectura de los *datasheets* del fabricante.

4.1.1 Motores Nema 17

Se trata de un motor pasa a paso de tipo bipolar. Se caracteriza por disponer de dos bobinas que necesitan cambios en el sentido de las intensidades que circulan por ellas para poder generar pasos. Se tienen dos terminales para cada bobina. Energizándolas a través de estos, de forma ordenada, se consigue que el motor genere un paso o micropaso (que será la media, cuarta u octava parte de un paso completo). Por tanto, se necesita producir una secuencia repetitiva de energización de las bobinas. Para generarla será necesario incluir un *microstepping motor driver*, descrito en el siguiente apartado.

Figura 4-2. Motor Nema 17

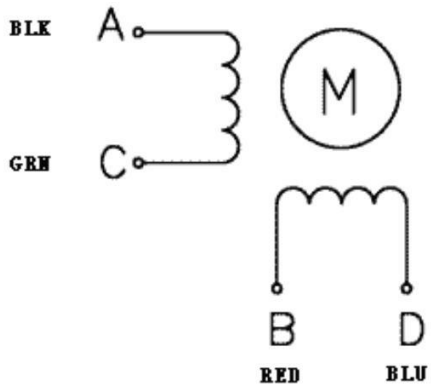


Figura 4-3. Esquema interno de un motor bipolar

Estos motores se caracterizan por tener alta precisión y potencia. Sus especificaciones podemos encontrarlas en la siguiente tabla:

Tabla 4-1. Características principales de los motores

Características	Medidas y unidades
Corriente nominal	1,2 A DC
Tensión de alimentación	12-36 V
Ángulo de avance	1.8º, 0.9º, 0.45º ó 0.225º
Par de fricción	12mN.m
Par de mantenimiento	≥300mN.m (I = 1.5 A)
Máx. Frecuencia de arranque sin carga	≥1500pps
Máx. Frecuencia de ejecución sin carga	≥8000pps
Peso del motor	0.23Kg
Tamaño (L * W * H)	42 x 42 x 34 mm

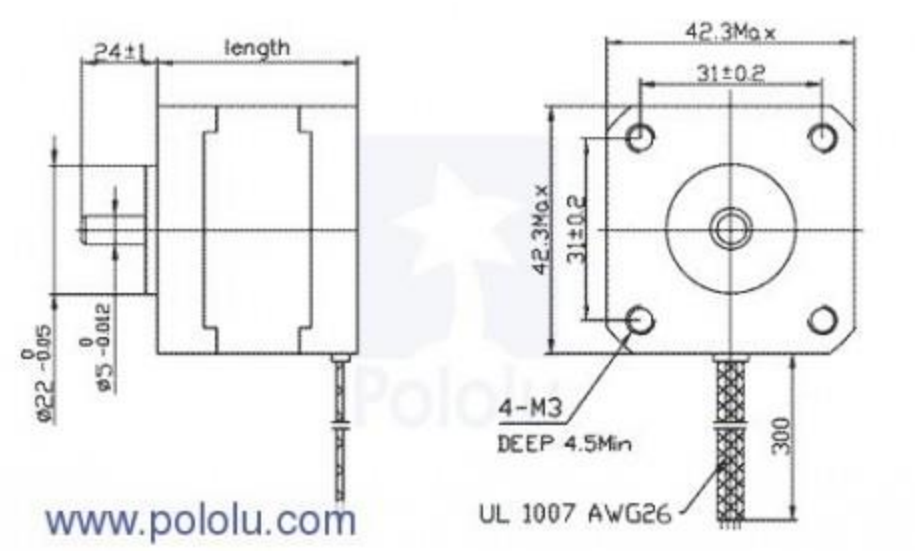


Figura 4-4. Dimensiones del Nema 17.

4.1.2 Driver motor paso a paso A3967

El *A3967* es un *driver* para motores paso a paso bipolares. Se puede usar para modos de medio, cuarto, octavo paso, y para paso completo. Pudiéndose alternar entre estos en tiempo de ejecución. Para ello consta de un repetidor que permite generar las secuencias necesarias en los devanados de las bobinas del motor, y además suministrar voltajes distintos al de sus entradas digitales (siempre y cuando se le esté proporcionando por los pines correspondientes el voltaje deseado).

Gracias al repetidor, el generar un paso/secuencia de pasos (o micropasos), se resume a enviar un pulso/tren de pulsos por el pin *STEP*.

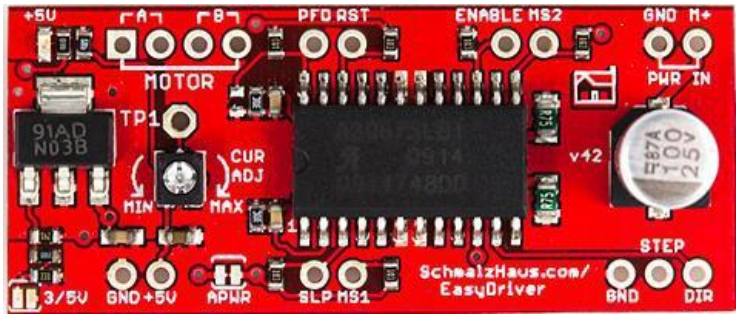


Figura 4-5. Driver V44 A3967.

4.1.2.1 Pines de entrada y salida

En este apartado, por simplicidad, solo se describen los pines que se utilizan en el proyecto. Para más información se puede recurrir al *datasheet*, incluido en la bibliografía.

- Entrada para reinicio (***RESET***): Cuando se active (a nivel bajo) el repetidor toma las condiciones iniciales y apaga todas las salidas. Las señales en el pin *STEP* son ignoradas hasta que se desactive *RESET*.
- Entrada de pasos (***STEP***): Un flanco de subida provoca un incremento unitario de paso (o micropaso). El movimiento se realizará en el sentido indicado por el pin *DIR*.
- Entrada para determinar el sentido de giro (**DIR**): Si se encuentra a nivel alto, el movimiento es un sentido, y si el nivel es bajo, el movimiento tendrá sentido contario.
- Las salidas para los devanados de las bobinas del motor (**A+, A-, B+, B-**): A diferencia de las entradas anteriores, son salidas analógicas. Entregan los voltajes correspondientes a cada bobina para energizarlas según la secuencia. Conocer los valores que toman no es relevante para la realización de este proyecto.
- Pines de alimentación (**GND** y **V+**): Reciben la alimentación tanto para alimentar los motores como para alimentar la propia placa.
- Entradas de selección de micropasos (***MS1*** y ***MS2***): Permite seleccionar modo de medio, cuarto u octavo de paso, o paso completo, según la combinación detallada en la siguiente tabla donde *L* y *H* denotan nivel bajo y alto, respectivamente:

Tabla 4-2. Tabla de verdad de resolución de micropasos

MS1	MS2	Resolución
L	L	Paso completo
H	L	Medio de paso
L	H	Cuarto de paso
H	H	Octavo de paso

4.1.2.2 Especificaciones principales

El rango de voltaje de salida para alimentar a los motores es de 4.75 a 30V, valores entre los que está comprendido el voltaje nominal (12V) de los escogidos.

El rango del voltaje de entradas lógicas para nivel bajo es de 0 a 1.155V, y el de nivel alto 1.98 a 5.5 V, por lo que es adecuado para las salidas del microcontrolador (*Arduino UNO R3*), que se encuentran a 0V a nivel bajo y a 5V a nivel alto, aproximadamente.

4.1.2.3 Restricciones temporales

La detección del cambio de estado en los pines de entrada está condicionada por un mínimo de tiempo. Esto limita la frecuencia de cambio de cada pin. En la siguiente figura, obtenida del *datasheet*, se puede ver un esquema explicativo, el cual se complenta con la siguiente tabla.

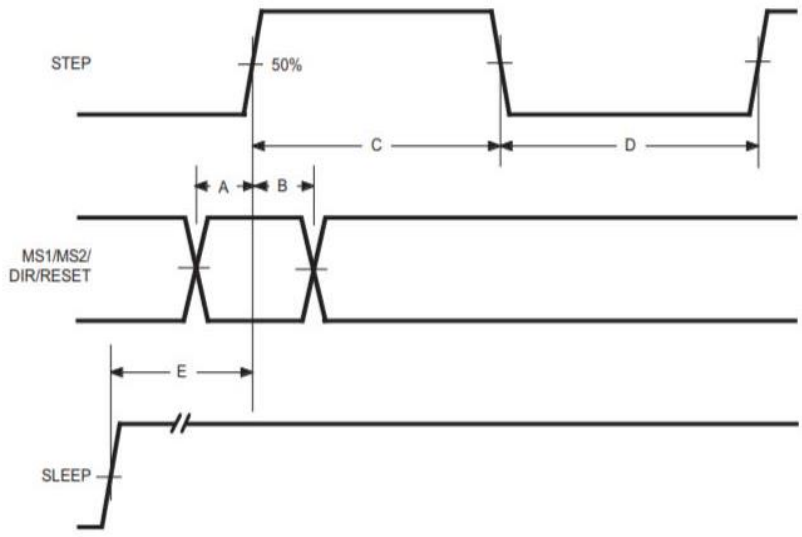


Figura 4-6. Restricciones temporales de V44 A3967.

Tabla 4-3. Restricciones temporales de *A3967*

Etiqueta	Descripción	Valor
A	Tiempo de configuración de datos	200 ns
B	Tiempo de retención de datos	200 ns
C	Mínimo ancho de pulso *STEP* (alto nivel)	1.0 µs
D	Mínimo ancho de pulso *STEP* (bajo nivel)	1.0 µs
E	Tiempo máximo de activación	1.0 ms

Debido a los mínimos ancho de pulso de *STEP*, la frecuencia máxima de pasos queda limitada a 500 kHz.

4.1.3 Batería Lipo

Para la alimentación completa del sistema se ha usado una batería Li-Po 3S (11.1 V) con una capacidad de 1500 mAh.

Figura 4-7. Batería *Li-Po* 3S 1500 mAh

Esta batería consta de 3 celdas. Cada celda tiene un voltaje nominal de 3.7 V, pero cuando están cargadas al máximo proporcionan 4.2 V, por lo que se tiene 12.6 V en total.

Sus dimensiones son 70 * 35 * 25 mm.

4.1.4 Unidad de medición inercial

Como se dijo anteriormente, en este proyecto, para el control del sistema, será necesario conocer el ángulo de inclinación del vehículo con respecto a al horizontal. Para ello se ha decidido usar la unidad de medidas inerciales *MPU6050*, perteneciente a la compañía Invensense.

Para facilitar su conexión con *Arduino*, se ha comprado ya integrada en el módulo GY-521. Este incluye los complementos necesarios para el correcto uso de la *IMU*, y facilita el acceso a sus pines.

La principal característica de la MPU6050 es que tiene seis grados de libertad, ya que tiene un acelerómetro y un giroscopio, ambos de 3 ejes. En este proyecto solo se usarán tres grados de libertad (dos ejes del acelerómetro y uno del giroscopio), lo que podría hacer pensar que podría encontrarse otra unidad de medidas inerciales más económica e igualmente válida. No obstante, el precio de la MPU6050 es muy reducido, ya que su uso está muy generalizado, por lo que no merece la pena buscar otras alternativas que solo proporcionen las prestaciones estrictamente necesarias.

Figura 4-8. IMU MPU6050.

Como se ha indicado en el apartado anterior, se usarán los ejes del acelerómetro y un eje del giroscopio.

Ambos sistemas de medida proporcionan el valor de la misma variable, pero, los datos serán útiles:

- El acelerómetro descompone la aceleracíon de la IMU en sus tres ejes. Suponiendo que las aceleraciones que mide a consecuencia de los cambios de velocidad en el vehículo o las vibraciones de los motores son despreciables frente a la aceleración de la gravedad, descomponiendo el vector g en sus tres componentes y aplicando trigonometría, es posible la obtención del ángulo con la vertical

- El giroscopio proporciona una medida de las componentes del vector velocidad angular. La integración de dicha medida nos da otra información del ángulo.

El giróscopo presenta el problema de que la integración prolongada de la velocidad angular diverge con el tiempo debido a los ruidos en la medida. El problema del acelerómetro es que proporciona una señal muy ruidosa, y además mide otras aceleraciones. Como se verá en el capítulo 5, se diseñará un filtro complementario que combine las 2 medidas para estimar el angulo con la vertical.

Tabla 4-4. Características principales del giroscopio

Característica	Medida
3 convertidores Analógico/Digital	16 bits
Rango de salida programable	±250, ±500, ±1000, y ±2000°/s
Intensidad nominal	3.6 mA

La mayor sensibilidad se obtiene programando el rango a ±250º/s, el cual es suficiente para el sistema estudiado, por lo que se ha decidido usar este.

Tabla 4-5. Características principales del acelerómetro

Característica	Medida
3 convertidores Analógico/Digital	16 bits
Rango de salida programable	±2g, ±4g, ±8g y ±16g
Intensidad nominal	500 μA

La mayor sensibilidad se obetiene programando el rango a ±2gº/s, el cual es suficiente para el sistema estudiado, por lo que se ha decidido usar este.

4.1.5 Arduino UNO

A continuación, se muestra una tabla con las características más importantes del microcontrolador Arduino Uno. Se recuerda que para más información véanse el datasheets del Atmega 328P, cuyo enlace se dejará en la bibliografía de la presente memoria.

Tabla 4-6. Características del Arduino UNO

Microcontrolador	Atmega328P
Voltaje de operación	5v
Tensión de entrada (Recomendada)	7-12v
Tensión de entrada (Límites)	6-20v
Pines digitales I/O	14 (de los cuales 6 configurables como PWM)
Salidas PWM	6
Entradas analógicas	6
Corriente por los pines I/O	20mA
Corriente en pines 3.3v	50mA
Memoria Flash	32KB
SRAM	2KB
EEPROM	1KB
Frecuencia de reloj	16MHz
Length	68.6 mm
Width	53.4 mm

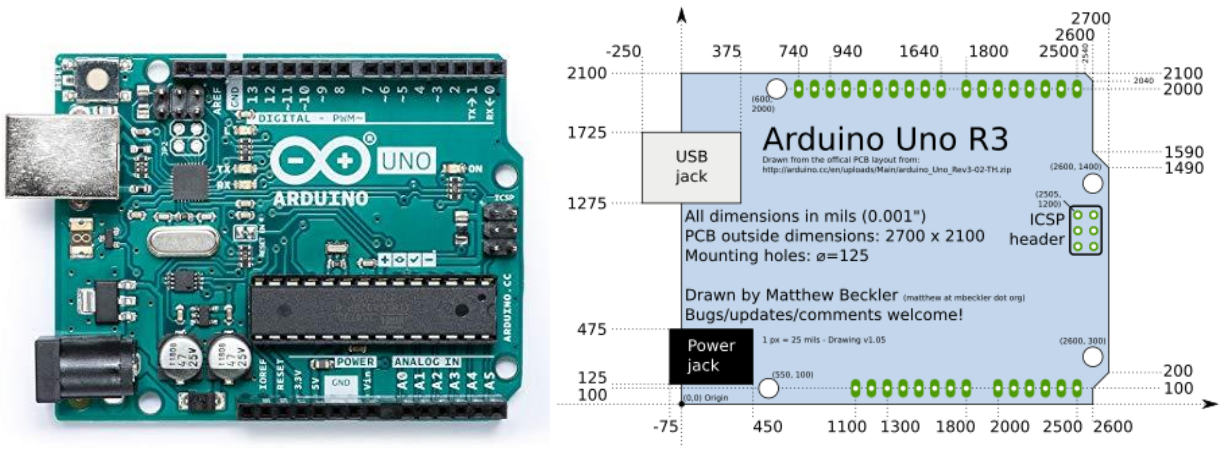


Figura 4-9. Arduino UNO R3 y sus dimensiones.

4.1.6 Modulo HC-06

Se adquiere un módulo Bluetooth HC-06, este dispositivo se comunica con la placa microcontroladora a través de un puerto serie y permite trabajar en un amplio rango de velocidades de transferencia que van desde 1200 baudios hasta 1382400 baudios.

Este módulo será usado para obtener datos del vehículo sin necesidad de conectar con un cable el ordenador y el segway, es decir, proporciona comunicación inalámbrica. Esto proporciona la opción de obtener los datos en tiempo real, siendo mucho más fácil y rápido el estudio del ajuste del funcionamiento del vehículo.

En el punto 5, donde diseñarmos el programa de control en Arduino, mostraremos cómo poder configurarlo mediante comandos AT.

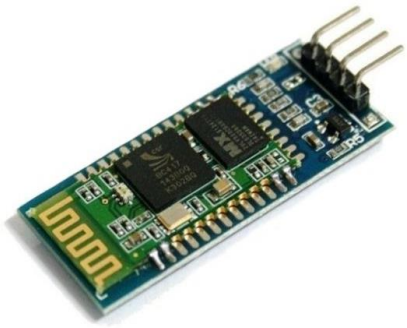


Figura 4-10. Bluetooch HC-06.

4.1.7 PCB

En la siguiente imagen se puede observar el diseño de una placa de circuito impreso PCB (Printed circuit board) realizado a través de la herramienta online *EasyEDA*.

En ella se sueldan los diferentes componentes electrónicos del sistema de control y, una vez montada sobre el Arduino, todos los dispositivos quedan conectados a través de las pistas, por lo que exhime del uso de cables.

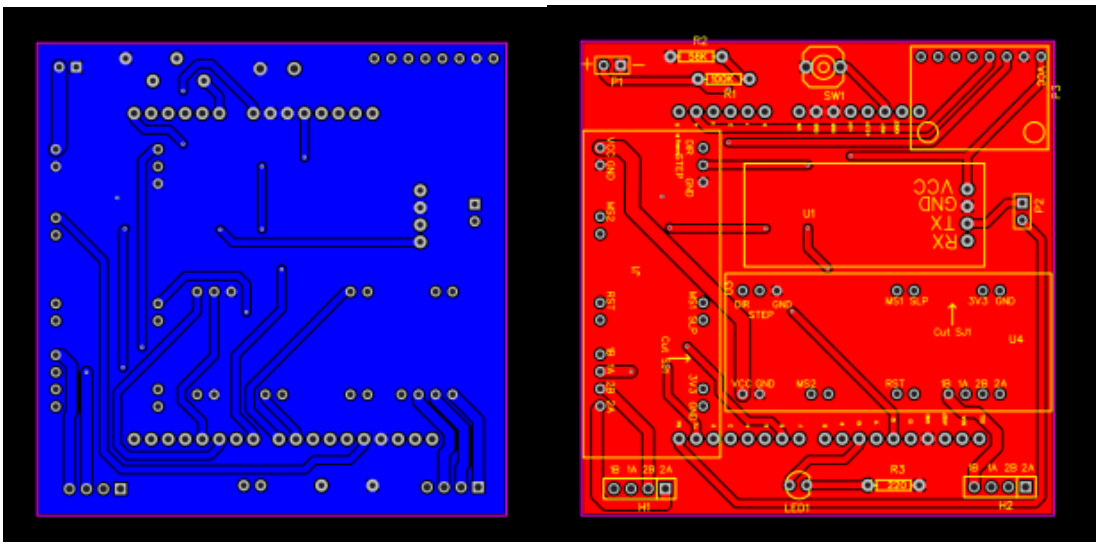


Figura 4-11. PCB del sistema de control.

4.2 Diseño estructural del vehículo

Los requisitos que ha de cumplir el chasis son:

- Una vez que conocemos toda la electrónica que se ha de montar sobre el vehículo, el objetivo es diseñar el chasis de tal manera que permita albergar en él los diferentes dispositivos. Por ello, es necesario ayudarse de las cotas de los motores Nema 17, de la batería Lipo, del Arduino Uno, de la PCB, etc.
- Las diferentes piezas que conforman al vehículo deben quedar unidas físicamente sin el uso de pegamentos, por lo que se utilizarán chavetas, las cuales aportarán rigidez absoluta al conjunto.

Con respecto al programa de diseño en 2D, se ha utilizado *CorelDRAW X8*. Su elección se explica porque la cortadora láser utiliza el programa *Corel Láser*, el cual trabaja con archivos con formato *.cdr*. Por tanto, se ha trabajado directamente con el software del fabricante, ya que trabaja directamente con ese formato. Su uso es bastante sencillo y fácil de aprender. Se dejará en la bibliografía el enlace a un curso en *Youtube*, donde se explica bastante bien cómo empezar a usar el programa.

Poco más se puede decir del diseño ya que, una vez cumplidos los requisitos, el resto es subjetivo. El archivo *two_wheeled_inverted_pendulum_v12.cdr* contiene el diseño del chasis, listo para la impresión láser. Se recuerda que al final de la memoria, en el punto *Anexo*, se dejarán las piezas del chasis con las cotas más representativas. Para más información acerca de las medidas, se recomienda abrir el fichero anterior, y mirar las cotas directamente allí, ya que el software es un poco limitado para realizar acotaciones.

A continuación, se muestran imágenes del prototipo final del vehículo.

Figura 4-12. Vista frontal del vehículo.

Figura 4-13. Vista trasera lateral del vehículo.

Figura 4-14. Vista superior del vehículo.

Figura 4-15. Vista frontal lateral del vehículo.

Figura 4-16. Vista lateral del vehículo.

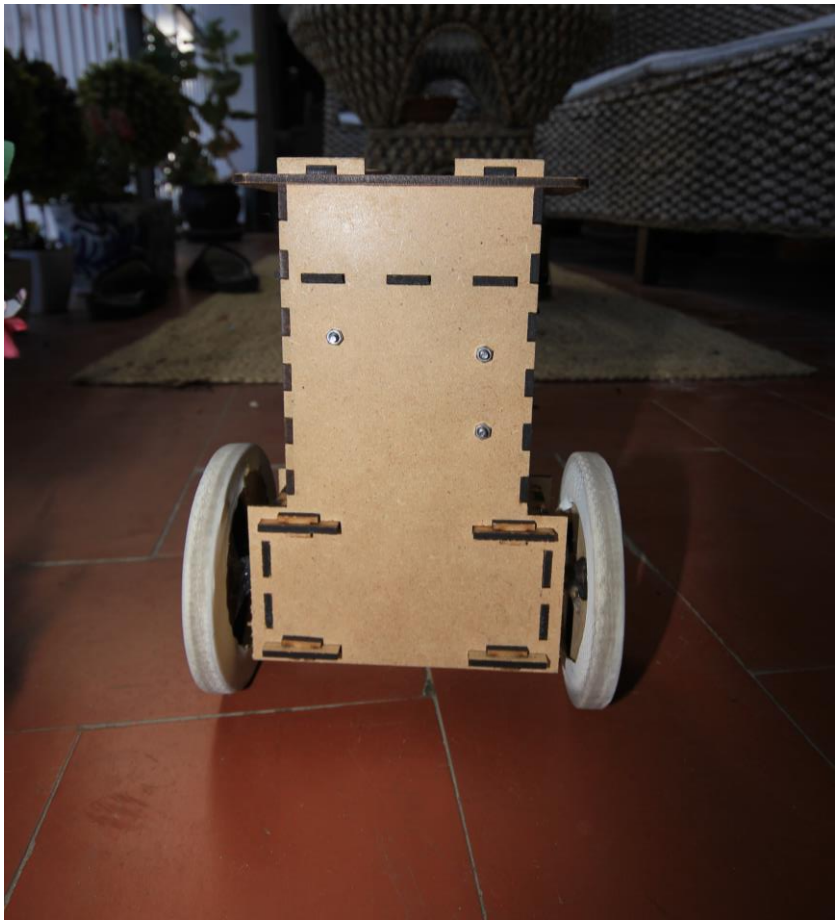


Figura 4-17. Vista trasera del vehículo.

4.3 Coste de fabricación

A continuación, se muestran los costes de los principales componentes. Se excluyen, por tanto, componentes como cables, interruptores, leds, ya que su coste es despreciable frente al de los principales. El objetivo, por tanto, no es establecer el coste exacto del vehículo, sino una aproximación de este.

Nota: La mayoría de los precios han sido obtenidos a partir de la mejor oferta de *Amazon* y *Aliexpress*.

Tabla 4-7. Coste aproximado del vehículo.

Nº Partida	Partida	Cantidad	Precio unitario (€/und)	Precio total (€)
1	Motores Nema 17	2	8 (Amazon) 6 (Aliexpress)	16(Amazon) 12(Aliexpress)
2	Arduino UNO R3	1	20	20
3	Bluetooch HC-06	1	8.5(Amazon) 2.5(Aliexpress)	8.5(Amazon) 2.5(Aliexpress)
4	MPU 6050	1	2(Amazon)	2(Amazon)
5	Drivers A3967	2	3.5(Amazon) 1(Aliexpress)	7(Amazon) 2(Aliexpress)
6	PCB	1	1.5	1.5
7	Batería LIPO 3s 1500mAh	1	19(Amazon) 10(Aliexpress)	19(Amazon) 10(Aliexpress)
7	Lámina madera	2	1	2
			Total min(Amazon, Aliexpress)	**52 €**

Como se puede ver en la tabla 4-7, el coste aproximado es de unos 52 €. Un par de ideas para aminorar el coste serían:

- Utilizar una placa Arduino Uno no original, de coste 3€ en Aliexpress (6€ en Amazon). frente a los 20€. de la original. No es una falsificación en realidad, ya que las placa Arduino son *Open-Source Hardware*, por lo que existen fabricantes que las fabrican y venden a precios más bajos.
- Hablar con los distintos proveedores y ver si a partir de una cierta cantidad de unidades el precio por unidad disminuye.

Nota: los gastos de envio no han sido incluidos. Aunque en Amazon con la suscripción prime son gratis.

Con esto se acaba el punto *Diseño electrónico y estructural del vehículo*.

5 PROGRAMACIÓN DEL MICROCONTROLADOR

Tanto si piensas que puedes, como si piensas que no, estás en lo cierto

Henry Ford

Llegados a este punto, ya tenemos nuestro *Two-Wheeled Inverted Pendulum* construido y, además, poseemos un modelo lineal en el espacio de estados a partir del cual diseñaremos estrategias de control. El objetivo de este punto es la programación del microcontrolador *Arduino UNO*, de tal manera que sea capaz de implementar cualquier ley de control diseñada.

5.1 Introducción

Como cualquier sistema de tiempo real, nuestro vehículo debe realizar numerosas tareas pseudo-concurrentes:

- Recibir comandos del teleoperador.
- Controlar al sistema:
    - Leer sensores
    - Cálcular acción de control.
    - Aplicar acción de control.
- Medir carga de la batería.
- Enviar información del estado a la estación remota.

Dado que el Arduino UNO con Atmega328p solo es capaz de ejecutar una instrucción en cada instante, la concurrencia temporal de las diferentes tareas es imposible. En su lugar, se repartirá el tiempo de CPU entre las distintas tareas, es decir, la CPU irá alternando entre las diferentes tareas de tal manera que se cumplan los tiempos de ejecución de cada una de ellas. De esta manera, parecerá que las diferentes tareas se están ejecutando concurrentemente, cosa que solo sería posible en un sistema con multiprocesador.

La solución adoptada ha sido la de un sistema ‘’monotarea’’ con dos actividades concurrentes. Una activada por interrupciones (‘’Foreground’’), y otra constituida por una tarea normal (‘’Background’’). La primera daría cuenta de las actividades críticas, y la segunda de las no críticas. En cada nivel, el programador tendría que distribuir cuidadosamente el tiempo entre las diferentes tareas.

Particularizando para nuestro caso, la tarea más crítica es la aplicación de la acción de control (Aplicación de los pasos al motor). La justificación es sumamente sencilla: para aplicar una velocidad determinada al motor es necesaria la aplicación de una señal PWM sobre el *Driver V44 A3967* con una frecuencia determinada por lo que, gracias a la prioridad de las interrupciones temporales, la CPU expulsa la tarea que estuviera en ejecución para pasar de HIGH a LOW, o de LOW a HIGH el pin del Arduino conectado al pin **STEP** del driver. Si no existiera la capacidad de expulsar a otra tarea de la CPU, sería imposible conseguir la frecuencia deseada de la

PWM, ya que habría que esperar a que la tarea en ejecución terminase para poder aplicar el paso.

Por tanto, el programa tendrá la estructura que se muestra en la figura que aparece a continuación. Se trata de un programa principal cuyo hilo de ejecución se va interrumpiendo para aplicar los correspondientes pasos a las ruedas. El programa principal se conoce comúnmente como ‘’Ejecutivo Cíclico’’ o “Gran Bucle While”. Se debe diseñar atendiendo a los periodos de repetición de las diferentes tareas que en él aparecen. La difultad de su diseño proviene del hecho de que el tiempo de coste de una tarea (Tiempo que tarda la CPU en ejecutar sus instrucciones) no puede hacer que una tarea siga ejecutándose cuando deberiá de empezar a ejecutarse la siguiente.

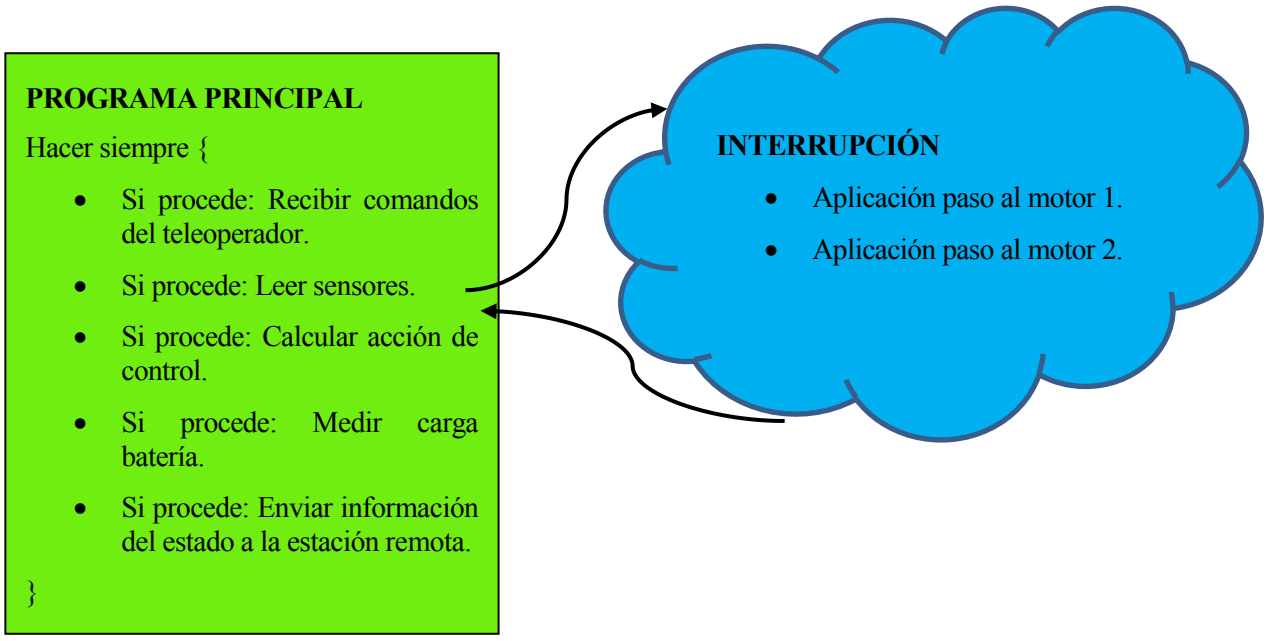


Figura 5-1. Esquema general del programa.

Lógicamente, el ejecutivo cíclico del programa principal debe diseñarse en las peores condiciones. Estas coinciden cuando la velocidad de las dos ruedas toma valores máximos, ya que esta situación coincide con la frecuencia máxima de la señal PWM, lo que significa que las interrupciones de la tarea ‘’Foreground’’ se producen a la máxima frecuencia y, por tanto, es cuando más interrumpen a las tareas del ejecutivo cíclico.

En los siguientes puntos veremos cómo, a partir del esquema de la figura anterior, se va programando el microcontrolador.

5.2 Programación del ejecutivo cíclico

El objetivo de este punto será del de mostrar el diseño del ejecutivo cíclico con tres tareas:

- Leer sensores.
- Cálcular acción de control.
- Medir nivel de la batería.

En el punto 7 titulado *Teleoperación* mostrarán los cambios que se deben hacer sobre este código para incorporar el resto de las tareas que se nombraron en el punto 5.1 *Introducción.*

En primer lugar, del punto *Estudio dinámico* sabemos que la acción de control es la aceleración angular de las ruedas. Dado que utilizamos un mantenedor de orden cero de la acción de control durante el tiempo de muestreo Tm, debemos de ser capaces de aplicar a los motores una aceleración constante en dicho intervalo.

El periodo de muestreo Tm se ha tomado como 20ms. Como se demostrará posteriormente en el capítulo 6, dicho valor es despreciable frente a la constante de tiempo del sistema, por lo que es un valor aceptable para el tiempo de muestreo. Sabiendo esto, la idea para calcular la acción de control es la siguiente: dado que la aceleración de las ruedas es constante durante el tiempo de muestreo, la velocidad angular de las ruedas durante el intervalo es una recta con condición inicial igual a la velocidad en el instante inicial, y con pendiente igual a la aceleración durante el intervalo.

Dado que solo podemos configurar la velocidad de los motores a través de la frecuencia de la señal PWM, la solución radica en aproximar el perfil de velocidad mediante una señal escalonada, tal y como se muestra en la siguiente imagen. Como se puede ver, el periodo de muestreo Tm se ha dividido uniformemente en 5 tramos de duración 4ms.

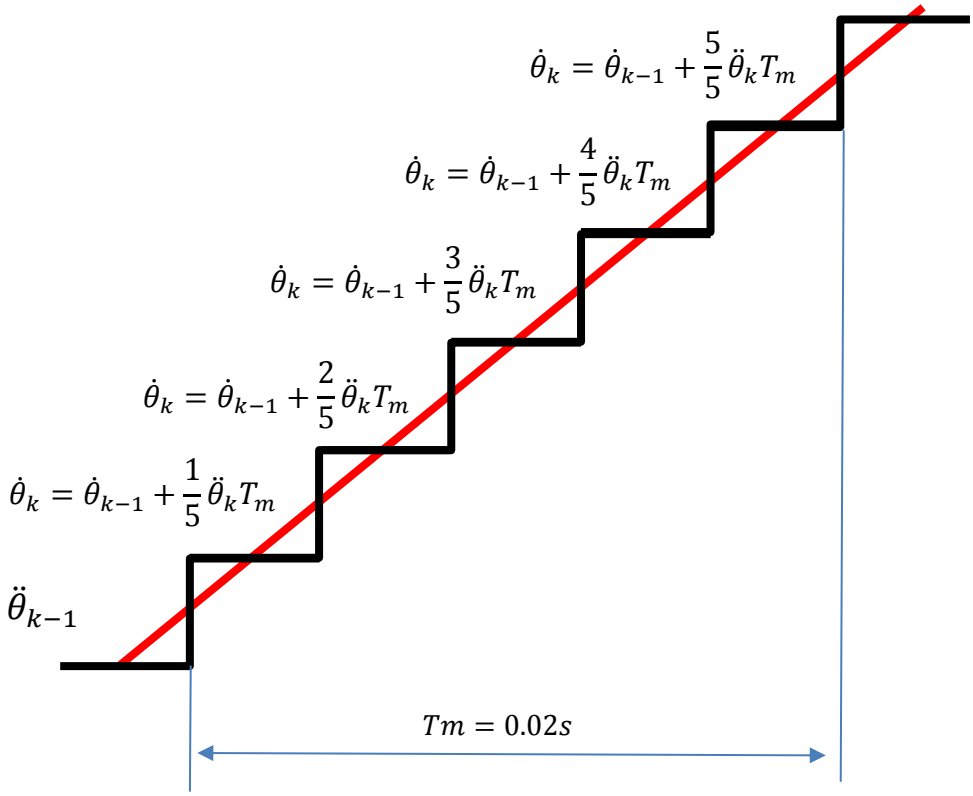


Figura 5-2. Aproximación velocidad mediante señal escalonada.

A continuación, se muestra el código de la función principal donde se programa el ejecutivo cíclico. Como se verá posteriormente, se ha programado el Timer 2 del Arduino de tal manera que provoque interrupciones temporales cada 4ms, de tal manera que cada 5 interrupciones es un nuevo periodo de muestreo. En cada interrupción se pone la variable *flag* a valor 1, indicando esto un nuevo tramo dentro de Tm. Y, dependiendo del número del tramo *ciclo*, se realizan las acciones que se pueden apreciar en el código.

Se puede demostrar que los tiempos de coste de la ejecución de las diversas funciones de cada tramo son inferiores a 4ms, por lo que cumplimos tiempos, lo cual es clave en un sistema de tiempo real. Una manera de comprobarlo es mediante el uso de la función *micros()*, la cual usa el *Timer 0*, el cual no se utiliza para programar interrupciones.

```
void loop() {

  while(flag==0);
  flag=0;
  switch(ciclo){
    case 1:
      leer_sensores();
      controlador();
      Wder_k=Wder_k1+0.2*Alfader*Tm;
      Wizq_k=Wizq_k1+0.2*Alfaizq*Tm;
      aplica_Uk();
    break;

    case 2:
      Wder_k=Wder_k1+0.4*Alfader*Tm;
      Wizq_k=Wizq_k1+0.4*Alfaizq*Tm;
      aplica_Uk();
    break;

    case 3:
      Wder_k=Wder_k1+0.6*Alfader*Tm;
      Wizq_k=Wizq_k1+0.6*Alfaizq*Tm;
      aplica_Uk();
    break;

    case 4:
      Wder_k=Wder_k1+0.8*Alfader*Tm;
      Wizq_k=Wizq_k1+0.8*Alfaizq*Tm;
      aplica_Uk();
      nivel_bateria();
    break;

    case 5:
      Wder_k=Wder_k1+1*Alfader*Tm;
      Wizq_k=Wizq_k1+1*Alfaizq*Tm;
      aplica_Uk();
    break;
  }
}
```

Figura 5-3. Código de la función principal *void loop()*.

En la figura 5-3 se puede observar la programación del ejecutivo cíclico dentro de la función *void loop()* de Arduino. En el tramo 1 se ejecutan las funciones *leer_sensores()* y *controlador(),* donde la primera recibe datos de la imu y estima la orientación mediante un filtro complementario, y la segunda cálcula la acción de control (Aceleración de las ruedas) a aplicar en el intervalo. Posteriormente, tal como se vio en la figura 5-2, se calcula la velocidad a aplicar en cada tramo, y mediante la función *aplica_Uk()* se genera la señal PWM sobre los *Drivers* de los motores paso a paso.

En el tramo 4 se ejecuta, además, la función *nivel_bateria()*, que enciende el led rojo de la placa si la carga es inferior al 20%.

En la siguiente figura, número 5-4, se muestra la configuración de Timer 2 para que provoque interrupciones temporales cada 4ms, como se dijo anteriormente. Notesé que se muestra solo la parte de la función *void setup()* donde se configura el Timer. En el punto 5.2.4 *Explicación de la función aplica_Uk()* se explicará más detenidamente cómo configurar un Timer.

```
void setup() {
//--------------------------------//
//Por aquí sigue la función//
//--------------------------------//

/*interrupciones del Timer 2: llevar cuenta de ciclos secundarios*/
  TCCR2A=0;
  TCCR2B=0;
  TCNT2=0;
  OCR2A=250;//4ms
  TCCR2A |= (1 << WGM21); //CTC
  TCCR2B |= (1 << CS22)|(1 << CS21);//256
  TIMSK2|=(1 << OCIE2A);

//--------------------------------//
//Por aquí sigue la función//
//--------------------------------//

}


ISR(TIMER2_COMPA_vect){
   ciclo++;
   flag=1;

   if(ciclo==6){
      ciclo=1;
      Wder_k1=Wder_k;
      Wizq_k1=Wizq_k;
   }

}
```

Figura 5-4. Configuración Timer 2 para interrupciones cada 4 ms dentro de *void setup()*.

Véase como, en la función interrupción del registro A del timer 2, se lleva la cuenta del tramo en el que nos encontramos, como avisa poniendo *flag* a 1 para indicar nuevo tramo, y como se actualizan las velocidades cuando se pasa de un periodo de control al siguiente.

En lo que sigue, se explicarán los códigos de las diferentes funciones que aparecen dentro del ejecutivo cíclico de la figura 5-3. Nota: se mostrará lo más importante de cada función por lo que, temas, como la inclusión de cabeceras, declaraciones de variables, etc, se excluyen de la explicación. Se procede de esta manera para no extendernos demasiado en la explicación. Si el lector desea saber más detalles acerca de la programación, véanse los archivos .ino.

5.2.1 Función leer_sensores()

La estimación del vector de estados es el primer problema a resolver en la implementación de un controlador. Si se diseñan, como se verá en el capítulo 6 de la memoria, controladores mediante la realimentación lineal del vector de estados, es condición indispensable el conocimiento del valor del vector de estados para el cálculo de la acción de control.

Para ello, haremos uso de las medidas obtenidas de la imu MPU6050 para diseñar un estimador del estado. Como se vió en el capítulo 4 *Electrónica del sistema y sus conexiones*, este dispositivo esta integrado por:

- Giróscopo: mide la velocidad angular. La integración de su medida proporciona la inclinación de nuestro vehículo. Presenta el problema de que la integración prolongada genera una deriva del ángulo que crece con el tiempo.
- Acelerómetro: proporciona una medida del ángulo de inclinación a partir de la descomposición del vector aceleración (idealmente el vector graveda g) en sus 3 componentes y de la aplicación de relaciones trigonométricas. Presenta el problema de que proporciona una señal muy ruidosa, a la vez que mide también otras aceleraciones.

Aunque existiría la posibilidad de utilizar un *Filtro de Kalman*, el cual nos proporcionaría la ‘’mejor’’ (más probable) estimación del estado, nos hemos decantado por la implementación de filtro complementario ya que el coste computacional es muchísimo menor que en el estimador de Kalman.

La idea del filtro complementario consiste en combinar las medidas de ambos sensores, de tal manera que a corto plazo nos guiamos de la medida del giróscopo, utilizando la medida del acelerómetro para corregir la estimación del ángulo a largo plazo. En la siguiente imagen se puede ver gráficamente la idea del filtro.

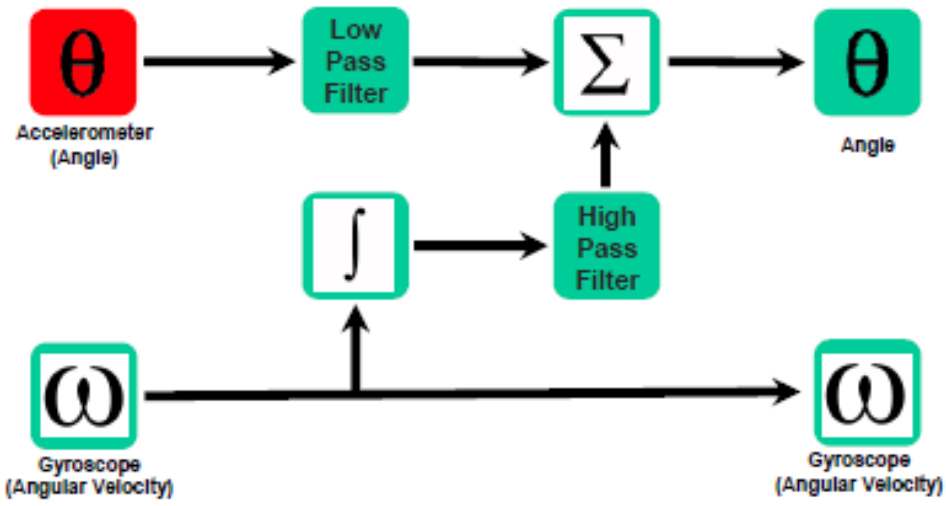


Figura 5-5. Esquema del funcionamiento del filtro complementario.

Matemáticamente, la expresión del filtro complementario se muestra a continuación

$$\phi_k = \alpha\left(\phi_{k-1} + \dot{\phi}_k \Delta t\right) + (1-\alpha)\phi_k^{acel} \tag{5–1}$$

Donde el parámetro mide el peso que tiene cada sensor sobre el ángulo inicial. Un valor demasiado bajo de α provoca mucho ruido en la estimación del ángulo, ya que se guiaría más de la medida del acelerómetro. Un valor demasiado alto de α haría que el estimador fuera muy lento y tardara mucho en estabilizarse en torno a su valor en régimen permanente.

En que caso que nos ocupa, se recomienda tomar $\alpha = 0.99$ y, una vez que estemos controlando al sistema en torno a la vertical, variar suavemente este valor y escoger el valor del mismo que nos diera mejores resultados. En concreto, en este proyecto se ha tomado $\alpha = 0.9975$.

En cuanto a la programación, la cual se explica a continuación, solo decir que se han hecho uso de las librerías *I2Cdev.h* y *MPU6050.h*.

```
void leer_sensores(void){

   mpu.getAcceleration(&ax, &ay, &az);
   mpu.getRotation(&gx, &gy, &gz);
   filtro_complementario();
}

void filtro_complementario()
{
   static float alfa=0.9975;

   tiempo=micros();
   dt=(tiempo-Tiempo_prev)*0.000001;
   Tiempo_prev=tiempo;
   accel_ang_y = atan(-az / sqrt(pow(ay,2)+ pow(ax,2)))*(180.0 / 3.14);
   ang_y = alfa*(ang_y_prev + (gy / 131)*dt) + (1-alfa)*accel_ang_y;
   ang_y_prev = ang_y;
   Fi=ang_y+offset;
}
```

Figura 5-6. Programación de la función *leer_sensores()*.

Su funcionamiento es bastante simple, se obtienen los datos de la imu mediante las funciones *mpu.getAcceleration()* y *mpu.getRotation()*. Posteriormente, se ejecuta la expresión (5-1) dentro de *filtro_complementario*.

5.2.2 Función nivel_bateria()

La idea para estimar el porcentaje de carga es establecer una relación entre el propio porcentaje y la tensión media por celda. Más concretamente, el objetivo es obtener una medida de la tensión media por celda y, a partir de la relación porcentaje-tensíon, calcular el porcentaje de carga.

Para el calcular el voltaje medio por celda nos ayudamos de divisor resistvo incorporado a la PCB. El punto medio del divisor resistivo se encuentra conectado a pin analógico A0 del Arduino por lo que, gracias al conversor analógico-digital asociado a ese pin, se convierte una tensión en el rango 0-5v a un valor entre 0-1023. Como dato, se tiene que las resistencias toman los valores $R_1 = 100000\Omega$ y $R_2 = 56000\Omega$.

La relación Porcentaje de carga-Tensión ha sido tomada del proyecto *Espada Láser*, del profesor Ignacio Alvarado Aldea.

A continuación, se muestra el código de la función y, posteriormente, se explicará su funcionamiento.

```
void nivel_bateria(void){
  if(voltage_measure()<20){
    *out2=(*out2)|bitled; //Encendemos Led rojo HIGH
  }
  else{
    *out2=(*out2)&(~bitled); //Apagamos Led rojo LOW
  }


byte voltage_measure() {
  voltage = 0;
  for (int i = 0; i < 10; i++) {
    voltage += (float)analogRead(VOLT_PIN) * 5 / 1023 * (R1 + R2) / R2;
  }
  voltage = voltage / 10;

  int volts = voltage / 3 * 100;    // 3 cells!!!
  if (volts > 387) // completamente cargada
    return map(volts, 420, 387, 100, 77);
  else if ((volts <= 387) && (volts > 375) )
    return map(volts, 387, 375, 77, 54);
  else if ((volts <= 375) && (volts > 368) )
    return map(volts, 375, 368, 54, 31);
  else if ((volts <= 368) && (volts > 340) )
    return map(volts, 368, 340, 31, 8);
  else if (volts <= 340)
    return map(volts, 340, 240, 8, 0);
}
```

Figura 5-7. Programación de la función *nivel_bateria()*.

La función *nivel_bateria()* calcula el porcentaje de carga haciendo uso de la función *voltage_measure()* y, si este es inferior al 20%, se enciende el led rojo para indicar batería baja. La función *voltage_measure()* mide la tensión que hay en el divisor resistivo conectado al pin A0 y aplica la relación porcentaje-tensión para obtener el porcentaje de carga.

5.2.3 Función controlador()

Poco que decir acerca de la programación de esta función, ya que depende de la estrategia de control utilizada. Solamente decir que, en el código, la acción de control (Aceleración de las ruedas ruedas) está definida por las variables *Alfader* y *Alfaizq*, por lo que cualquier ley de control debe de actualizar los valores de dichas variables.

5.2.4 Función aplica_Uk()

En el código de la figura 5-3 se puede observar cómo, tras el cálculo de los valores de la velocidad de cada rueda en cada subciclo-tramo, tiene lugar la ejecución de la función *aplica_Uk()*. Esta función realiza básicamente dos cosas:

1. Según el signo de la velocidad, el cual determina el sentido de giro de cada rueda, se actualiza el estado del pin **DIR** del driver correspondiente. En el punto 5.2.4.1 se profundizará sobre este tema.
2. Según el módulo de la velocidad angular, se genera una señal PWM sobre el pin **STEP** del driver correspondiente con una frecuencia dependiente de dicho módulo. En el punto 5.2.4.2 se profundizará más sobre este tema.

Antes de comenzar con la explicación, conviene comentar algo acerca de la función *digitalWrite()* de Arduino. El coste computacional de esta función es del orden de magnitud del periodo de la señal PWM cuando la frecuencia de esta es muy grande, lo cual coincide para velocidades altas. Esto es debido a que, cuando queremos cambiar el valor de un pin, al ejecutar esta función (Atendiendo a la estructura interna de *digitalWrite()*):

1. Se calcula el número de puerto al que pertenece el pin con la función *digitalPinToPort()*.
2. Se calcula el registro de salida asociado a ese puerto con *portOutputRegister()*.
3. Se cambia el bit correspondiente dentro del registro de salida.

Como se puede intuir, resulta absurdo repetir infinitamente los puntos 1 y 2 por lo que, dentro de *void setup(),* ejecutaremos esas funciones una única vez para cada pin y guardaremos dichos valores de ta manera que, para cambiar el valor de un pin, solo tendremos que realizar la acción numero 3.

En la siguiente figura se muestra la parte de la función *void setup()* donde se calcularán los registros de salida para los diferentes puertos. Con esta forma de proceder se consigue que el coste computacional de cambiar el valor de un pin sea despreciable para las velocidades angulares con las que vamos a trabajar.

```
void setup() {
//-------------------------------//
//Por aquí sigue la función//
//-------------------------------//
 /*Direcciones de los diferentes pines anteriores para evitar uso
digitalWrite(), por su gran coste*/

  bitMderStep = digitalPinToBitMask(StepMder);
  bitMizqStep = digitalPinToBitMask(StepMizq);
  bitMderdir = digitalPinToBitMask(dirMder);
  bitMizqdir = digitalPinToBitMask(dirMizq);
  bitMderMS1 = digitalPinToBitMask(MS1der);
  bitMizqMS1 = digitalPinToBitMask(MS1izq);
  bitled= digitalPinToBitMask(led);

  port = digitalPinToPort(StepMder);//Los pines estan en el midmo
  port2 = digitalPinToPort(led);    // puerto menos el del led
  out = portOutputRegister(port);
  out2 = portOutputRegister(port2);

//-------------------------------//
//Por aquí sigue la función//
//-------------------------------//
}
```

Figura 5-8. Obtención de los registros de salida en la función *void setup()*.

5.2.4.1 Cómo establecer el sentido de giro del motor

Nuestro vehículo posee dos ruedas independientes, cada una con su *Driver V44 A3967*, tal como se vió en el capítulo 4. En la hoja de características del *driver* se nos dice que el sentido de giro del motor conectado a dicho driver es dependiente del valor lógico aplicado sobre el pin **DIR** de tal manera que, con un valor '1' el motor girará en una dirección y con un valor '0' girará en el contrario.

Para ver el sentido de giro para los dos valores lógicos se recomienda experimentar con el sistema ya que depende de si el motor hace girar a la rueda derecha o a la izquierda (Si un motor colocado en la rueda derecha hace mover el vehículo hacia delante, ese mismo motor colocado en la rueda izquierda movería al vehículo hacia atrás, para un mismo valor lógico en el pin **DIR**).

Particularizando para el vehículo se tiene:

Tabla 5-1. Sentido de giro motor derecho.

Rueda derecha	**Signo de DIR**
$\boldsymbol{\omega > 0}$	'1'
$\boldsymbol{\omega < 0}$	'0'

Tabla 5-2. Sentido de giro motor derecho.

Rueda izquierda	Signo de DIR
$\omega > 0$	'0'
$\omega < 0$	'1'

Más adelante se mostrará dentro del código de la función *aplica_Uk()* dónde aparecen estos cambios de los valores lógicos de DIR para cada rueda.

5.2.4.2 Cómo generar la señal PWM para aplicar los pasos

En este punto se mostrará cómo, a partir de interrupciones temporales del Timer 1, se generarán las correspondientes señales PWM que producirán los pasos en los dos motores.

Antes de nada, se hace necesario configurar el tamaño de paso en ambos motores. El tamaño de paso es el ángulo que gira cuando se aplica un flanco de subida en el pin **STEP** del *driver* correspondiente a dicho motor. Como se comento en el capítulo 4, el tamaño del paso es configurable entre los valores paso, medio paso, cuarto de paso y octavo de paso, siendo un paso 1.8º según el fabricante. Por tanto, para mayor precisión, se configurará el *driver* en modo octavo de paso, lo que significa que, con cada flanco de subida en el pin **STEP**, el motor avanzará 1.8º/8=0.225º.

En la figura 5-7 que se muestra a continuación, se configuran los pines **MS1** y **MS2** de cada Driver, con los valores dados por la tabla 4-2.

```
void setup() {
 /*Configuracion pines del motor derecho*/

  pinMode(dirMder,OUTPUT);
  pinMode(StepMder,OUTPUT);
  pinMode(MS1der,OUTPUT);
  pinMode(MS2der,OUTPUT);
  digitalWrite(MS1der,HIGH);
  digitalWrite(MS2der,HIGH);
  digitalWrite(dirMder,HIGH);     //HIGH es hacia delante
  digitalWrite(StepMder,LOW);

  /*Configuracion pines del motor izquierdo*/
  pinMode(dirMizq,OUTPUT);
  pinMode(StepMizq,OUTPUT);
  pinMode(MS1izq,OUTPUT);
  pinMode(MS2izq,OUTPUT);
  digitalWrite(MS1izq,HIGH);
  digitalWrite(MS2izq,HIGH);
  digitalWrite(dirMizq,LOW);      //LOW es hacia delante
  digitalWrite(StepMizq,LOW);
//La función sigue ejecutándose por aquí
}
```

Figura 5-9. Configuración en octavo de paso de los drivers.

Se recuerda nuevamente que el código de la figura 5-9 solo muestra la parte de la función *void setup()* donde se configuran los pines **MS1** y **MS2** de cada driver.

Para empezar a aprender a generar la señal PWM planteemonos el siguiente problema:

Sabiendo que se quiere aplicar una velocidad de 500º/s al motor, calcule el tiempo de paso sabiendo que el driver esta configurado en octavos de paso.

La solución a esto es sumamente sencilla:

$$\omega = 500°s^{-1} = \frac{0.225°}{T_{paso}}$$

De donde se deduce que:

$$T_{paso} = \frac{0.225°}{500°s^{-1}} = 0.00045s = 0.45ms$$

Es decir, para aplicar dicha velocidad basta con aplicar un paso cada 0.45ms.

Ahora que sabemos cada cuánto tiempo hay que aplicar un paso, la pregunta es ¿Cómo aplicar un paso cada Tpaso?

Aplicar un paso se resume básicamente en la aplicación de un flanco de subida en el pin STEP del *driver*, por lo que el tiempo de paso coincide con el tiempo entre flancos de subida.

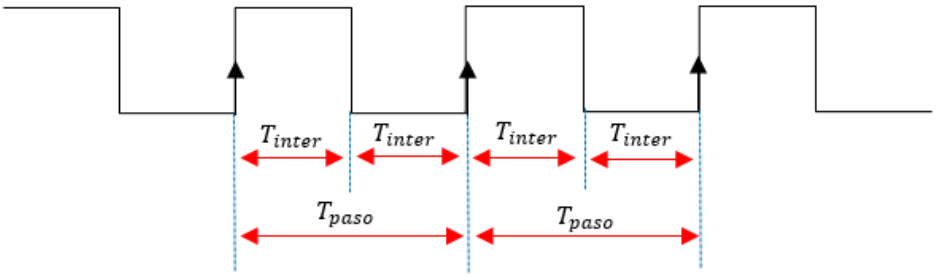


Figura 5-10. Relación tiempo de paso y tiempo de interrupción.

En la figura 5-10 se muestra la clave para generar la señal PWM. Debemos de generar una interrupción con un periodo igual a la mitad del tiempo de paso($Tinter = Tpaso/2$) y, en cada interrupción, aplicar una operación NOT en el pin correspondiente, es decir, si el valor es '1' lo pasamos a '0' y viceversa.

A continuación, en la figura 5-11 se muestra el código de las funciones de interrupción para cada motor, donde se muestra la función NOT sobre el pin **STEP** del driver correspondiente. Nótese que la operación NOT se realiza sola y exclusivamente sobre el bit del registro que corresponde con el pin **STEP**, dejando al resto de bits del registro sin modificar.

```
ISR(TIMER1_COMPA_vect){

  OCR1A+=p1;
  *out=((~*out)&bitMderStep)|((*out)&(~bitMderStep));
}

ISR(TIMER1_COMPB_vect){

    OCR1B+=p2;
    *out=((~*out)&bitMizqStep)|((*out)&(~bitMizqStep));
}
```

Figura 5-11. Funciones de interrupción A y B del Timer 1.

El último problema que se nos presenta es el de generar interruciones cada cierto tiempo igual a Tinter, y su solución es el uso de Timer 1 de Arduino.

Un Timer básicamente es un contador que, en el caso del Timer 1, permite contar desde 0 a $2^{16} - 1 = 65535$, ya que posee una resolución de 16 bits, al contario que el timer 0 y 2 del Atmega328p, los cuales poseen 8 bits. La frecuencia con la que actualiza el valor del contador es configurable a divisores (Preescalador) de 16MHz, siendo esta última la máxima posible. Planteemonos el siguiente ejercicio, con objeto de entender mejor lo que se ha comentado.

¿Cuánto tarda el Timer 1 en contar desde 0 hasta 65535 para cada uno de los divisores/preescaladores de la frecuencia del mismo?

En primer lugar, la variable que lleva la cuenta se llama TCNT1 (TCNT0 y TCNT2 para los otros dos Timer del Atmega328p), y en modo normal, cuenta desde 0 a 65535. Cuando alcanza el valor máximo, se resetea y comienza de nuevo a contar desde 0.

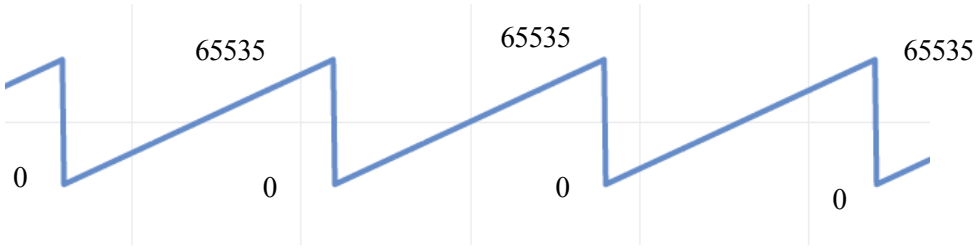


Figura 5-12. Forma de onda del valor TCNT1 para timer en modo Normal.

El ejercicio lo que nos pide es hallar el periodo de la anterior forma de onda para los diferentes preescaladores (divisores de la frecuencia fundamental 16MHz).

Como se puede observar en la siguiente figura, obtenida de datasheet del Atmega328p, los preescaladores que existen para el Timer 1 son: 1, 8, 64, 256 y 1024.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)

Tabla 5-3. Preescaladores del timer 1.

Matemáticamente, la solución a este ejercicio para cada uno de los preescaladores es la siguiente:

$$T_1 = \frac{1}{16000000} 65535s = 4{,}09593ms$$

$$T_8 = \frac{8}{16000000} 65535s = 32{,}7675ms$$

$$T_{64} = \frac{64}{16000000} 65535s = 0.26214s$$

$$T_{256} = \frac{256}{16000000} 65535s = 1{,}04856s$$

$$T_{1024} = \frac{1024}{16000000} 65535s = 4{,}19424s$$

Ahora ya sabemos configurar el preescalador para cambiar el periodo con el que se actualiza la cuenta de TCNT1. El objetivo ahora es el de entender significado y funcionamiento del registro de comparación OCR1A. Para ello, planteemonos el siguiente ejercicio.

¿Cuál debe ser el valor de OCR1A para generar interrupciones cada 1ms con el Timer1 en modo normal[3] y con preescalador 1 ? ¿Y si el preescalador fuera 8?

El registro de comparación OCR1A es un valor entre 0 y 65535 que esta continuamente comparándose con

[3] De las diferentes opciones de funcionamiento del Timer, el modo normal consiste en resetear TCNT1(ponerse a 0 la cuenta) cada vez que supere el valor de 65535. Otro modo sería por ejemplo el CTC (Clear Timer Compare), donde TCNT1 se resetea a 0 cuando se alcanza el valor de OCR1A.

TCNT1, de manera que, cuando coinciden, se dispara la función interrupción programada.

Matemáticamente, si el timer configurado con preescaler 1 aumenta la cuenta de TCNT1 cada $\frac{1}{16000000} s = 62{,}5 \times 10^{-9} s$, ¿Cuántas veces debe aumentar su cuenta para contar 1ms?

$$\frac{1}{16000000} s \times X = 0.001s$$

De donde se deduce que cada vez que cuente 16000 habrá pasado 1ms. Por tanto, ¿Qué valores debe de tomar OCR1A?

La primera interrupción se producirá para OCR1A=16000, la segunda para OCR1A=32000, la tercera para OCR1A=48000, la cuarta para OCR1A=64000, la quinta para OCR1A=64000+16000-65536=14464, etc. Véase la siguiente figura aclarativa.

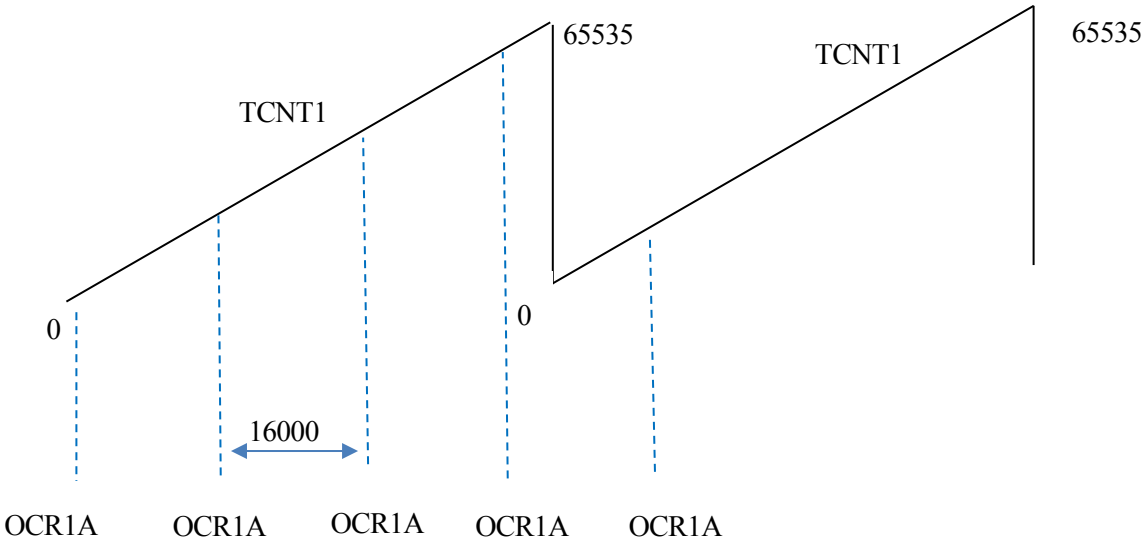


Figura 5-13. Actualización de OCR1A dentro de la función interrupción.

Por tanto, cada vez que salte la interrupción temporal, se debe de actualizar el valor del registro de comparación OCR1A a su siguiente valor. Puede observar la figura 5-13 y ver cómo se actualizan dichos valores para generar la próxima interrupción (*p1* y *p2* son las variables que incrementan OCR1A y OCR1B dentro del código).

Si procedemos de la misma manera con el preescalador de 8 se tiene que:

$$\frac{8}{16000000} s \times X = 0.001s$$

De donde se deduce debe contar 2000 para que transcurra 1ms.

Los valores de OCR1A asociados serian 0, 2000, 4000, 6000, 8000, etc.

Cuando resulte un valor de X con decimales, se debe de aproximar al entero más próximo, por lo que se cometerán errores y la velocidad real que se aplica no coincidirá con la teórica.

Llegados a este punto, ya sabemos cómo configurar el timer para que provoque una interrupción cada cierto tiempo T. Mantenga en mente que, tal como se ha visto en el ejercicio anterior, es posible generar interrupciones temporales mediante diferentes configuraciones del Timer, ya que será muy importante en lo que se verá a continuación. Como se verá, existe una configuración óptima del Timer.

Para continuar, nos planteamos el siguiente ejercicio.

¿Cómo aplicar una velocidad ω al motor? Calcule las velocidades máximas y mínimas que se podrían aplicar para cada preescalador.

Supongamos que queremos aplicar una velocidad alta ω=1500º/s . Si el motor está configurado en octavo de paso: 0.225º/paso, el tiempo de paso según vimos en la figura 5-9 es:

$$T_{paso} = \frac{0.225°/paso}{1500°/s} = 0.15ms/paso$$

Por tanto, el tiempo de interrupción debe ser de

$$T_{inter} = \frac{0.15}{2} ms = 0.075ms$$

Si utilizáramos el preescalador de 1(sin preescalador):

$$\frac{1}{16000000} s \times X = 0.075ms$$

$$X = 1200$$

Por lo que OCR1A seguiría la secuencia 0, 1200, 2400, 3600, etc.

Si utilizáramos el preescalador de 8 obtendríamos:

$$\frac{8}{16000000} s \times X = 0.075ms$$

$$X = 150$$

Lo que correspondería con la secuencia de OCR1A de 0, 150, 300, 450, etc.

Como se ve de nuevo, y se dijo anteriormente, no existe una única forma de configurar un Timer. Pronto se justificará cuál es la forma óptima. Ahora se verán los intervalos de velocidad teóricamente aplicables al motor para cada preescalador. Sus expresiones son las siguientes:

$$\omega^{máx} = \frac{0.225}{2 \times \frac{Preescalador}{16000000} \times 1}$$

$$\omega^{min} = \frac{0.225}{2 \times \frac{Preescalador}{16000000} \times 65535}$$

Siendo la expresión genérica:

$$\omega = \frac{0.225}{2 \times \frac{Preescalador}{16000000} \times X} \tag{5-2}$$

Tabla 5-4. Velocidades máxima y mínima teóricas para cada preescalador.

Preescalador	**ω máxima (º/s)**	**ω mínima(º/s)**
1	1.800.000	27,46
8	225.000	3,43
64	28.125	0.42
256	7.031,25	0,10
1024	1.757,81	0,026

La tabla 5-4 pone de manifiesto que una misma velocidad puede ser expresada configurando el timer con diferentes preescaladores. Para explicar cuál es la forma óptima, véase la siguiente figura.

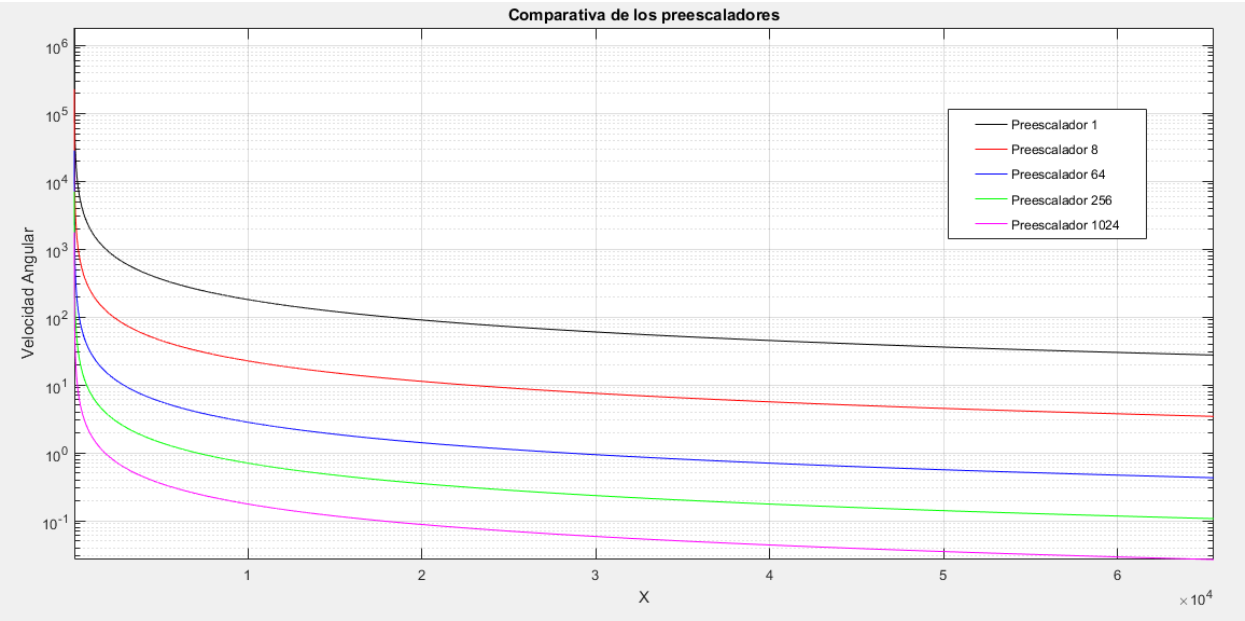


Figura 5-14. Función (5-2) aplicada a cada preescalador. Escala logarítmica en el eje OY.

Como se puede observar en la figura 5-14, la función (5-2) aplicada a cada preescalador presenta una asíntota vertical en el origen. Como se puede apreciar, para velocidades grandes y para un mismo X, el módulo de la derivada $\left|\frac{d\omega}{dx}\right|$ es mayor mientras mayor sea el preescalador. Por tanto, un mismo incremento ΔX, a

velocidades grandes, provoca un incremento de velocidad $\Delta\omega$ que es mayor mientras mayor sea el preescalador. Esto quiere decir que, lo óptimo para aplicar una velocidad, es utilizar el preescalador más chico posible, de tal manera que se aumente la precisión.

Por tanto, y como resumen, para aplicar una cierta velocidad hay que seguir estos dos pasos:

1. Configurar el sentido de los motores según lo visto en el punto 5.2.4.1.
2. Para generar la PWM, intentar utilizar el preescalador 1 para expresar dicha velocidad. Si no es posible, intertarlo con el preescalador de 8. Si no es posible, con el de 64. Y así hasta con el de 1024.

5.2.4.3 El problema del preescalador único

Imaginemos que queremos aplicar velocidades diferentes a cada rueda. Si seguimos los anteriores pasos se puede dar el caso de que el preescalador óptimo para la velocidad en la rueda derecha sea diferente del preescalador óptimo en la rueda izquierda. Entonces se nos plantea la pregunta: si el Timer solo posee un preescalador, ¿Cuál elegimos?

Esta pregunta posee dos posibles respuestas:

- Si solo queremos que nuestro vehículo realice un movimiento unidireccional, moviéndose hacia delante y hacia detrás, no existe problema alguno, ya que las dos ruedas girarán a la misma velocidad y, por tanto, con el mismo preescalador. Dicha solución se muestra el el punto 5.2.4.4.
- Si se desean aplicar velocidades diferentes en cada rueda para hacer girar al vehículo, se puede configurar el timer con un preescalador fijo, y aplicar velocidades en el rango de dicho preescalador. Notesé que para grandes velocidades disminuirá la precisión, como ya se demostró en el punto anterior. En el punto 5.2.4.5 se muestra como se ha resuelto lo anterior.

A continuación, se muestran las soluciones adoptadas para cada uno de los casos.

5.2.4.4 Solución para movimiento unidireccional

Para movimiento unidireccional, como se ha dicho, se tomará la solución óptima, por lo que la velocidad angular se generará a partir del preescalador más pequeño que permita la aplicación de dicha velocidad.

A continuación, se muestra el código de la función *aplica_Uk()* particularizado para movimiento unidireccional. Solo comentar dos puntos importantes:

- La velocidad angular se satura entre los limites alcanzables. La velocidad mínima aplicable se obtiene de la tabla 5-4 para el preescalador 1024 y es 0.027º/s. La velocidad máxima alcanzable se supone de 3000º/s (Valor experimental).
- Antes de configurar los registros de configuración del Timer y de cambiar el sentido de giro de los motores, es necesario deshabilitar las interrupciones. Esto es debido a que se puede dar la situación en la que se configure el registro OCR1A de la rueda derecha antes que el registro OCR1B de la rueda izquierda y, en ese tiempo, una rueda gire a una velocidad diferente de la otra, haciendo que el vehíclo rote ligeramente en torno a la vertical. Deshabilitando las interrupciones con *cli()* y habilitándolas posteriormente con *sei()* nos quitamos este problema.

```
void aplica_Uk(void){

 if(Wder_k>0&&Wizq_k>0){
        Wder_k=constrain(Wder_k,0.027,3000);   //Saturamos la velocidad
entre los limites alcanzables
        Wizq_k=constrain(Wizq_k,0.027,3000);   //0.027 viene de
0.225/(65535*1024*2/16000000)~0.02682°/s: velocidad minima la
rueda(Preescalador de 1024 contando hasta 65535)

        TimeInterrDer=(0.225/(Wder_k*2))*1000000;   //Calculamos el
tiempo entre interrupciones para cada rueda.
        TimeInterrIzq=(0.225/(Wizq_k*2))*1000000;   //El tiempo de
interrupcion es la mitad del tiempo de paso

        if(TimeInterrDer<=TIMER_ONE_RESOLUTION/16.0){   //¿Podemos contar
TimeInterrDer(uS) sin preescalador?
          p1_copy=TimeInterrDer*16;                     //Solo posible
para Velocidades mayores de 0.225/(65535*2/16000000)~27,46°/s
          p2_copy=TimeInterrIzq*16;
          cli();
          *out&=~bitMizqdir;
          *out|=bitMderdir;
          p1=p1_copy;
          p2=p2_copy;
          TCCR1B=(1 << CS10);  //Sin Preescalador
          OCR1A=TCNT1+p1;
          OCR1B=TCNT1+p2;
          sei();
        }
        else{
          if(TimeInterrDer<=TIMER_ONE_RESOLUTION*8.0/16.0){ //¿Podemos
contar TimeInterrDer(uS) con preescalador de 8?
            p1_copy=TimeInterrDer*16/8;                     //Solo
posible para Velocidades mayores de 0.225/(65535*8*2/16000000)~3.43°/s
            p2_copy=TimeInterrIzq*16/8;
            cli();
            *out&=~bitMizqdir;
            *out|=bitMderdir;
            p1=p1_copy;
            p2=p2_copy;
            TCCR1B=(1 << CS11);  //Preescalador de 8
            OCR1A=TCNT1+p1;
            OCR1B=TCNT1+p2;
            sei();
          }
          else{
            if(TimeInterrDer<=TIMER_ONE_RESOLUTION*64.0/16.0){    //¿Pode
mos contar TimeInterrDer(uS) con preescalador de 64?
              p1_copy=TimeInterrDer*16.0/64.0;                    //Solo
posible para Velocidades mayores de 0.225/(65535*64*2/16000000)~0.42°/s
              p2_copy=TimeInterrIzq*16.0/64.0;
              cli();
              *out&=~bitMizqdir;
              *out|=bitMderdir;
              p1=p1_copy;
              p2=p2_copy;
              TCCR1B=(1 << CS10)|(1 << CS11);  //Preescalador de 64
              OCR1A=TCNT1+p1;
              OCR1B=TCNT1+p2;
              sei();
            }
```

```
            else{
               if(TimeInterrDer<=TIMER_ONE_RESOLUTION*256.0/16.0){    //¿P
odemos contar TimeInterrDer(uS) con preescalador de 256?
                 p1_copy=TimeInterrDer*16.0/256.0;                    //So
lo posible para Velocidades mayores de
0.225/(65535*256*2/16000000)~0.10°/s
                 p2_copy=TimeInterrIzq*16.0/256.0;
                 cli();
                 *out&=~bitMizqdir;
                 *out|=bitMderdir;
                 p1=p1_copy;
                 p2=p2_copy;
                 TCCR1B=(1 << CS12);  //Preescalador de 256
                 OCR1A=TCNT1+p1;
                 OCR1B=TCNT1+p2;
                 sei();
               }
               else{                                                  //¿Po
demos contar TimeInterrDer(uS) con preescalador de 1024?
                  p1_copy=TimeInterrDer*16.0/1024.0;                  //Sol
o posible para Velocidades mayores de
0.225/(65535*1024*2/16000000)~0.027°/s
                  p2_copy=TimeInterrIzq*16.0/1024.0;
                  cli();
                  *out&=~bitMizqdir;
                  *out|=bitMderdir;
                  p1=p1_copy;
                  p2=p2_copy;
                  TCCR1B=(1 << CS12)|(1 << CS10);  //Preescalador de 1024
                  OCR1A=TCNT1+p1;
                  OCR1B=TCNT1+p2;
                  sei();
               }
            }
          }
        }
  }
  else{
      if(Wder_k<0&&Wizq_k<0){
            Wder_k=constrain(Wder_k,-3000,-0.027);
            Wizq_k=constrain(Wizq_k,-3000,-0.027);
           TimeInterrDer=(-1)*(0.225/(Wder_k*2))*1000000;   //Calculamos
el tiempo entre interrupciones para cada rueda.
            TimeInterrIzq=(-1)*(0.225/(Wizq_k*2))*1000000;   //El tiempo
de interrupcion es la mitad del tiempo de paso

            if(TimeInterrDer<=TIMER_ONE_RESOLUTION/16.0){   //¿Podemos
contar TimeInterrDer(uS) sin preescalador?
              p1_copy=TimeInterrDer*16;
              p2_copy=TimeInterrIzq*16;
              cli();
              *out&=~bitMderdir;
              *out|=bitMizqdir;
              p1=p1_copy;
              p2=p2_copy;
              TCCR1B=(1 << CS10);  //Sin Preescalador
              OCR1A=TCNT1+p1;
```

```
             OCR1B=TCNT1+p2;
              sei();
            }
            else{
              if(TimeInterrDer<=TIMER_ONE_RESOLUTION*8.0/16.0){
//¿Podemos contar TimeInterrDer(uS) con preescalador de 8?
                p1_copy=TimeInterrDer*16/8;
                p2_copy=TimeInterrIzq*16/8;
                cli();
                *out&=~bitMderdir;
                *out|=bitMizqdir;
                p1=p1_copy;
                p2=p2_copy;
                TCCR1B=(1 << CS11);  //Preescalador de 8
                OCR1A=TCNT1+p1;
                OCR1B=TCNT1+p2;
                sei();
              }
              else{
                if(TimeInterrDer<=TIMER_ONE_RESOLUTION*64.0/16.0){    //¿
Podemos contar TimeInterrDer(uS) con preescalador de 64?
                  p1_copy=TimeInterrDer*16.0/64.0;
                  p2_copy=TimeInterrIzq*16.0/64.0;
                  cli();
                  *out&=~bitMderdir;
                  *out|=bitMizqdir;
                  p1=p1_copy;
                  p2=p2_copy;
                  TCCR1B=(1 << CS10)|(1 << CS11);  //Preescalador de 64
                  OCR1A=TCNT1+p1;
                  OCR1B=TCNT1+p2;
                  sei();
                }
                else{
                  if(TimeInterrDer<=TIMER_ONE_RESOLUTION*256.0/16.0){
//¿Podemos contar TimeInterrDer(uS) con preescalador de 256?
                    p1_copy=TimeInterrDer*16.0/256.0;
                    p2_copy=TimeInterrIzq*16.0/256.0;
                    cli();
                    *out&=~bitMderdir;
                    *out|=bitMizqdir;
                    p1=p1_copy;
                    p2=p2_copy;
                    TCCR1B=(1 << CS12);  //Preescalador de 256
                    OCR1A=TCNT1+p1;
                    OCR1B=TCNT1+p2;
                    sei();
                  }
                  else{                                                 /
/¿Podemos contar TimeInterrDer(uS) con preescalador de 1024?
                      p1_copy=TimeInterrDer*16.0/1024.0;
                      p2_copy=TimeInterrIzq*16.0/1024.0;
                      cli();
                      *out&=~bitMderdir;
                      *out|=bitMizqdir;
                      p1=p1_copy;
                      p2=p2_copy;
                      TCCR1B=(1 << CS12)|(1 << CS10);  //Preescalador de
```

```
                                                                    //1024
                        OCR1A=TCNT1+p1;
                        OCR1B=TCNT1+p2;
                        sei();
                    }
                }
            }
        }
    }
}
```

Figura 5-15. Código de la función *aplica_Uk()* para movimiento unidireccional.

Recuérdese que las funciones de interrupción se muestran en la figura 5-11.

Para mayor entendimiento, los registros p1 y p2 son las cantidades que van actualizando OCR1A y OCR1B para ir generando las interrupciones (Véase figura 5-11).

Dentro del registro TCCR1B se configura el valor del preescalador según la tabla 5-3.

Importantísimo: se inicializan los valores de OCR1A y OCR1B al valor actual de la cuenta TCNT1 más *p1* y *p2* para que la primera interrupción se produzca un tiempo Tinter (tal y como debe ser) tras la ejecución del de *aplica_Uk()*. Si no se hace esto, es impredecible el momento en el que se dará el primer flanco de la PWM.

5.2.4.5 Solución para movimiento plano

En la programación anterior se puede observar como resulta innecesarío trabajar con variables diferentes para cada rueda ya que ambas poseen el mismo valor. Eso se hace así para mayor claridad en su función.

En este apartado se usará una única variable para representar el valor teórico de la velocidad angular de ambas ruedas (como si la trayectoria fuera unidireccional). El ejecutivo cíclico que se muestra a continuación es equivalente al mostrado en la figura 5-3, sin más que realizar el cambio de trabajar con dos variables a una.

Una vez que tenemos la velocidad angular teórica que deben tomar las ruedas durante cada tramo del ciclo de muestreo, se introduce una varable llamada *giro,* la cual introduce una perturbación en la obtención de las velocidades de la rueda izquierda y rueda derecha *Wder_k* y *Wizq_k*.

En la función *aplica_Uk* de la figura 5-17 se muestra el código de dicha función. Como se puede ver, el preescalador está configurado a 8, tal y como se dijo anteriormente.

¿Qué hace la función?

1. Satura el valor de la velocidad teórica a aplicar entre el máximo y el mínimo valor aplicable. Dicho valor mínimo depende del preescalador que, para el caso de valor 8, la velocidad mínima toma el valor 3.45º/s.

2. Para conseguir girar:
    - Si el vehículo se mueve hacia delante:
        - Si giro es mayor que 0 : Wizq_k=W_k*(1+giro) y Wder_k=W_k.
        - Si giro es menor que 0: Wder_k=W_k*(1-giro) y Wizq_k=W_k.
    - Si el vehículo se mueve hacia detrás:
        - Si giro es mayor que 0: Wizq_k=W_k*(1+giro) y Wder_k=W_k.
        - Si giro es menor que 0: Wder_k=W_k*(1-giro) y Wizq_k=W_k.

Con esta forma de proceder se consigue que las velocidades de ambas ruedas sean expresables con el preescalador 8. Es decir, la rueda con la mínima velocidad en valor absoluto será igual a W_k, cuyo valor esta previamente saturado entre los límites alcanzables.

```
void loop() {

  static char aux=1,reff=0,gir=0;
  while(flag==0);
  flag=0;
  switch(ciclo){
    case 1:
      leer_sensores();
      controlador();
      W_k=W_k1+0.2*Alfa*Tm;
      aplica_Uk();
    break;

    case 2:
      W_k=W_k1+0.4*Alfa*Tm;
      aplica_Uk();
   break;

    case 3:
      W_k=W_k1+0.6*Alfa*Tm;
      aplica_Uk();
   break;

    case 4:
      W_k=W_k1+0.8*Alfa*Tm;
      aplica_Uk();
   break;

    case 5:
     W_k=W_k1+Alfa*Tm;
      aplica_Uk();
   break;
  }
}
```

Figura 5-16. Ejecutivo cíclico en función de la aceleración de la base.

```
void aplica_Uk(void){

 if(W_k>0){

        W_k=constrain(W_k,3.44,3000);          //Saturamos la velocidad
entre los limites alcanzables
                                                //0.027 viene de
0.225/(1024*2/16000000)~0.02682°/s: velocidad minima la
rueda(Preescalador de 1024 contando hasta 65535)
        if(giro>0){
           Wizq_k=W_k*(1+giro);
           Wder_k=W_k;
        }
        else{
           Wizq_k=W_k;
           Wder_k=W_k*(1-giro);
        }
       TimeInterrDer=(0.225/(Wder_k*2))*1000000;   //Calculamos el
tiempo entre interrupciones para cada rueda.
        TimeInterrIzq=(0.225/(Wizq_k*2))*1000000;   //El tiempo de
interrupcion es la mitad del tiempo de paso
        p1_copy=TimeInterrDer*16/8;
        p2_copy=TimeInterrIzq*16/8;
        cli();
        *out&=~bitMizqdir;
        *out|=bitMderdir;
        p1=p1_copy;
        p2=p2_copy;
        OCR1A=TCNT1+p1;
        OCR1B=TCNT1+p2;
        sei();
 }
  else{
      W_k=constrain(W_k,-3000,-3.44);
      if(giro>0){
         Wizq_k=W_k*(1+giro);
         Wder_k=W_k;
      }
      else{
         Wizq_k=W_k;
         Wder_k=W_k*(1-giro);
      }
      TimeInterrDer=(-1)*(0.225/(Wder_k*2))*1000000;   //Calculamos el
tiempo entre interrupciones para cada rueda.
      TimeInterrIzq=(-1)*(0.225/(Wizq_k*2))*1000000;   //El tiempo de
interrupcion es la mitad del tiempo de paso
      //¿Podemos contar TimeInterrDer(uS) con preescalador de 8?
      p1_copy=TimeInterrDer*16/8;
      p2_copy=TimeInterrIzq*16/8;
      cli();
      *out&=~bitMderdir;
      *out|=bitMizqdir;
      p1=p1_copy;
      p2=p2_copy;
      OCR1A=TCNT1+p1;
      OCR1B=TCNT1+p2;
      sei();
  }
}
```

Figura 5-17. Función *aplica_Uk* para habilitar giro con preescalador constante e igual a 8.

Por último, decir que en los ficheros *two_wheeled_inverted_pendulum.ino* y *one_wheeled_inverted_pendulum.ino* se pueden encontrar los códigos completos para los dos tipos de movimiento que hemos visto: movimiento unidireccional y movimiento plano. Dichos códigos se dejarán con más comentarios en los márgenes que los vistos en las figuras de este capítulo.

5.2.4.6 El problema de las velocidades lentas

Recordemos la figura 5-2. En ella, se dividió el periodo de muestreo de 20ms en 5 tramos de 4ms y, en cada tramo, se aplicaba una velocidad diferente con objeto de aproximar la señal de velocidad dada por una aceleración constante. Dado que generar una velocidad se resume en generar una señal PWM cuya frecuencia es mayor cuán mayor es la velocidad a aplicar, ocurre un problema para velocidades bajas.

Si la velocidad es alta, se generan numerosos flancos de subida dentro del tramo de 4ms haciendo que la aproximación dada por la figura 5-2 sea correcta, pero, cuando la la velocidad es pequeña, el periodo de la señal PWM se vuelve mayor que los 4 ms del tramo, por lo que no se generan flancos de subida que provoquen pasos.

Por tanto, existe una zona muerta en la aplicación de velocidad al motor. Por ello, cuando se analicen los resultados de los diferentes controladores para diferentes referencias en velocidad, mientras mayor sea la velocidad de referencia mejor será el control ó, de la forma contraria, mientras menor sea la referencia, peor será el control.

El objetivo del siguiente punto es el diseño e implementación de controladores sobre nuestro sistema.

Aquí acaba el capítulo 6 *Programación del microcontrolador*.

6 CONTROL DEL SISTEMA

El que quiere algo conseguirá un medio, el que no una escusa

Stephen Dolley

En este cápítulo se diseñarán diferentes controladores para seguimiento de referencias en velocidad, y se implementarán sobre el sistema real. En particular, se han desarrollado tres estrategias de control diferentes:

- Controlador PID para el ángulo con la vertical y controlador PID para la velocidad del vehículo mediante técnicas heurísticas.
- Controlador LQR a partir del modelo dinámico en el espacio de estados.
- Controlador LQR con efecto integral a partir del modelo dinámico en el espacio de estados.

La primera estrategia de control se desarrolló, como bien se dijo en el capítulo 3, para recoger datos para estimar los parámetros del modelo. Las otras dos estrategias de diseñan a partir del modelo obtenido en dicho capítulo.

En este capítulo se mostrará cómo se han diseñado los controladores LQR y LQR con efecto integral. Una vez diseñados, se pondrán a prueba bajo numerosas situaciones; seguimiento de referencias en velocidad; perturbaciones tipo pulso a favor del movimiento; perturbaciones tipo pulso en contra del movimiento; etc.

6.1 Control en cascada

Mediante técnicas heurísticas, se sintoniza un PID para el control de velocidad de tal manera que, recibiendo el error en velocidad, calcula el ángulo con la vertical que será la referencia en angulo para otro controlador PID. Este último, calcula la aceleracíon de las ruedas que hay que aplicar para seguir la referencia en ángulo que nos proporciona el anterior PID.

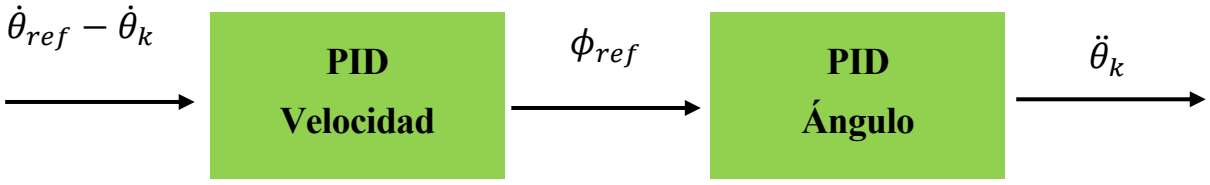


Figura 6-1. Control en cascada.

Para explicar la manera de proceder, se han seguido los siguientes pasos:

1. Sintonización del primer PID para seguimiento de referencias en ángulo.

   1.1 Cálculo de Kp de tal manera que mantenga de pie al vehículo durante el máximo tiempo posible.

   2.2 Añadir el término integral disminuyendo Ti desde un valor muy grande (Término integral nulo) hasta que se mejoren los resultados obtenidos con el paso anterior.

   3.3 Con objeto de añadir un término de carácter predictivo, incrementar desde 0 el valor de Td (Término derivativo nulo) hasta obtener mejores resultados que en el punto anterior. El término derivativo, además, aumenta la robustez del sistema ante perturbaciones externas como las de tipo pulso (Golpe seco sobre el vehículo).

2. Sintonización del segundo PID para cálculo de referencias en ángulo a partir del error en velocidad.

   2.1 Para una referencia nula en velocidad, sintonizar Kpv de tal manera que nuestro vehículo se mantenga lo más estático posible.

   2.2 Añadir un término integral Tiv que mejore los resultados del punto anterior. Este término bien escogido provocará que, dada una perturbación en una dirección (Golpe), el vehículo se pare rápido gracias a que durante ese periodo la integral del error crece rápidamente.

   2.3 Con objeto de disminuir las oscilaciones en velocidad, añadir un término de carácter predictivo Td que consiga tal objetivo.

Como nota, tenga mucha paciencia sintonizando los diferentes parámetros, ya que este proceso es muy lento y desesperante.

6.1.1 Seguimiento de referencia nula en velocidad

A continuación, se muestran los resultados para seguimiento de referencia en velocidad nula.

Se recuerda la existencia de una zona muerta para velocidades bajas, tal como se demostró en el capítulo 5. Esto provoca que nunca sea posible alcanzar la referencia exacta de 0º/s. De hecho, las oscilaciones que se observan en la siguiente figura poseen aproximadamente la misma amplitud que con los controladores LQR y LQR con efecto integral que se verán posteriormente.

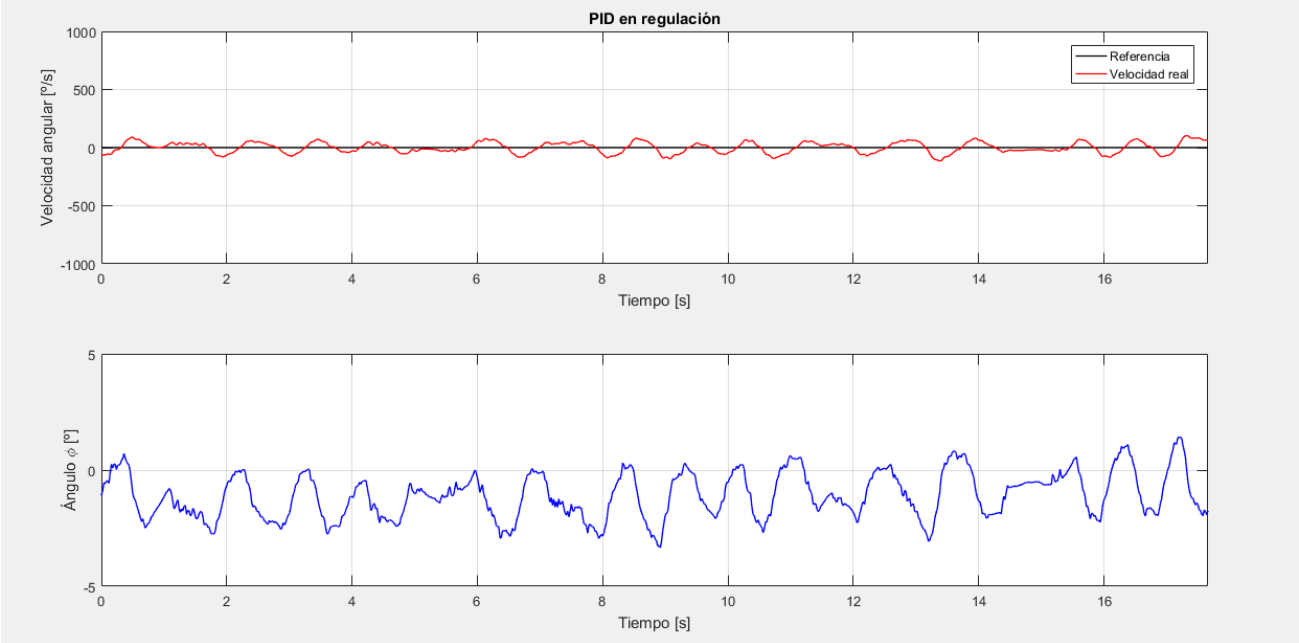


Figura 6-2. Seguimiento de referencia nula en velocidad.

6.1.2 Seguimiento de referencia nula en velocidad con perturbaciones tipo pulso

En la siguiente imagen, para una referencia nula en velocidad, se le aplican perturbaciones (golpes en la parte delantera y trasera del vehículo) para ver si el controlador es capaz de estabilizar de nuevo al sistema. Note el lector que el vehículo es capaz de alcanzar casi los 40º de inclinación y estabilizarse de nuevo.

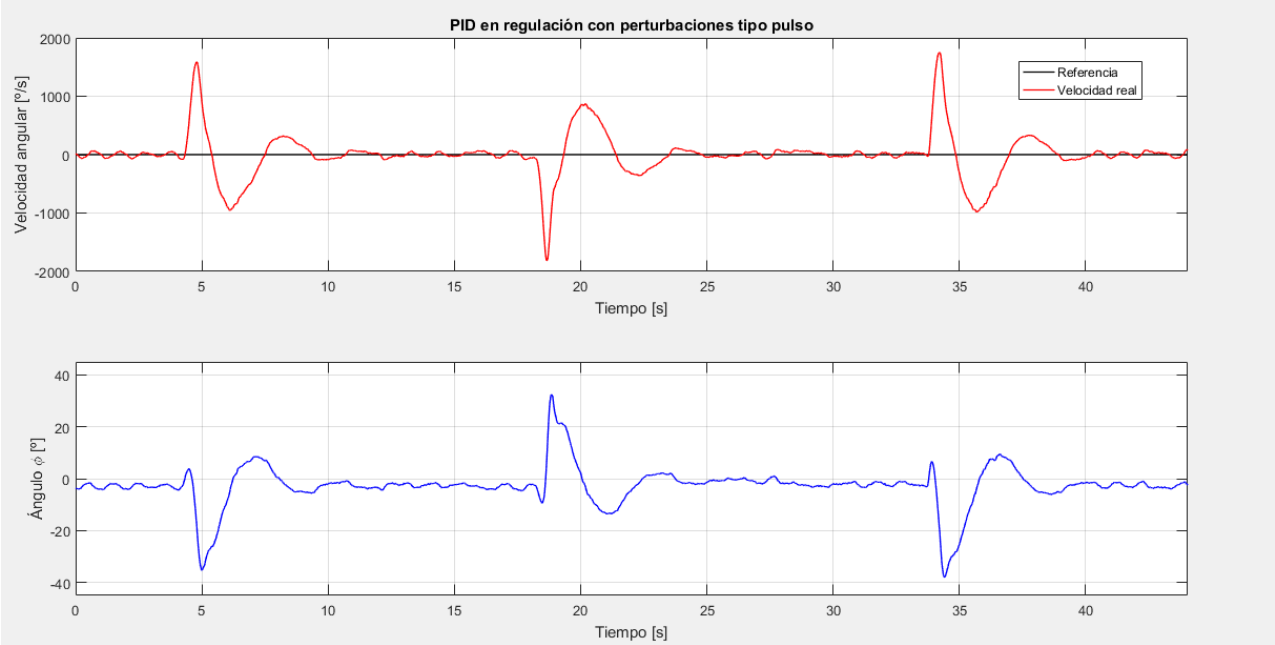


Figura 6-3. Seguimiento de referencia nula con pertubaciones.

6.1.3 Seguimiento de referencias en velocidad

A continuación, se envían diferentes referencias en velocidad al vehículo para ver si el controlador es capaz alcanzarlas. Los resultados obtenidos se muestran en la siguiente figura, donde se puede observar que no son del todo bueno en comparación con los obtenidos con las próximas estrategias de control.

- Cambios muy bruscos al cambiar de referencia en velocidad.
- Buenos resultados en regulación, no tan buenos en seguimiento (Demasiadas oscilaciones).

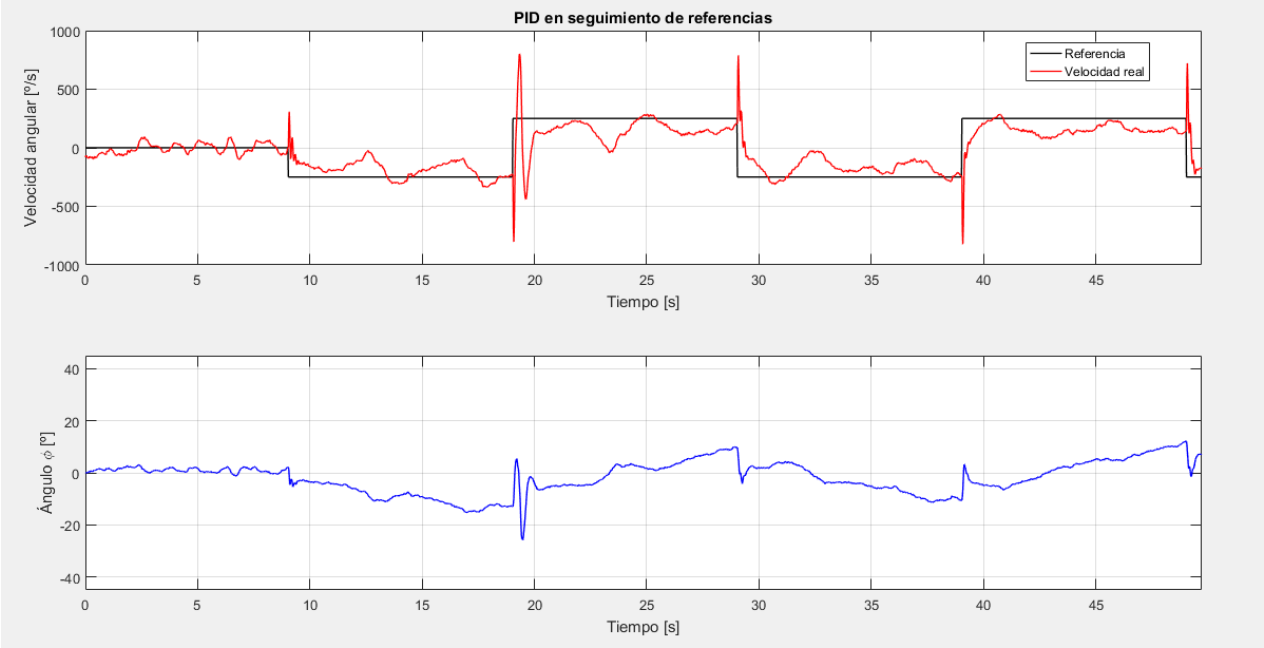


Figura 6-4. Seguimiento de referencias en velocidad.

6.2 Controlador LQR

En este punto se diseñará un controlador mediante realimentación lineal del vector de estados que proporcione mejores resultados que los obtenidos con el control en cascada.

A continuación, se mostrará la base teórica del controlador, se mostrará un programa en Matlab que calcula la ganacia de realimentación K y, se mostrarán los resultados obtenidos bajo diferentes situaciones. También se demuestra la validez del periodo de muestreo obtenido.

6.2.1 Base matemática del controlador óptimo LQR

Dado el modelo dinámico del sistema de la ecuacíon (2-74) en su forma discreta para Tm=0.02s,

$$x_{k+1} = Ax_k + Bu_k \tag{6–1}$$

y dado el funcional J,

$$J = \sum_{\forall k} (x_k - x_e)^T Q (x_k - x_e) + (u_k - u_e)^T R (u_k - u_e) \tag{6–2}$$

donde Xe es el punto de equilibrio dado por la ecuación (2-65) y Ue la acción de control en dicho punto (0º/s^2), se calculará el vector de realimentación K que:

- Haga que el sistema sea estable.
- Minimice el valor del funcional J.

Dicha ley de control tiene la forma dada por (6-3).

$$u_k = -K(x_k - x_e) + u_e \tag{6–3}$$

El funcional J, donde Q y R son matrices semidefinidas positivas (Por tanto, J>0), mide el coste tanto de la aplicación de la acción de control (R) como el coste de la evolución del estado a Xe.

Dado que Xe es un punto de equilibrio, se verifica que

$$x_e = Ax_e + Bu_e \tag{6–4}$$

Haciendo el siguiente cambio de variable y operando

$$z_{k+1} = x_{k+1} - x_e \tag{6–5}$$

$$z_{k+1} = Ax_k + Bu_k - x_e \tag{6-6}$$

$$z_{k+1} = A(x_k - x_e + x_e) + B(u_k) - x_e \tag{6-7}$$

Si aplicamos como ley de control la realimentación lineal dada por (6-3) se obtiene junto con (6-7):

$$z_{k+1} = Az_k + Ax_e + B(-Kz_k + u_e) - x_e \tag{6–8}$$

$$z_{k+1} = Az_k - BKz_k + \underbrace{Ax_e + Bu_e - x_e}_{=0} \tag{6-9}$$

Al final nos queda:

$$z_{k+1} = (A - BK)z_k \tag{6–10}$$

$$J = \sum_{\forall k} {z_k}^T (A - BK)\, z_k \tag{6-11}$$

La solución del diseño es el cálculo de la ganancia de realimentación K que

- Haga que el sistema sea estable (Que Zk tienda al origen implica que Xk tienda a nuestro punto de equilibrio).
- Minimice el valor del funcional J.

6.2.2 Obtención de la ganancia de realimentación K con MATLAB

El siguiente programa en MATLAB cálcula, a partir de los parámetros del modelo, y para unas matrices semidefinidas positivas Q y R dadas, la ganancia de realimentación K.

Nota, escoger unos valores de Q y R que hagan que los órdenes de magnitud de las componentes del vector de realimentación K sean parecidos a los obtenidos en el diseño de los dos controladores PID del apartado anterior.

```
%Parametros del modelo

M=0.945;          %Masa del solido 2 en Kg
mr=0.070;         %Masa de la rueda en Kg
R=0.05;           %Radio de las ruedas en m
L=0.05;           %Distancia desde la base ejes solido 2 hasta el centro de masas del solido 2 en
metros
g=9.81;           %Aceleración de la gravedad ms^-2
Iyy=0.00994356;   %Inercia del solido 2 respecto de los ejes 2 Kgm^2
Ir=0.00098421;    %Inercia de las ruedas respecto al eje del motor kgm^2

%Sistema en tiempo continuo
A=[0 1 0;(M*g*L)/(Iyy+M*R*L) 0 0;0 0 0];
B=[0;-(2*Ir+2*mr*R^2+M*R^2+M*R*L)/(Iyy+M*R*L);1];
C=[0 0 1];
D=0;

SYS=ss(A,B,C,D); %Se crea el modelo en el espacio de estados en continuat
SYSD = c2d(SYS,0.02); %Se discretiza el anterior con Tm=0.02s
Q=[1 0 0;0 1 0;0 0 1];
R=0.025;
[K]=lqr(SYS,Q,R) %Se calcula el vector de realimentación K
```

Figura 6-5. Obtención del vector de realimentación K con MATLAB.

6.2.3 Demostración de la validez del tiempo de muestreo empleado

Si, con el vector de realimentación del vector de estados K obtenido con el programa de la anterior figura (6-5) calculamos los autovalores del sistema en bucle cerrado (A-BK), obtenemos:

$$K = [-268.7722 \quad -44.2357 \quad -6.3246] \tag{6–12}$$

$$Autovalores = \begin{bmatrix} -9.12 \\ -4.93 + 1.32i \\ -4.93 - 1.32i \end{bmatrix} \tag{6-13}$$

Del cálculo de los autovalores se deduce que la constante de tiempo más rápida toma el valor

$$\Gamma = \frac{1}{9.12} \approx 0.10s \tag{6–14}$$

Por tanto, el periodo de muestreo de 0.02ms es válido ya que es mucho menor que la dinámica más rápida del sistema.

6.2.4 Resultados obtenidos

En este apartado se controlará al vehículo con la ganancia de realimentación *K* dada por la ecuación (6-12). Se pondrá a prueba para multitud de situaciones y, además, para cada situación, se compararán los dos programas de control (Para movimiento unidireccional y para movimiento bidireccional) desarrollados en el capítulo 5, para ver si mejoran o empeoran el control.

6.2.4.1 Seguimiento de referencias en velocidad

En esta prueba, el vehículo tendrá que seguir referencias en la velocidad angular de las ruedas. Se mostrará la comparativa entre movimiento unidireccional y movimiento circular.

Nota: para el caso de movimiento circular, donde una rueda gira más que la otra, el control se realizará sobre la rueda que gira a menor velocidad.

Se podrá observar que, aunque los resultados son bastante mejores que para el control en cascada, el error en régimen permanente no termina de anularse. Con la estrategia de control LQR con efecto integral, como se verá mas adelante, se solucionará este problema.

Respecto a la comparativa entre el seguimiento de referencias en movimiento circular y rectilíneo, no existen diferencias significativas en los resultados.

A continuación, se muestran las gráficas.

6.2.4.1.1 Para movimiento rectilineo

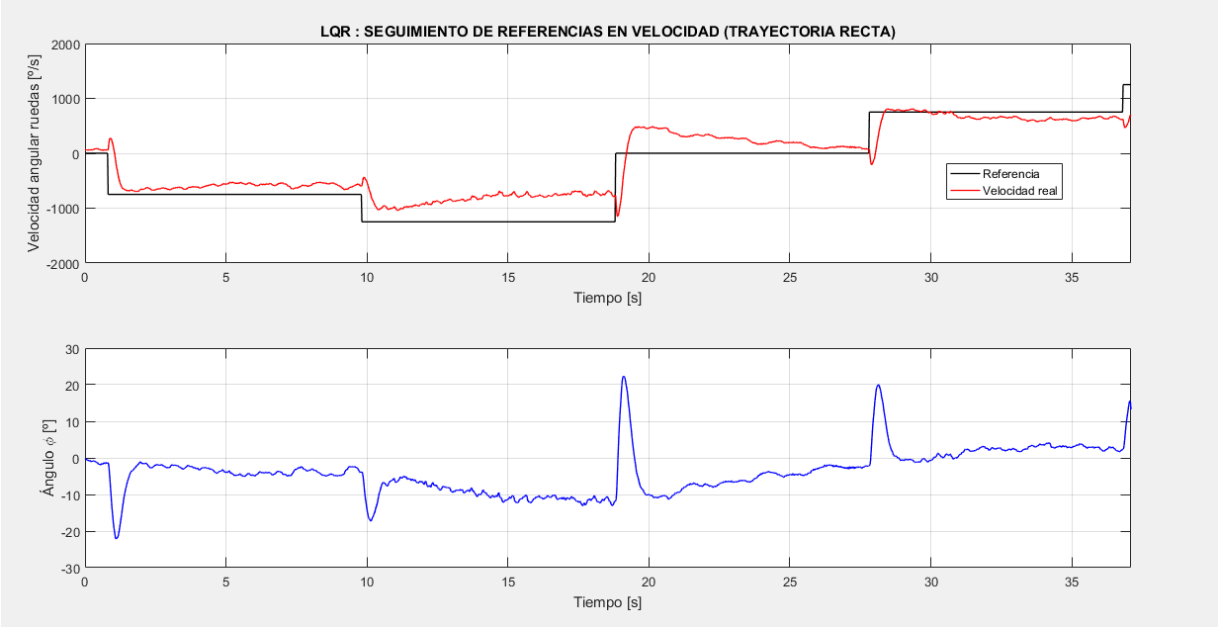


Figura 6-6. LQR: Seguimiento de referencias en velocidad (Trayectoria recta).

6.2.4.1.2 Para movimiento circular

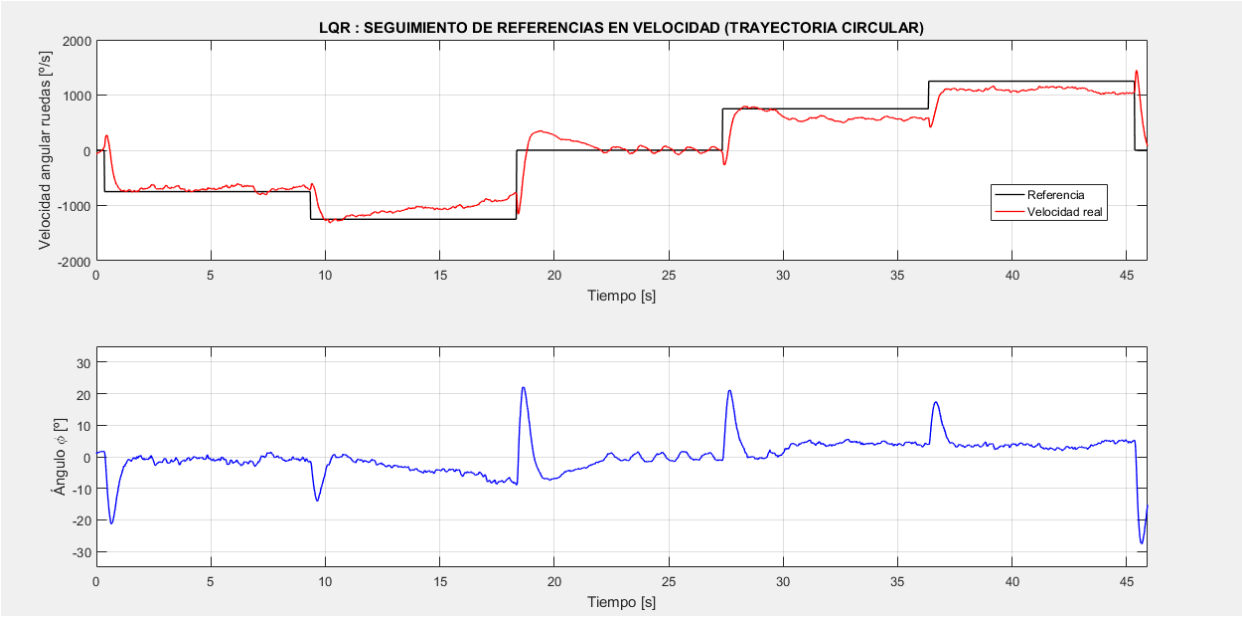


Figura 6-7. LQR: Seguimiento de referencias en velocidad (Trayectoria circular).

6.2.4.2 Seguimiento de referencias en velocidad con perturaciones tipo pulso

En esta prueba, introducimos perturbaciones tipo pulso (Golpes secos) sobre el sistema mientras este intenta seguir las referencias en velocidad.

Como se puede observar, el sistema de control es capaz de mantener el control sobre el sistema ante este tipo de perturbaciones.

Al ser el sistema de fase no mínima, el sistema tiende a acelerarse en la dirección de dicha perturbación para, posteriormente, poder reducir su velocidad.

6.2.4.2.1 Para movimiento rectilineo

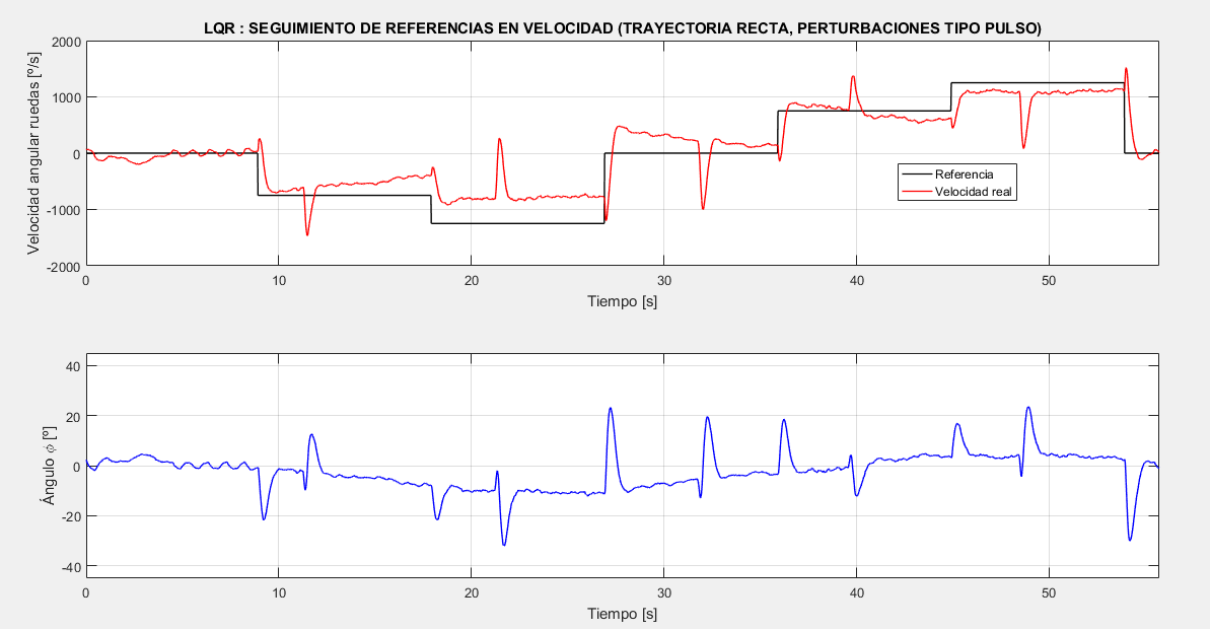


Figura 6-8. LQR: Seguimiento de referencias en velocidad (Trayectoria recta, perturbaciones tipo pulso).

6.2.4.2.2 Para movimiento circular

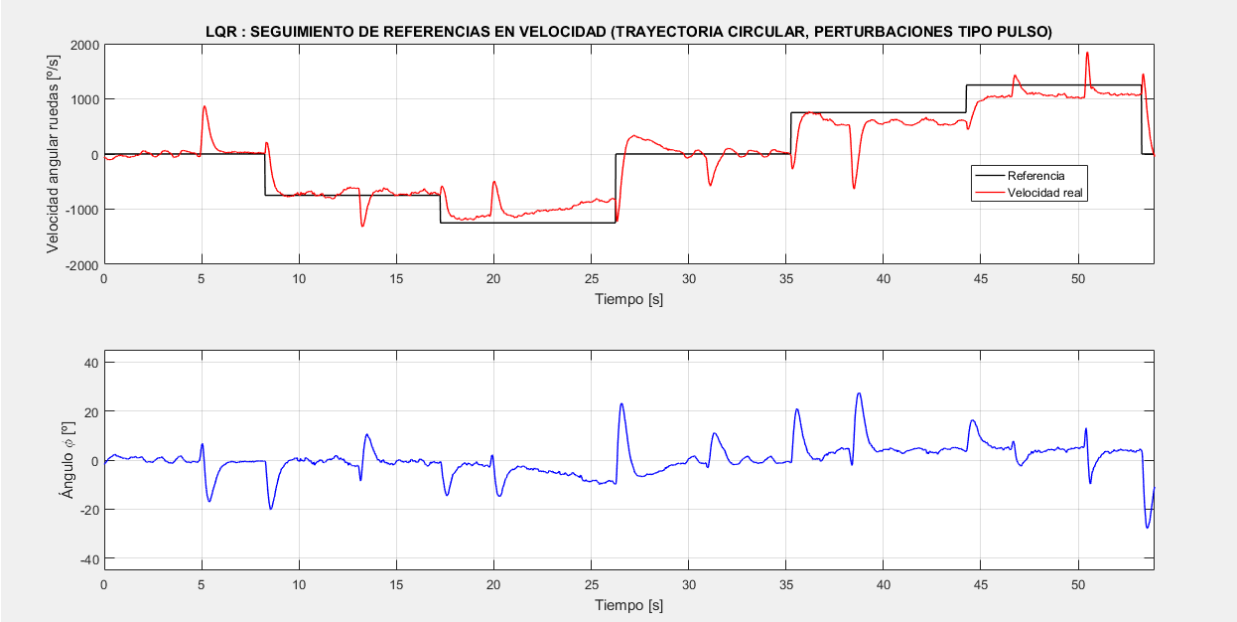


Figura 6-9. LQR: Seguimiento de referencias en velocidad (Trayectoria circular, perturbaciones tipo pulso).

6.2.4.3 Seguimiento de referencia en velocidad nula con perturbación de tipo fuerza constante

En esta prueba, se aplicará una fuerza constante sobre el vehículo, y se verá como responde el controlador a tal estímulo. La trayectoria será curva.

Véase como, ante una perturbación de este tipo, el sistema evoluciona trasladándose en la dirección de la perturbación. Una vez deja de actuar, el sistema se estabiliza en el lugar al que ha sido desplazado.

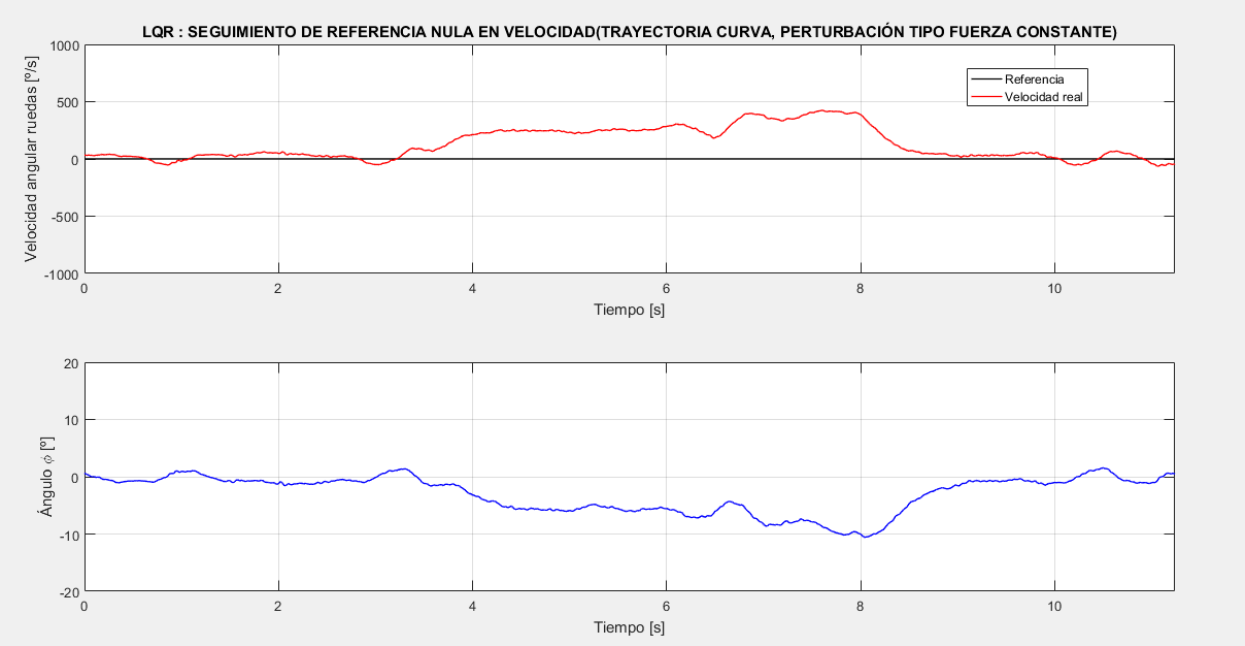


Figura 6-10. LQR: Referencia nula(Trayectoria circular, perturbación de fuerza constante).

6.3 Controlador LQR con efecto integral

Como se ha visto, el anterior controlador LQR no tiene en cuenta que existe siempre una pequeña desviación entre el centro de masas y el eje de simetría. Este hecho se considera como una perturbación que provoca que el sistema nunca llegue a estabilizarse en torno a la vertical. Además, la zona muerta en la aplicación de la velocidad provoca que el sistema no se quede completamente estático.

En este punto se diseñará un controlador LQR con efecto integral, el cual tendrá en cuenta la desviación existente entre el centro de masas y el eje de simetría: el propio controlador inclinará al vehículo de tal manera que intentará colocar al centro de masas en la vertical y no al eje de simetría del vehículo.

6.3.1 Base matemática del controlador óptimo LQR con efecto integral

A continuación, se muestra brevemente la base teórica de este tipo de controlador. Para más información, véase el trabajo ‘’*Low cost self balancing robot for control eduaction*’’, de Ignacio Alvarado Aldea, David Muñóz de la Peña y C. Gonzalez.

La idea de este controlador es la siguiente:

Suponiendo una desviación del centro de masas del cuerpo con respecto del eje de simetría, el controlador intenta colocar al vehículo de tal forma que las variaciones del ángulo y velocidad ángular sean cero. Por tanto, no existe una referencia del ángulo con la vertical, ya que es el propio controlador el que tiende a colocar al centro de masas en dicha vertical para así conseguir que las variaciones del estado entre un intervalo de

muestreo y el siguiente sean nulas. Veasé como, en la deducción matemática siguiente, el objetivo es que el estado no varie, y la única forma en que no varie es que el centro de masas se coloque en a vertical. Por otra parte, añadiendo el error en velocidad al estado, se consigue que el sistema siga referencias en velocidad, independientemente del ángulo con la vertical.

La deducción matemática se muestra a continuación.

Dado el modelo en tiempo discreto se tiene que:

$$x_{k+1} = Ax_k + Bu_k \tag{6–14}$$

$$x_k = Ax_{k-1} + Bu_{k-1} \tag{6-15}$$

$$\Delta x_{k+1} = x_{k+1} - x_k = A\Delta x_k + B\Delta u_k \tag{6-16}$$

Si definimos el error en velocidad como

$$e_k = Ref_k + [0 \quad 0 \quad 1]x_k \tag{6–17}$$

se tiene que para una referencia constante:

$$\Delta e_k = 0 - [0 \quad 0 \quad 1]\Delta x_k \tag{6–18}$$

$$e_k = e_{k-1} - [0 \quad 0 \quad 1]\Delta x_k \tag{6-19}$$

Nótese que la velocidad es la tercera componente del vector de estados.

Juntando (6-14) con (6-17) obtenemos:

$$\begin{bmatrix}\Delta x_{k+1}\\ e_k\end{bmatrix} = \begin{bmatrix} A & 0\\ -[0 \quad 0 \quad 1] & 1\end{bmatrix}\begin{bmatrix}\Delta x_k\\ e_{k-1}\end{bmatrix} + \begin{bmatrix}B\\ 0\end{bmatrix}\Delta u_k \tag{6–20}$$

Si aplicamos la ley de control

$$\Delta u_k = -K\begin{bmatrix}\Delta x_k\\ e_{k-1}\end{bmatrix} \tag{6–21}$$

y sustituyendo en (6-18) obtenemos:

$$\begin{bmatrix}\Delta x_{k+1}\\ e_k\end{bmatrix} = \left(\begin{bmatrix} A & 0\\ -[0 \quad 0 \quad 1] & 1\end{bmatrix} - \begin{bmatrix}B\\ 0\end{bmatrix}K\right)\begin{bmatrix}\Delta x_k\\ e_{k-1}\end{bmatrix} \tag{6–22}$$

El objetivo es

- Encontrar la ganacia K que haga que nuestro sistema sea estable
- Minimice el funcional J, con J igual a

$$J = \sum_{\forall k} \begin{bmatrix} \Delta x_k \\ e_{k-1} \end{bmatrix}^T Q \begin{bmatrix} \Delta x_k \\ e_{k-1} \end{bmatrix} + \Delta u_k{}^T R \Delta u_k \tag{6–23}$$

Una vez calculado K de la expresión (6-20), la acción de control en el periodo k vale, atendiendo a la expresión (6-19):

$$u_k = u_o - K \begin{bmatrix} x_k - x_o \\ \sum_{i=0}^{k-1} e_i \end{bmatrix} \tag{6–24}$$

6.3.2 Obtención de la ganancia de realimentación K con MATLAB

El siguiente programa en MATLAB cálcula, a partir de los parámetros del modelo, y para unas matrices semidefinidas positivas Q y R dadas, la ganancia de realimentación K.

Nota, escoger unos valores de Q y R que hagan que los ordenes de magnitud de las componentes del vector de realimentación K sean parecidos a los obtenidos en el diseño del LQR del apartado anterior

```
%Parametros del modelo

M=0.945;          %Masa del solido 2 en Kg
mr=0.070;         %Masa de la rueda en Kg
R=0.05;           %Radio de las ruedas en m
L=0.05;           %Distancia desde la base ejes solido 2 hasta el centro de masas del solido 2 en
m
g=9.81;           %Aceleración de la gravedad ms^-2
Iyy=0.00994356;   %Inercia del solido 2 respecto de los ejes 2 Kgm^2
Ir=0.00098421;    %Inercia de las ruedas respecto al eje del motor kgm^2

%Sistema en tiempo continuo
A=[0 1 0;(M*g*L)/(Iyy+M*R*L) 0 0;0 0 0];
B=[0;-(2*Ir+2*mr*R^2+M*R^2+M*R*L)/(Iyy+M*R*L);1];
C=[0 0 1];
D=0;

SYS=ss(A,B,C,D);

%Sistema en tiempo discreto

SYSD=c2d(SYS,0.02);
AA=[SYSD.A zeros(3,1);-SYSD.C 1];
BB=[SYSD.B;0];
CC=zeros(1,4);
DD=0;
Q=[10 0 0 0;0 1 0 0;0 0 50 0;0 0 0 50];
R=800;
```

```
K=dlqr(AA,BB,Q,R)
```

Figura 6-11. Código en Matlab para la obtención de la ganancia de realimentación K.

6.3.3 Resultados obtenidos

Los resultados obtenidos con esta estrategia de control son fantásticos. Los resultados de controladores anteriormente diseñados son bastante peores en comparación con esta estrategia de control. El porqué el controlador óptimo LQR con efecto integral ofrece tan buenos resultados se explicó en la anterior deducción matemática. Este controlador tiene en cuenta un posible desvio del centro de masas, mientras que los anteriores no.

Al considerar ese desvió en el centro de masas respecto al eje de simetría, el controlador tiende a estabilizar al vehículo de tal manera que el centro de masas quede en la vertical. A su vez, con el término integral, se consigue el seguimiento de referencias en velocidad.

Se verá que, con los dos programas diseñados en el capítulo 5, los resultados son asombrosos, no pudiéndose apreciar diferencia alguna para el rango de velocidades en el que nos movemos.

Véanse, a continuación, los diferentes resultados para multitud de situaciones.

6.3.3.1 Seguimiento de referencias en velocidad

En esta prueba, el controlador deberá de seguir diferentes referencias en velocidad. Se recuerda que, para el caso de movimiento circular, donde una rueda gira más que la otra, el control se realizará sobre la rueda que gira a menor velocidad.

A continuación, se muestran los resultados obtenidos. Veasé como es posible alcanzar la referencia de 2500º/s, que equivale a 7.85 Km/h (El radio son 5cm).

6.3.3.1.1 Para movimiento rectilíneo

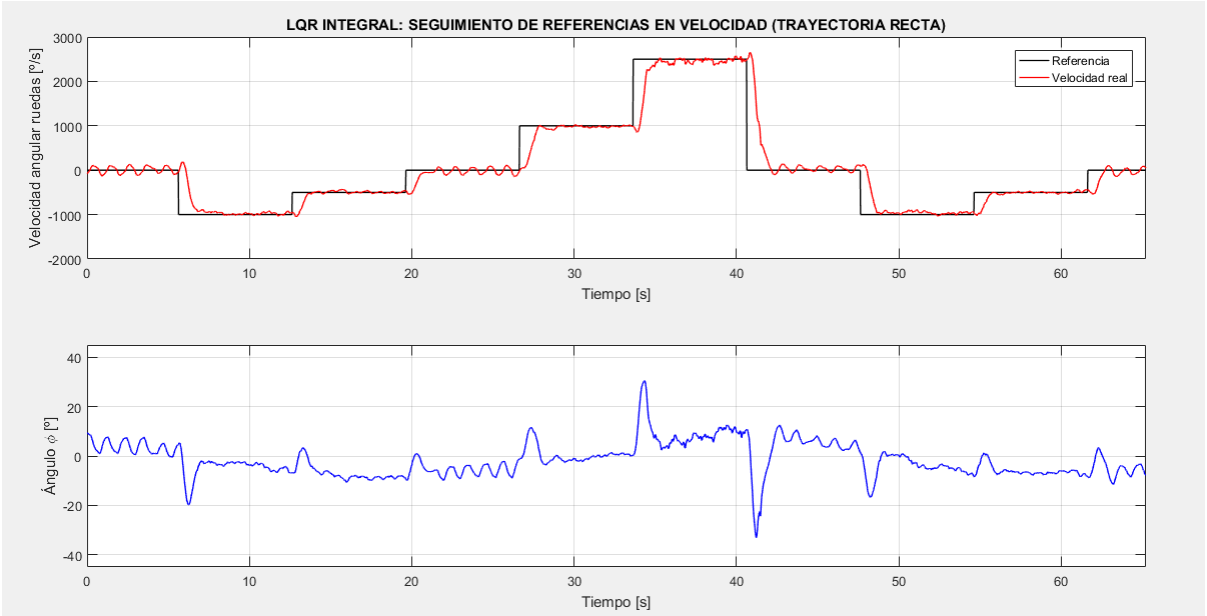


Figura 6-12. Seguimiento de referencias en velocidad en trayectoria rectilinea.

6.3.3.1.2 Para movimiento circular

Se puede observar como el disminuir el radio de circunferencia de la trayectoria que realiza el vehículo no perjudica apreciablemente el control. Por tanto, no se ve necesaria la utilización de un modelo en 3d del vehículo. El modelo dinámico obtenido para movimiento unidireccional es suficiente para los radios de curvatura de las taryectorias a las que someteremos a nuestro vehículo.

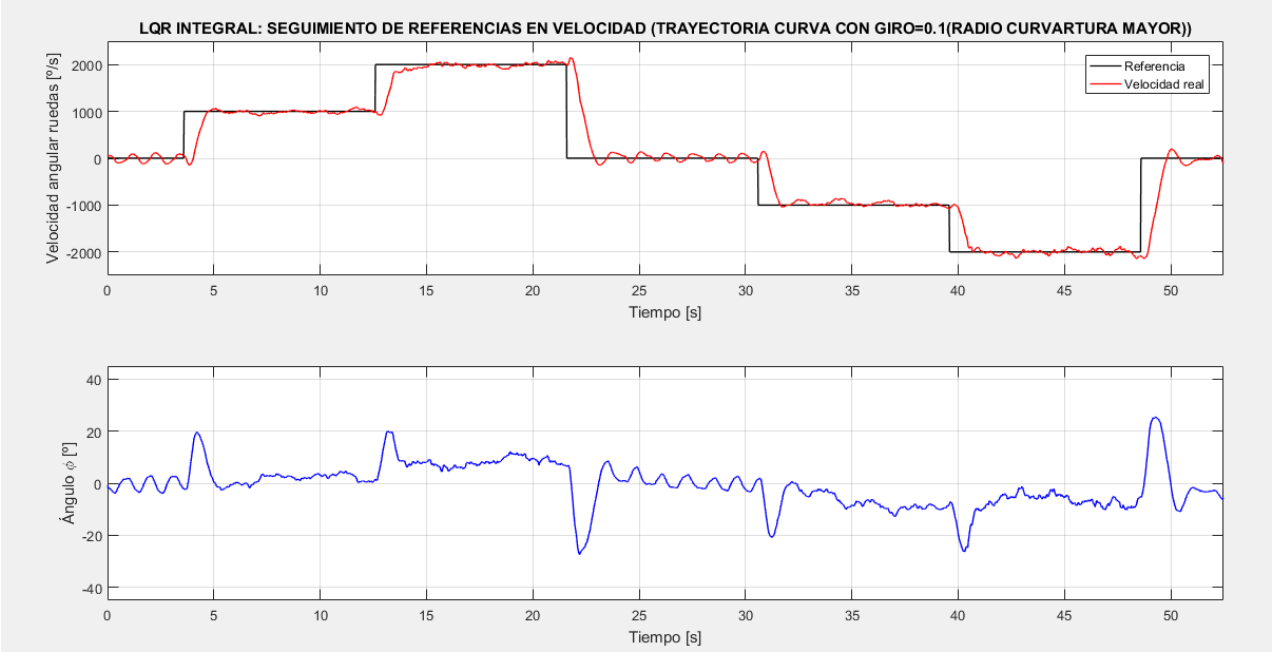


Figura 6-13. Seguimiento de referencias en velocidad en trayectoria circular con *giro*=0.1.

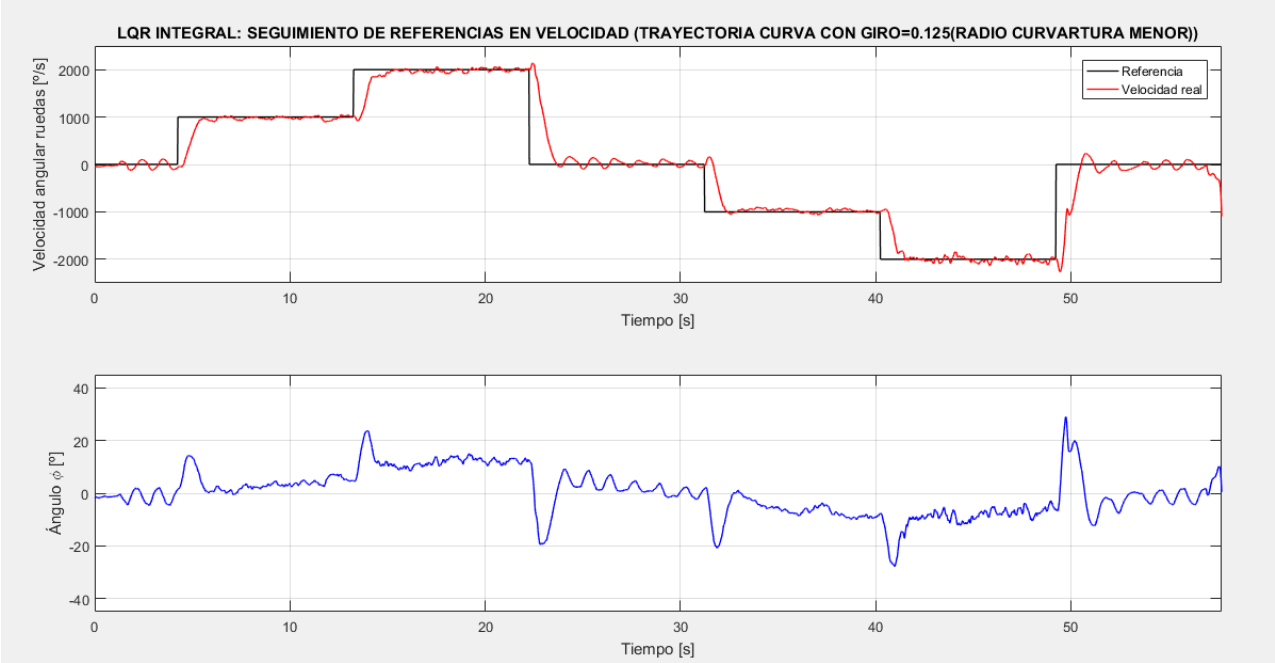


Figura 6-14. Seguimiento de referencias en velocidad en trayectoria circular con *giro*=0.125.

6.3.3.2 Seguimiento de referencias en velocidad con perturaciones tipo pulso

En esta prueba, el vehículo debe seguir referencias en velocidad mientras recibe perturbaciones tipo pulso (Golpes secos).

La capacidad para mantener el control del vehículo es óptima, existiendo situaciones en las que la inclinación ronda los 40º y, aun así, se mantiene el control.

6.3.3.2.1 Para movimiento rectilineo

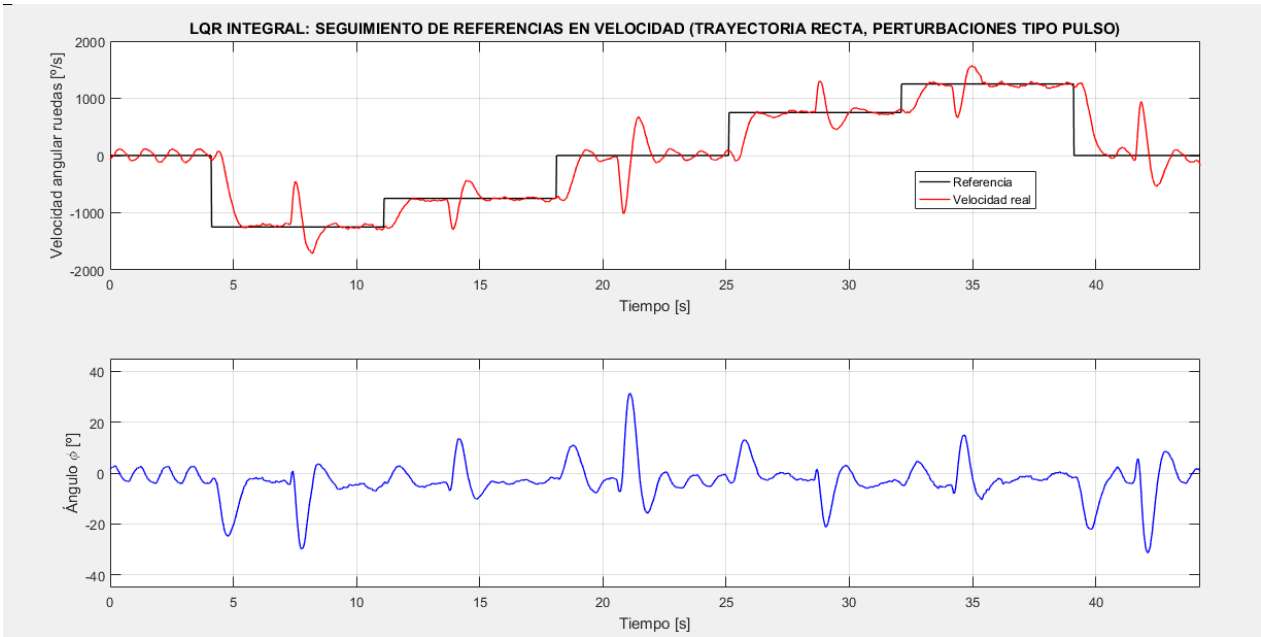


Figura 6-15. Seguimiento de referencias en velocidad con perturbaciones tipo pulso.

6.3.3.2.2 Para movimiento circular

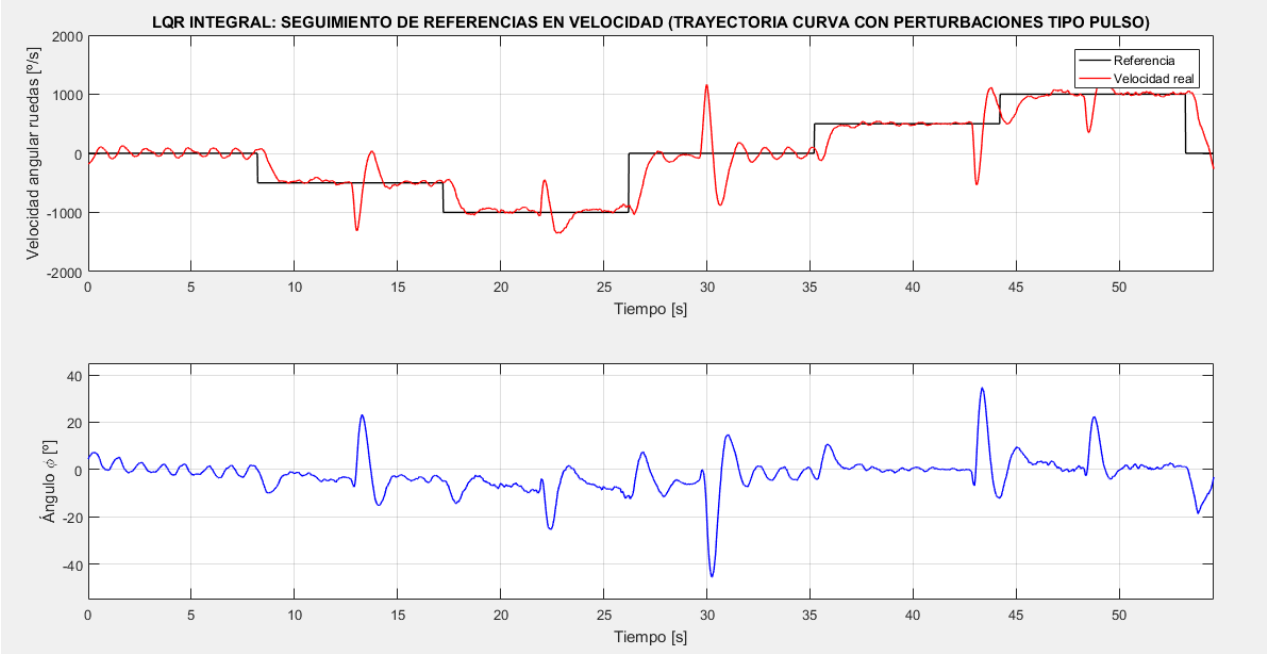


Figura 6-16. Seguimiento de referencias en velocidad con perturbaciones tipo pulso.

6.3.3.3 Seguimiento de referencia en velocidad nula con perturbación de tipo fuerza constante

En esta prueba, el vehículo intenta seguir una referencia nula en velocidad, pero una fuerza constante tiende a desplazarlo en una dirección.

Se puede observa como el péndulo se inclina para luchar contra dicha fuerza. En el momento en el que desaparece, el vehículo regresa a la posición donde comenzon la perturbación. Esto es así porque el error en velocidad se integra y, dado que la posición es la integral de la velocidad, el sistema tiende a la posición inicial. Recuérdese que con el controlador LQR no pasaba esto, ya que se quedaba en la posición donde lo habiá llevado la fuerza.

6.3.3.3.1 Para movimiento rectilineo

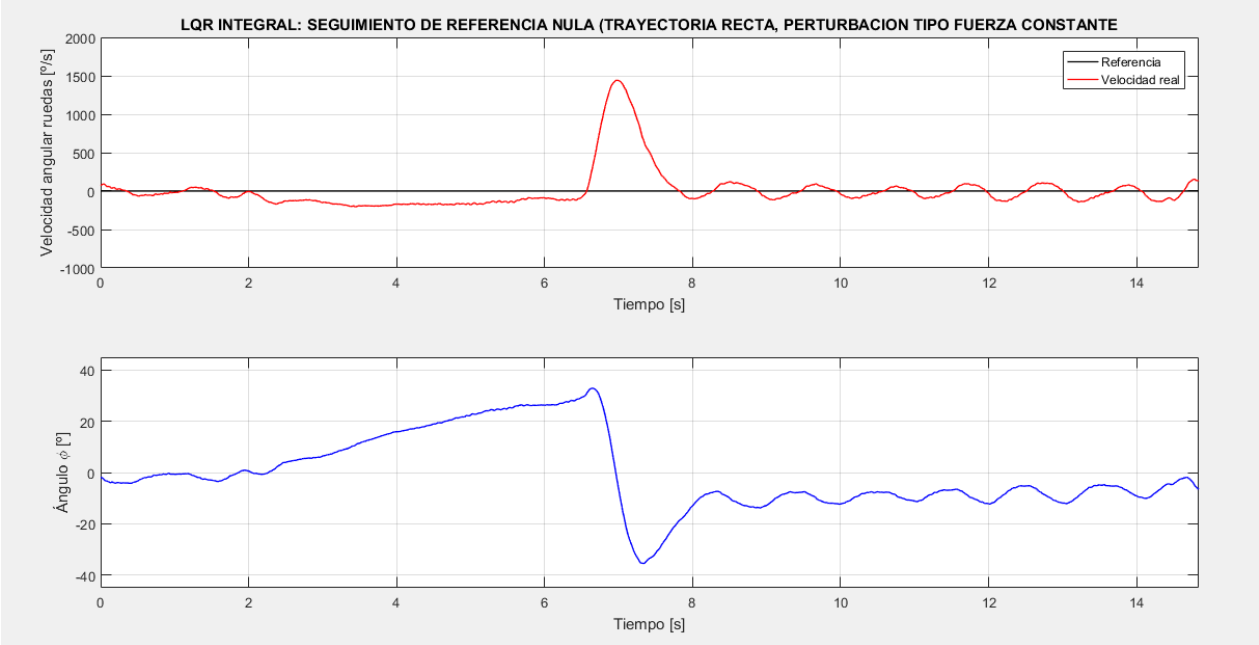


Figura 6-17. Seguimiento de referencia nula con perturbaciones tipo fuerza constante.

6.3.3.3.2 Para movimiento circular

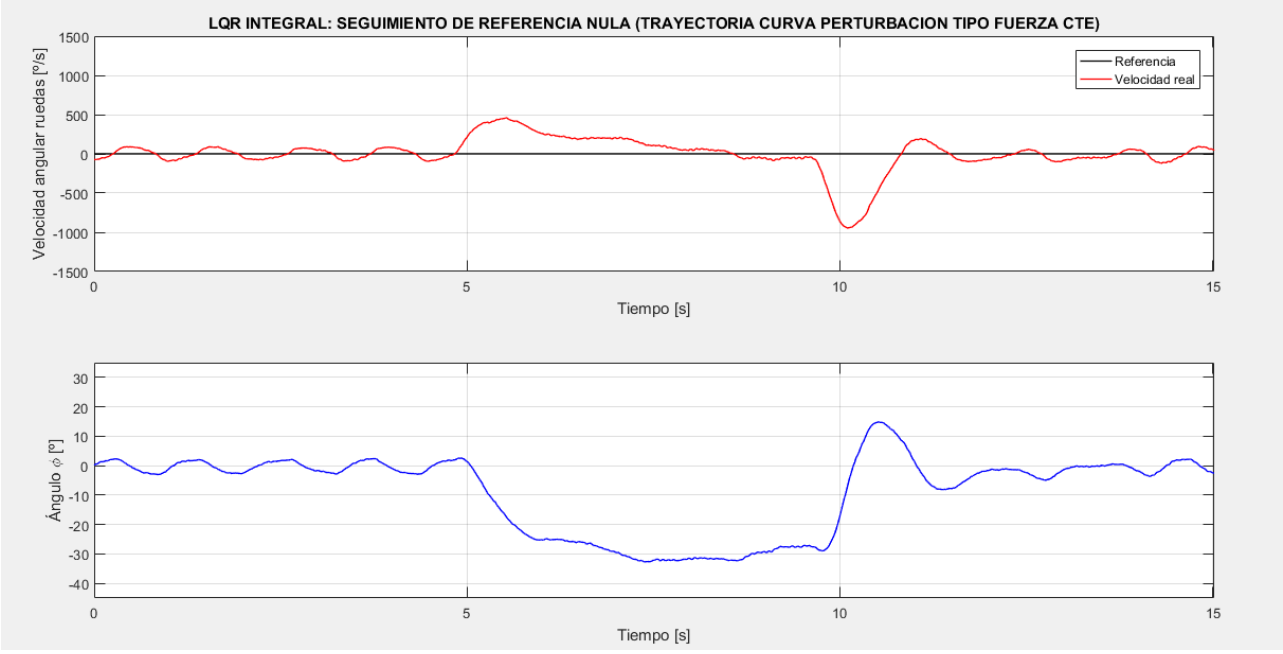


Figura 6-18. Seguimiento de referencia nula con perturbaciones tipo fuerza constante.

6.3.3.4 Seguimiento de referencias en velocidad en un plano inclinado

Solo se mostrará la gráfica para movimiento unidireccional.

El experimento consiste en el seguimiento de una velocidad de referencia bajando y subiendo una rampa.. Como comentario, se puede observar como el vehículo se inclina hacia la parte de arriba de la rampa (cerca de 20º) para anular la perturbación introducida por la componente del peso en la dirección del movimiento.

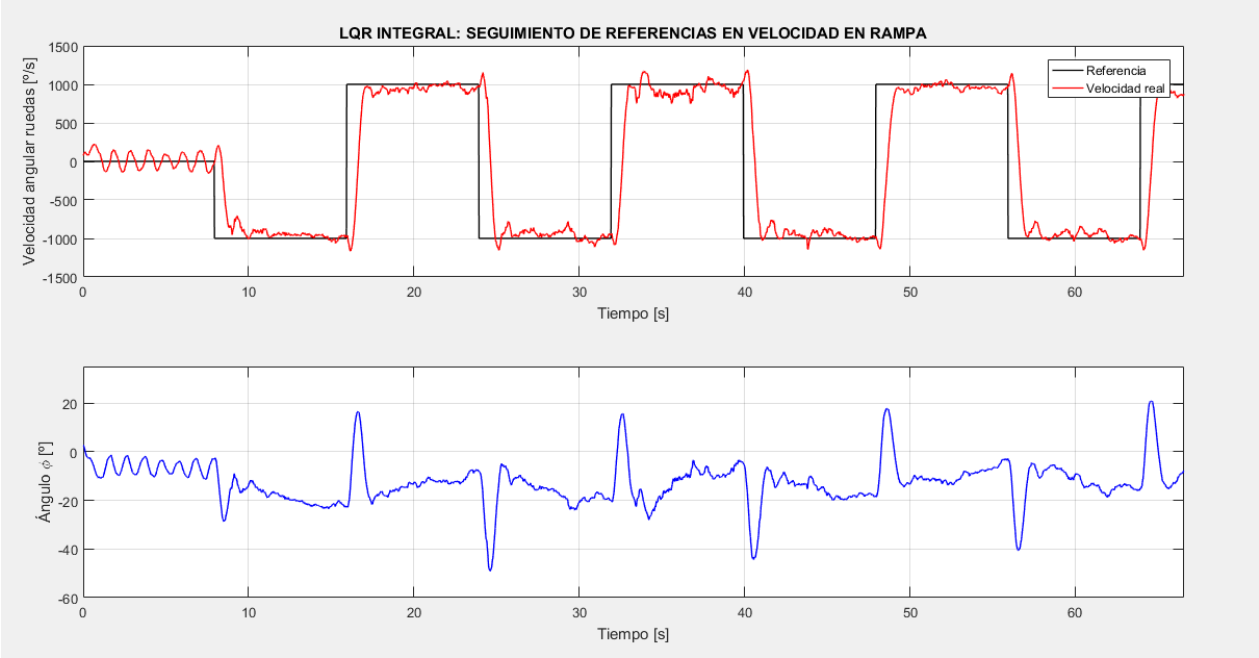


Figura 6-19. Seguimiento de referencia en velocidad bajando rampa.

6.3.3.5 Seguimiento de referencias con desviación del centro de masas

En este experimento se modifica la posición del centro de masas del vehículo mediante la colocación de diversos pesos. Una vez modificada la posición de G, se intenta seguir diversas referencias en velocidad.

Los resultados son los esperados:

- Dado que el añadir diversas masas modifica completamente la dinámica del sistema, el controlador LQR con efecto integral diseñado anteriormente no posee la suficiente ganancia como para ser capaz de controlar correctamente al sistema. Por ello, el seguimiento de referencias en velocidad oscila mucho en torno a la referencia.
- La inclinación en régimen permanente es de -35º aproximadamente. La explicación ya se dio en el diseño del controlador. El controlador intenta colocar al centro de masas en la vertical, y no el eje de simetría del vehículo. Por tanto, debe inclinarse de tal manera que G quede en la vertical.

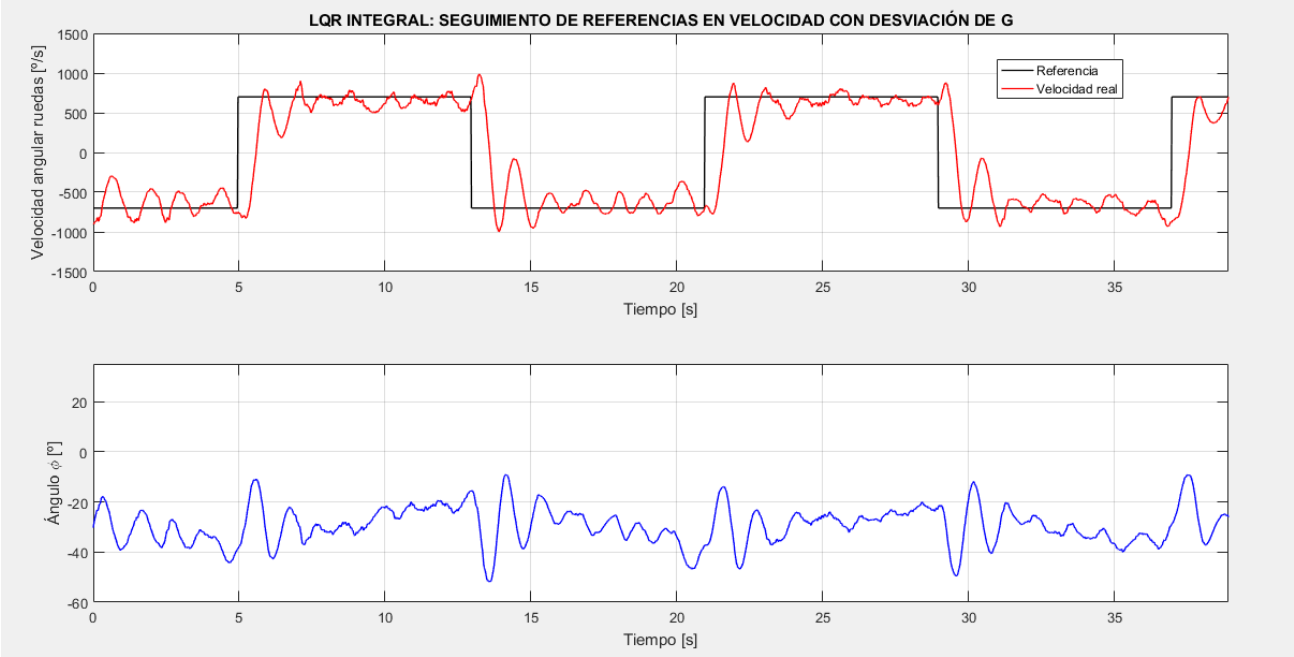


Figura 6-20. Seguimiento de referencia en velocidad con desviación de G.

Como se ha podido observar durante este capítulo, los resultados obtenidos añadiéndole un término integral al controlador LQR son brillantes. El error prácticamente nulo en el seguimiento de referencias, y la rapidez con la que se alcanzan las referencias, dejan prácticamente en 'ridículo' al resto de estrategias de control vistas en este documento. Además, gracias a que, ante una desviación del centro de masas, es capaz de autoequilibrarse manteniendo al nuevo centro de masas en la vertical, hacen que esta estrategia de control sea ideal para transporte de pequeñas masas sobre el vehículo.

Aquí concluye el actual capítulo *Control*.

7 TELEOPERACIÓN

Fija tu rumbo a una estrella y podrás navegar a través de cualquier tormenta.

Leonardo DaVinci

En este capítulo aprenderemos a configurar mediante comandos AT los módulos Bluetooch HC-05 y HC-06. El módulo HC-06 es un dispositivo esclavo que se colocará en la PCB del vehículo. El módulo HC-05 formará parte del equipo de teleoperación, junto con otro microcontrolador Arduino UNO R3 y el mando Numshuk de la consola Wii. Este dispositivo tiene la posibilidad de configurarse como maestro y de conectarse activamente con el dispositivo remoto, que en este caso será el HC-06 colocado en el vehículo.

Una vez montado el sistema de teleoperación, lo programaremos para dos modos diferentes:

- Movimiento unidireccional del vehículo: Le mandaremos referencias en velocidad proporcionales al desplazamiento del Joystick de mando Numshuk de la Wii.
- Movimiento bidireccional del vehículo: mandaremos referencias en velocidad y valores para la variable giro que se vio en el capítulo 5.

7.1 Configuración módulo HC-06

Realice las siguientes conexiones:

- Vcc Bluetooch con Vcc Arduino UNO.
- Gnd Bluetooch con Gnd Arduino UNO.
- Rx del Bluetooch con pin 11 (pin que simula Tx) Arduino UNO.
- Tx del Bluetooch con pin 10 (pin que simula Rx) Arduino UNO

Cambie el nombre, pin y velocidad de transmisión tocando las líneas correspondientes del código de la figura 7-1. Luego programe el Arduino con dicho programa y abra el monitor Serie para ver las contestaciones del módulo. Si vemos que no nos responde es porque la velocidad de transmisión con la que nos comunicamos con él es diferente de la que se programo por ultima vez. Si la desconoce, vaya cambiando la velocidad hasta obtener una respuesta correcta.

De hecho, la velocidad es lo último que se configura porque, una vez cambiada ya no coincide con la velocidad con la que nos estábamos comunicando antes. En este caso, pasaría de 9600 a 115200.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  Serial.begin(9600);//velocidad de comunicación con el ordenador
  while (!Serial) {
  }
  Serial.println("Configuracion Bluetooth segway");

  mySerial.begin(9600);//Velocidad de comunicación con el HC-06
                       //Su valor depende del último que la configurara
}

void loop() {


  mySerial.write("AT\r\n");//Comprobamos conexión correcta
  delay(1000);
    while(mySerial.available()>0) {//Esperamos OK
    Serial.write(mySerial.read());
  }

    mySerial.write("AT+NAMESEGWAY");//SEGWAY sería el nombre
  delay(1000);
    while(mySerial.available()>0) {//Esperamos OK
    Serial.write(mySerial.read());
  }

  mySerial.write("AT+PIN1996");//1996 sería la contraseña
  delay(1000);
    while(mySerial.available()>0) {//Esperamos OK
    Serial.write(mySerial.read());
  }

  mySerial.write("AT+BAUD8");// Mirar tabla de abajo
  delay(1000);
    while(mySerial.available()>0) {//Esperamos OK
    Serial.write(mySerial.read());
  }

  while(1);
}
/*
AT+BAUD1---------1200 Baudios
AT+BAUD2---------2400 Baudios
AT+BAUD3---------4800 Baudios
AT+BAUD4---------9600 Baudios
AT+BAUD5---------19200 Baudios
AT+BAUD6---------38400 Baudios
AT+BAUD7---------57600 Baudios
AT+BAUD8---------115200 Baudios
AT+BAUD9---------230400 Baudios
AT+BAUDA---------460800 Baudios
AT+BAUDB---------921600 Baudios
AT+BAUDC---------1382400 Baudios
*/
```

Figura 7-1. Configuración HC-06.

7.2 Configuración HC-05

Las conexiones son la mismas que con el HC-06:

- Vcc Bluetooch con Vcc Arduino UNO.
- Gnd Bluetooch con Gnd Arduino UNO.
- Rx del Bluetooch con pin 11 (pin que simula Tx) Arduino UNO.
- Tx del Bluetooch con pin 10 (pin que simula Rx) Arduino UNO.

El problema de desconocer el valor de la velocidad de transmisión del módulo HC-06 desaparece con el HC-05 ya que, si previamente a alimentarlo mantenemos pulsado el botón que tiene y, manteniéndolo pulsado se le da alimentación, la velocidad para configurarlo es de 38400 Baudios.

A parte de configurar nombre, contraseña y velocidad de transmisión, también se configura al dispositivo en modo maestro y en modo 0, lo que significa que nada más encenderse se conectará activamente a la dirección fijada con el comando AT+BIND.

Para conocer la dirección Bluetooch del dispositivo remoto (HC-06), se recomienda conectarse con el móvil a dicho dispositivo y, una vez conectado, mirar su dirección dentro de los ajustes. Una vez conocida, escribirla en el código de la forma en que aparece en la figura 7-2 (XXXX, XX, XXXXXX).

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
void setup() {
  Serial.begin(9800);
  while (!Serial) {
    ;
  }
  Serial.println("Configuracion Bluetooth NUMSHUK");
  mySerial.begin(38400);
}
void loop() {
    mySerial.write("AT\r\n");//Vemos que la conexión es correcta
    delay(1000);
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }
    mySerial.write("AT+NAME=NUMSHUK\r\n");//Nombre del dispositivo
    delay(1000);
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }
    mySerial.write("AT+UART=115200,0,0\r\n");//Velocidad de transmisión
    delay(1000);
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }
    mySerial.write("AT+PSWD=1996\r\n");//contraseña
    delay(1000);
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }
    mySerial.write("AT+CMODE=0\r\n");//Para que se conecte activamente
    delay(1000);                     //a la dirección especificada
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }

    mySerial.write("AT+ROLE=1\r\n");//Configuración como maestro
    delay(1000);
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }
    mySerial.write("AT+BIND=0014,03,067DAE\r\n");//Dirección bluetooch
    delay(1000);                                 //dispositivo esclavo:
SEGWAY
    while(mySerial.available()>0) {
      Serial.write(mySerial.read());
    }
  while(1);
}
```

Figura 7-2. Configuración HC-05.

7.3 Conexión mando Numshuk

En el blog de Luis Llamas (https://www.luisllamas.es/) se nos explica todo acerca de este mando, por lo que aquí se explicará de la forma mas breve posible.

Siguiendo su trabajo, el diagrama de conexiones es el siguiente:

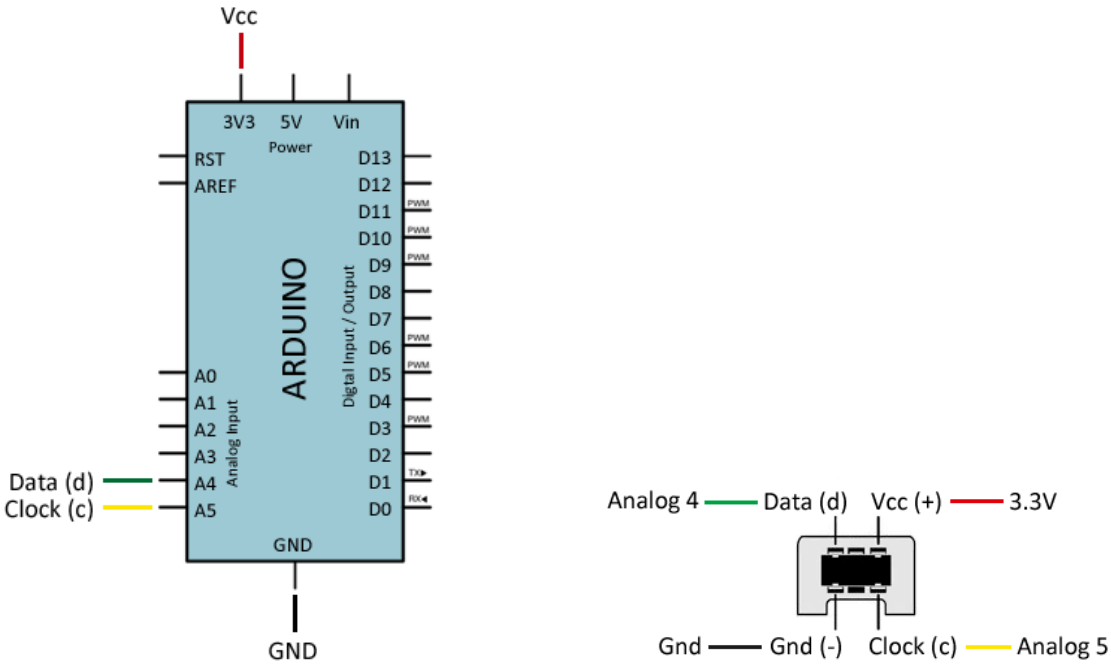


Figura 7-3. Diagrama de conexiones.

Nota importante: Los colores de los cables del mando pueden no ser los mismos que en la anterior imagen por lo que se deben identificar por su posición:

- Arriba izquierda Analog 4.
- Arriba derecha Vcc 3.3v.
- Abajo izquierda Gnd.
- Debajo derecha Analog 5.

7.4 Teleoperación para movimiento unidireccional

La idea del programa del teleoperador es sencilla:

- Se cálcula el ángulo de la posición del joystick. Y:
  - Si apunta hacia delante (Ángulo menor de 30º y mayor de -30º): la velocidad será positiva.
  - Si apunta hacia detrás (Ángulo entre 150º y 210º): la velocidad será negativa.
  - Si no apunta en ninguna de las direcciones anteriores: la velocidad es nula.
- Si estamos en alguna de las dos primeras zonas, se aplicará una velocidad proporcional al desplazamiento del Joystick. El desplazamiento del Joystick será una cantidad comprendida entre 0 y 127 (Valor entero máximo para un *Signed Char de tamaño* un Byte)
- Se enviará el desplazamiento con signo (Valores posibles entre -127 y 127) al vehículo y este, mediante una regla de tres, pasará dicha cantidad a una velocidad. Por ejemplo, 127 se podría convertir en 1500º/s y -127 en -1500º/s.

Al programa del vehículo que vimos en el capítulo 5 habrá que añadirle una función dentro del ejecutivo cíclico que lea el puerto serie y convierta el *signed char* que le llega a un *float* mediante una regla de tres.

Se usa un *Signed Char* por su poco tamaño (8 bits), de manera que el tiempo de coste de leer el puerto serie sea despreciable y siga funcionando correctamente el ejecutivo cíclico.

A continuación, se muestran los códigos.

Figura 7-4. Cambio en el ejecutivo cíclico del vehículo.

```
#include <Wiichuck.h>
#include <Wire.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
Wiichuck wii;

void setup() {
  Serial.begin(9600);
  wii.init();
  wii.calibrate();
  while (!Serial) {
    ;
  }

  Serial.println("Iniciando programa...");

  mySerial.begin(115200);
}

void loop() {

  static float alfa=0;
  static signed char r;

  if(wii.poll()){
    alfa=(360/(2*3.1415))*atan2((wii.joyX()-125),(wii.joyY()-125));
    if(alfa<0){
      alfa+=360;
    }
    r=sqrt(pow(wii.joyX()-125,2)+pow(wii.joyY()-128,2));
    r=constrain(r,0,90);
    if(alfa<30||alfa>330){
      mySerial.write(r);
    }
    else{
      if(alfa>150&&alfa<210){
        r*=(-1);
        mySerial.write(r);
      }
    }
  }
}
```

Figura 7-5. Código del Teleoperador.

7.5 Teleoperación para movimiento bidireccional

De manera resumida, lo que se hace en este punto es el diseño de un programa que mande los valores de la referencia en velocidad y de la variable *giro* en función de la posición del joystick.

Dichos datos se mandarán, como se comentó en el apartado anterior, mediante un *Signed Char*, de tal manera que solo se envie un byte por cada parámetro.

Concretando en el funcionamiento, el módulo de la referencia en velocidad es proporcional a la distancia del joystick al centro. Para el signo de dicha velocidad, calculamos el coseno del ángulo alfa:

- Si dicho coseno es mayor que 0º (alfa mayor de -90º y menor de 90º), el signo será positivo. Por tanto, el segway se moverá hacia delante.
- Si dicho coseno es negativo (alfa mayor de 90º y menor de 270º), el signo será negativo y, por tanto, el segway se moverá hacia detrás.

Como se comentó en el apartado anterior, cuando el vehículo reciba esta referencia en velocidad mediante un solo byte (valores entre -127 y 127), este realizará una regla de tres para obtener la referencia en velocidad en un determinado rango.

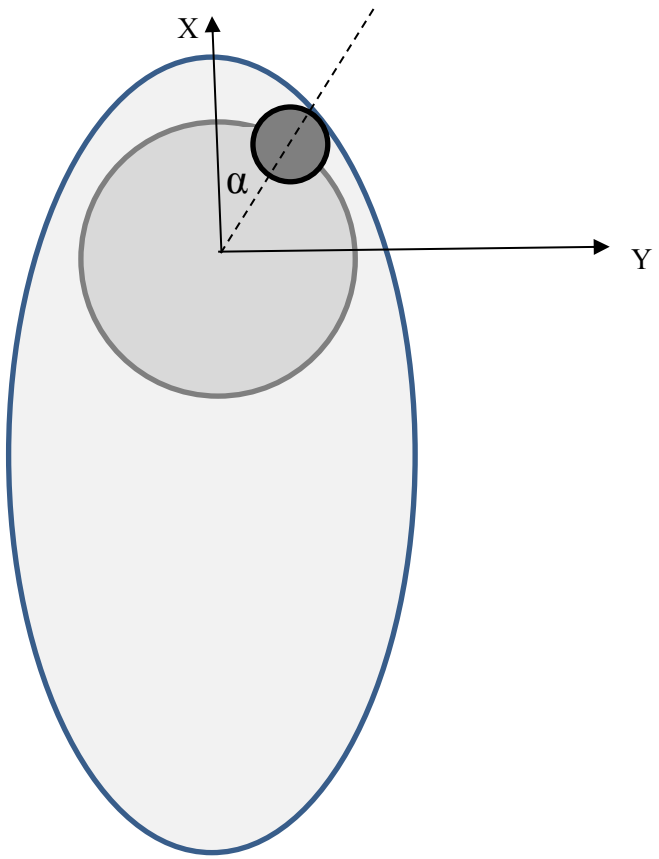


Figura 7-6. Mando Numshuk: orientación ejes del Joystick.

Con respecto al envío del parámetro *giro*, este irá dado por el seno del ángulo alfa de manera que:

- Si el joystick se encuentra en la mitad derecha, la variable *giro* será positiva y comprendida entre 0º y 180º. Y mientras mas cercano este el joystick del eje Y, mas grande en módulo será dicho valor.
- Si el joystick se encuentra en la mitad izquierda, la variable *giro* será negativa y comprendida entre 180º y 360º. Y mientras mas cercano este el joystick del eje Y, mas grande en módulo será dicho valor.

En la practica, este valor de *giro* comprendido entre -1 y 1 resulta problemático ya que, ante un muy muy leve movimiento del joystick, hace que la diferencia de velocidad entre las ruedas cambie muchísimo, haciendo que sea muy difícil manejarlo. Para solucionar esto, se multiplica el valor de *giro* por un factor de 0.4, de tal manera que el manejo se vuelve más fácil.

Dado que queremos enviar el valor de *giro* mediante un byte, se hace necesario mutiplicar por un factor 100 su valor (127 tambien valdría) y aproximar al entero más próximo. Posteriormente, en el programa del vehículo, se deshacerá el cambio para obtener el valor real.

```
//GND - GND
//VCC - VCC
//SDA - Pin A4
//SCL - Pin A5

#include <Wiichuck.h>
#include <Wire.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
Wiichuck wii;

void setup() {
  Serial.begin(9600);
  wii.init();
  wii.calibrate();
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Iniciando programa...");

  // set the data rate for the SoftwareSerial port
  mySerial.begin(115200);
  delay(10000);
}
void loop() {

  static float alfa=0;
  static float refv=0,giro=0;
  static signed char gir=0,ref=0;
  if(wii.poll()){
    alfa=atan2((wii.joyX()-125),(wii.joyY()-125));

    refv=sqrt(pow(wii.joyX()-125,2)+pow(wii.joyY()-128,2));
    refv=constrain(refv,0,90);
    if(cos(alfa)<0){
      refv*=(-1);

    }
    giro=0.4*sin(alfa);

    ref=(signed char)refv;
    gir=(signed char)100*giro;

    mySerial.write(ref);
    mySerial.write(gir);
  }
}
```

Figura 7-7. Código del Teleoperador.

A continuación, se muestra la función dentro del vehículo, la cual se llama en el tercer subtramo del ejecutivo cíclico.

Dado que el teleoperador envia los datos de la referencia en velocidad y *giro* de manera consecutiva, se puede observar como, con la variable *aux*, se guardan alternativamente los datos en la variables *ref_vel* y *giro*. De esta manera evitamor el error de que se guarde el valor de velocidad como si fuera el valor de giro y viceversa.

Por último, para garantizar de que el primer valor que se lea sea de velocidad y no de giro, se realiza una espera de 10s dentro del void setup de la figura 7-7. De esta manera garantizamos que el vehículo está ya preparado para leer comandos y que, por tanto, el primero que llegue será leído como referencia en velocidad.

```
void lectura_teleoperador(void){
  if(Serial.available()>0){
    if(aux2==1){
      ref_vel=Serial.read();
      refv=(float)ref_vel*10;
      aux2=0;
    }
    else{
      gir=Serial.read();
      giro=(float)gir*0.01;
      aux2=1;
    }
  }

}
```

Figura 7-8. Recepción de datos en el programa del vehículo.

Aquí concluye el capítulo *Teleoperación*.

8 CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

Nunca andes por el camino trazado, pues te conducirá a donde otros ya fueron.

Alexander Graham Bell

Tal como se puede apreciar en este documento, se ha llevado a cabo el diseño, fabricación, estudio dinámico, estimación de parámetros, programación de microcontrolador, implementación de diferentes técnicas de control y la teleoperacíon del mismo. Este trabajo se ha desarrollado tanto como Trabajo de Fin de Grado como de proyecto de la Beca de Colaboración de Ministerio, dentro del departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla.

Una vez terminada mi estancia en el departamento, y desde mi perspectiva de autor, existen innumerables proyectos a realizar a partir del mismo.

- Solucionar el problema de la aplicación de velocidades lentas que se vió en el capítulo 5.
- Cambios en el hardware para pasar de un microcontrolador Arduino Uno a un microcontrolador Arduino Mega, el cuál posee 3 Timer de 16 bits de resolución, por lo que desaparecería el problema del preescalador único visto en el capítulo 5.
- Cambios en la PCB para incorporar otros dispositivos como un sensor de ultrasonidos, cuyo uso podría emplearse en la evitación de obstáculos.
- Cambios en el hardware para pasar de usar un microcontrolador Arduino UNO a un sistema multiprocesador como la Raspberry Pi 3 con un sistema operativo en tiempo real como el *Emlid RTlinux*. Personalmente, este sería el cambio que haría. Los cuatro núcleos de la Raspberry Pi trabajando a una frecuencia de 1.2GHz (Arduino trabaja a 16MHz) dotarián al vehículo de una potencia increíble, perimitiendo la realización de proyectos de mayor envergadura como:
    - Desarrollo de estimadores de posición del vehículo mediante el uso de técnicas como la odometría, visión mediante cámaras, balizas, etc.
    - Desarrollo de técnicas de comunicación entre varios de estos vehículos de manera que lleven a cabo una misión encomendada como, por ejemplo, la formación de figuras sobre el suelo.
    - Desarrollar técnicas de SLAM (Simultaneous Localization And Mapping) para la elaboración de mapas del entorno.
    - Desarrollar técnicas de seguimiento de líneas sobre el suelo.
    - Teleoperación via Sockets con visión en streaming de la situación del vehículo.

Personalmente, me gustaría desarrollar nuevas estrategias de control como controladores predictivos lineales, no lineales, etc. Me gustaría desarrollar un estudio para conseguir que el vehículo realice un ‘’Looping’’manteniendo el equilibrio. Otro proyecto fascinante sería, mediante controladores no lineales basados en la inyección de energía, conseguir el llamado efecto *swing_up*, consiguiendo que, mediante balanceo, el péndulo pasara de su posición de equilibrio estable a su posición de equilibrio inestable.

Figura 8-1. Looping.

Aquí concluye el capítulo *Conclusiones y líneas futuras de investigación.*

ANEXO I: DEDUCCIÓN TCM Y TMC

Estos teoremas no se deducen, sino que son leyes físicas que se cumplen. Lo que se va a hacer es mostrar cómo de la forma discreta para un sistema de partículas se pasará a la forma continua particularizada para un sólido rígido. Los razonamientos matemáticos que se presentarán han sido obtenidos de los apuntes de Amplicaión de Física de 2º de Ingeniería de las Tecnologías Industriales, impartida por los profesores Dº. Manuel Toscano y Dº. Enrique Drake Moyano, a quién admiro profundamente. A continuación, se muestran los ‘’pantallazos’’ de los apuntes para ahorrarnos el tiempo en escribirlos de nuevo.

3.1. Teorema de la Cantidad de Movimiento (T.C.M.)

Hay que demostrar que: $\boxed{\dot{\vec{C}} = M\vec{a}_G = \vec{F}^{ext}}$, donde $\vec{F}^{ext}$ es la resultante de las fuerzas externas a S.

$$\vec{C} = \sum_i m_i\vec{v}_i = \imath\ \vec{v}_G = \frac{1}{M}\sum_i m_i\vec{v}_i \longrightarrow M\vec{v}_G = \sum_i m_i\vec{v}_i \imath = M\vec{v}_G \dashrightarrow \dot{\vec{C}} = M\dot{\vec{v}}_G = M\vec{a}_G\,.$$

Queda demostrar que $\dot{\vec{C}} = \vec{F}^{ext}$:

$$\vec{C} = \sum_i m_i\vec{v}_i \dashrightarrow \dot{\vec{C}} = \sum_i m_i\dot{\vec{v}}_i = \sum_i m_i\vec{a}_i = \imath\ 2^a\ ley\ Nw.\ \imath = \sum_i \vec{F}_i = \vec{F} = \vec{F}^{ext} + \vec{F}^{int}\,.$$

Falta demostrar finalmente que $\vec{F}^{int} = \vec{0}$:

$$\vec{F}^{int} = \sum_{parej} \left(\vec{F}_{ij} + \vec{F}_{ji}\right) = \imath\ 3^{er}\ Ppio.\ : \vec{F}_{ij} = -\vec{F}_{ji}\ \imath = \sum_{parej} \vec{0} = \vec{0}\ \ C.Q.D.$$

Este teorema también es conocido como *Teorema del Centro de Masas* por el protagonismo que G juega en él.

Del teorema se concluye que el movimiento de G depende únicamente de las fuerzas externas. Cuando no hay fuerzas externas (sistema totalmente aislado) G no se mueve o lo hace con movimiento rectilíneo y uniforme. La fuerzas internas son incapaces de influir en el movimiento del centro de masas: *"Nadie puede levantarse tirando únicamente de los pelos"*.

La reducción en A de las las fuerzas externas viene dada por $\left\{\vec{F}^{ext}, \vec{M}_A^{ext}\right\}$. Si no hay fuerzas externas $\left\{\vec{F}^{ext} = \vec{0}, \vec{M}_A^{ext} = \vec{0}\right\}$, el sistema se encuentra totalmente aislado. Se dice que un sistema está parcialmente aislado si al expresar en una base fija los vectores $\left\{\vec{F}^{ext}, \vec{M}_A^{ext}\right\}$ alguna de sus componentes es nula. Veremos que esto va a estar asociado con la existencia de magnitudes que se conservan.

3.2. Teorema del Momento Cinético (T.M.C.)

Hay que demostrar que:

$$\boxed{\dot{\vec{L}}_A = \vec{M}_A^{ext} + \vec{C} \wedge \dot{\overrightarrow{O_1A}}}\,.$$

donde $\vec{M}_A^{ext}$ es el momento resultante respecto a A de las fuerzas externas a S.

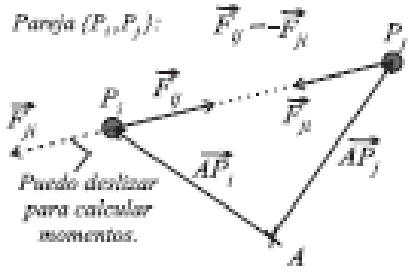
Partiendo del momento cinético:

$$\vec{L}_A = \sum_i \overrightarrow{AP}_i \wedge \vec{p}_i = \sum_i \overbrace{(\overrightarrow{O_1P_i} - \overrightarrow{O_1A})}^{\overrightarrow{AP_i}} \wedge m_i\vec{v}_i$$

Derivando:

$$\dot{\vec{L}}_A = \sum_i \dot{\overrightarrow{O_1P}}_i \wedge m_i\vec{v}_i - \sum_i \dot{\overrightarrow{O_1A}} \wedge m_i\vec{v}_i + \sum_i \overrightarrow{AP}_i \wedge m_i\dot{\vec{v}}_i = \imath\ \dot{\overrightarrow{O_1A}}\ sale\ fuera\ ;\ 2^a\ Ley\ m_i\dot{\vec{v}}_i = \vec{F}_i\ \imath =$$

$$\sum_i \underbrace{\vec{v}_i \wedge m_i\vec{v}_i}_{=\vec{0}\ ;\ paral.} - \dot{\overrightarrow{O_1A}} \wedge \underbrace{\sum_i m_i\vec{v}_i}_{=\vec{C}} + \underbrace{\sum_i \overrightarrow{AP}_i \wedge \vec{F}_i}_{=\vec{M}_A} = -\dot{\overrightarrow{O_1A}} \wedge \vec{C} + \vec{M}_A = \imath\ permuto\ en\ \wedge\ \imath = \vec{C} \wedge \dot{\overrightarrow{O_1A}} + \vec{M}_A\,.$$



Queda demostrar que $\vec{M}_A = \vec{M}_A^{ext}$.

Como $\vec{M}_A = \vec{M}_A^{ext} + \vec{M}_A^{int}$, faltaría demostrar que $\vec{M}_A^{int} = \vec{0}$:

$$\vec{M}_A^{int} = \sum_{parej} \left(\overrightarrow{AP_i} \wedge \vec{F}_{ij} + \overrightarrow{AP_j} \wedge \vec{F}_{ji} \right) = \text{deslizo } \vec{F}_{ji} \text{ hasta } P_i$$

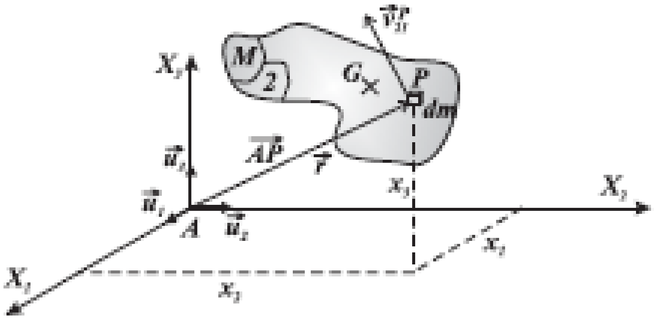
$$= \sum_{parej} \overrightarrow{AP_i} \wedge \left(\vec{F}_{ij} + \vec{F}_{ji} \right) = 3^{er}\ Ppio. : \vec{F}_{ij} = -\vec{F}_{ji} \ = \sum_{parej} \vec{0} = \vec{0} \quad C.Q.D.$$

- **Casos particulares:**
  - $A = G$: $\vec{C} \wedge \dot{\overrightarrow{O_1 A}} = \vec{C} \wedge \dot{\overrightarrow{O_1 G}} = \underbrace{\vec{C} \wedge \vec{v}_G}_{=\vec{0}\ ;\ paral.} = \vec{0} \Longrightarrow \boxed{\dot{\vec{L}}_G = \vec{M}_G^{ext}}$
  - $A = O$ (punto fijo): $\vec{C} \wedge \dot{\overrightarrow{O_1 A}} = \vec{C} \wedge \underbrace{\dot{\overrightarrow{O_1 O}}}_{=\vec{0}} = \vec{0} \Longrightarrow \boxed{\dot{\vec{L}}_O = \vec{M}_O^{ext}}$

1. **Momento cinético de un sólido rígido.**

- Dado un sólido rígido "2" cuyo estado cinemático $\left\{\vec{v}_{21}^{A}\,,\,\vec{\omega}_{21}\right\}$ y distribución de masas $\left\{\overleftrightarrow{I}_{A}\,,\,G\,,\,M\right\}$ son conocidos, vamos a demostrar que el momento cinético en A es igual a:

$$\boxed{\vec{L}_A = \overleftrightarrow{I}_A\,\vec{\omega}_{21} + M\overrightarrow{AG}\wedge\vec{v}_{21}^{A}}\,.$$



- Esta demostración se basa en tres pilares:
  - La definición del momento cinético de un sistema:

$$\vec{L}_A = \sum_{i=1}^{N}\overrightarrow{AP_i}\wedge m_i\vec{v}_i = \wr\ \textit{stma. continuo}\ \wr = \int_M \overrightarrow{AP}\wedge\vec{v}_{21}^{P}\,dm\,.$$

  - El campo de velocidades de un sólido rígido y la definición de la matriz de inercia:

$$\vec{v}_{21}^{P} = \vec{v}_{21}^{A} + \vec{\omega}_{21}\wedge\overrightarrow{AP} \quad , \quad \overleftrightarrow{I}_A = \overleftrightarrow{U}\int_M r^2\,dm - \int_M \vec{r}\,\vec{r}^{\,T}\,dm \quad (\vec{r} = \overrightarrow{AP})\,.$$

En la demostración usaremos provisionalmente la notación:

$$\overrightarrow{AP} = \vec{r} \quad , \quad \vec{v}_{21}^{P} = \vec{v}_P \quad , \quad \vec{v}_{21}^{A} = \vec{v}_A \quad , \quad \vec{\omega}_{21} = \vec{\omega}\,.$$

Desarrollando $\vec{L}_A$:

$$\vec{L}_A = \int_M \vec{r}\wedge\vec{v}_P\,dm = \int_M \vec{r}\wedge(\vec{v}_A + \vec{\omega}\wedge\vec{r})\,dm = \wr\ \vec{v}_A\ \ \textit{sale de}\ \int_M$$

$$;\ \textit{Doble}\wedge : \vec{A}\wedge(\vec{B}\wedge\vec{C}) = \vec{B}(\vec{A}\cdot\vec{C}) - \vec{C}(\vec{A}\cdot\vec{B})\ \wr = \Big[\underbrace{\int_M \vec{r}\,dm}_{=M\vec{r}_G}\Big]\wedge\vec{v}_A + \int_M\left[\vec{\omega}r^2 - \vec{r}\,(\vec{r}\cdot\vec{\omega})\right]dm\,.$$

Como $\vec{r}_G = \overrightarrow{AG}$ y $\vec{v}_A = \vec{v}_{21}^{A}$, falta demostrar que:

$$\int_M\left[\vec{\omega}r^2 - \vec{r}\,(\vec{r}\cdot\vec{\omega})\right]dm = \overleftrightarrow{I}_A\,\vec{\omega}\,.$$

Veamos:

$$\overleftrightarrow{I}_A\,\vec{\omega} = \underbrace{\overleftrightarrow{U}\,\vec{\omega}}_{=\vec{\omega}}\int_M r^2\,dm - \int_M \vec{r}\,\underbrace{\vec{r}^{\,T}\vec{\omega}}_{=\vec{r}\cdot\vec{\omega}}\,dm = \int_M\left[\vec{\omega}r^2 - \vec{r}\,(\vec{r}\cdot\vec{\omega})\right]dm \quad C.Q.D.$$

- Casos particulares:
  - $\boxed{A=G}$: $\overrightarrow{AG} = \vec{0} \longrightarrow \boxed{\vec{L}_G = \overleftrightarrow{I}_G\,\vec{\omega}\,.}$
  - Sólido con $\boxed{\text{punto fijo}}$ O en $\{21\}$: $\vec{v}_{21}^{A} = \vec{v}_{21}^{O} = \vec{0} \longrightarrow \boxed{\vec{L}_O = \overleftrightarrow{I}_O\,\vec{\omega}\,.}$

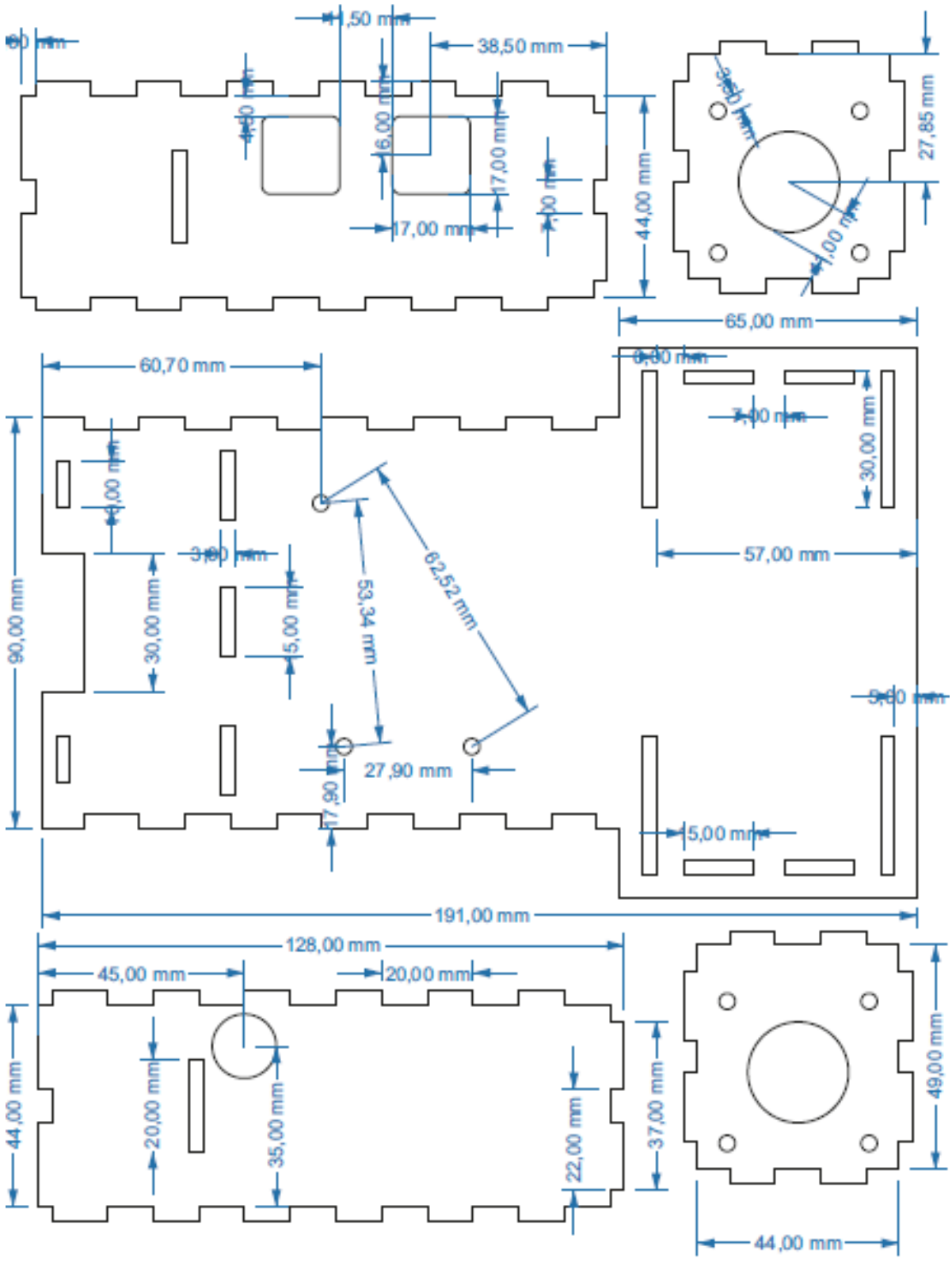
ANEXO II: USO DE LA CORTADORA LÁSER

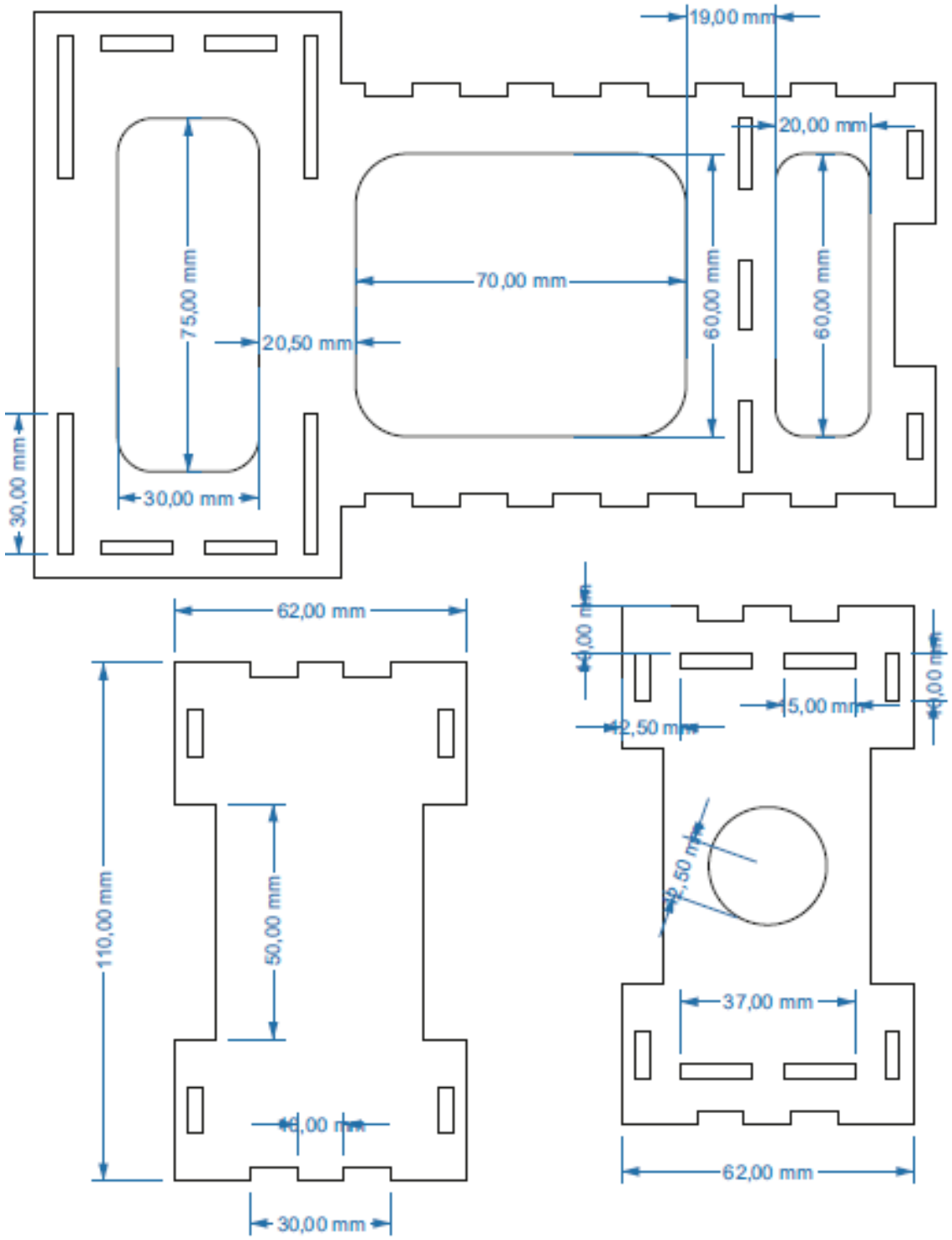
En este anexo se presenta un manual con la serie de pasos a seguir para el uso de la cortadora láser del departamento a partir de un diseño en 2D en formato .cdr en versión 12.

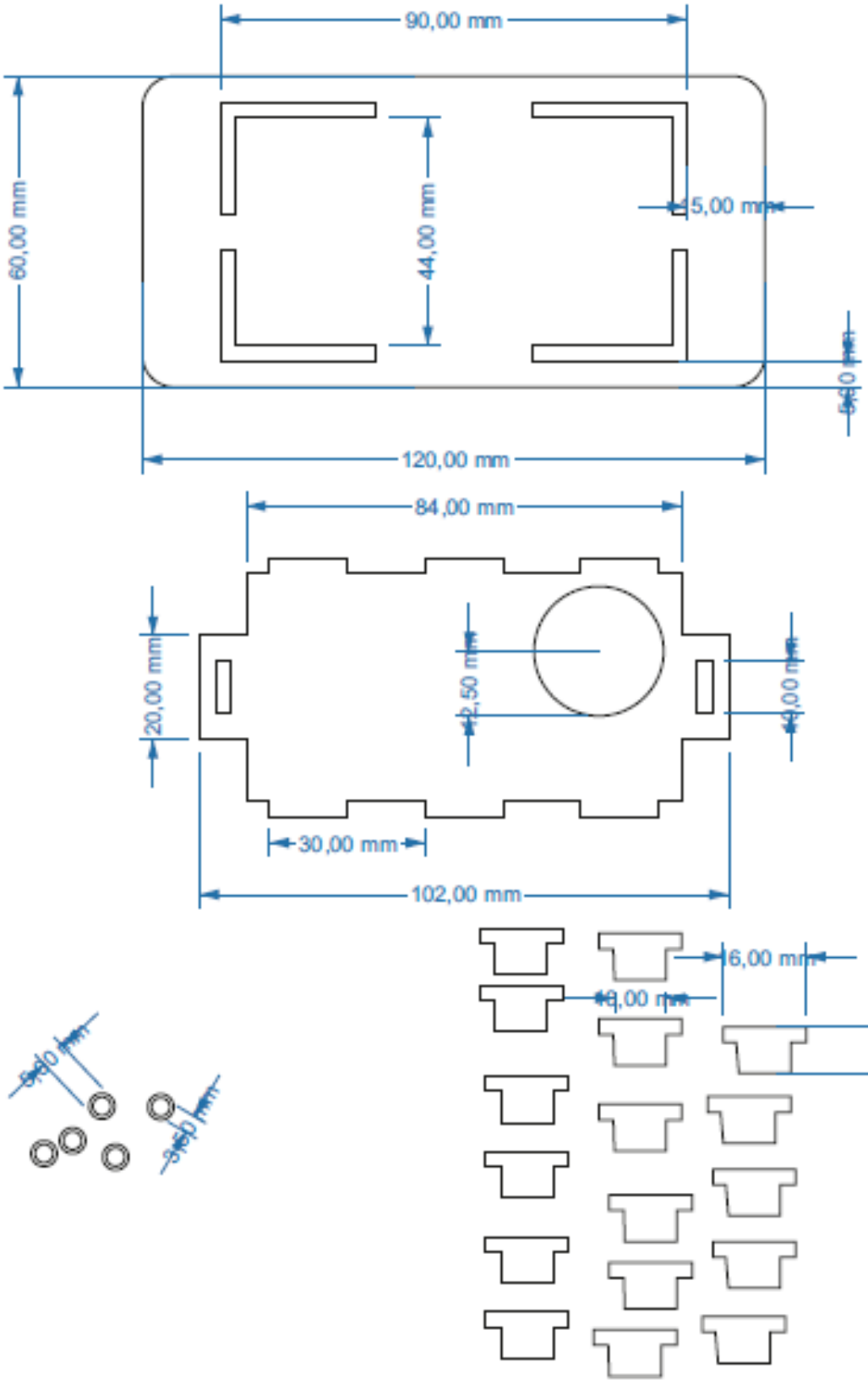
- Encender en ordenador en el caso de que estuviera apagado (Contraseña: impresora3d).
- Retirar posibles restos de madera del interior, colocar panel y configurar la altura del láser sobre el tablón a la dada por el groso de la siguiente pieza.
- Encender extractor.
- Configurar la potencia del láser a 90% (Para madera).
- Abrir programa *corelLaser* del escritorio u abrir nuestro archivo .cdr en versión v12 con nuestro diseño 2D.
- Configurar dimensiones del tablón 600x400mm en la esquina superior izquierda, y distribuir las diferenes piezas del archivo dentro de él.
- Pinchar en Cutting. Dentro de este:
    a. Configurar velocidad del láser a 15mm/s.
    b. Realizar un preview de forma rectangular para ver si nuestro diseño cabe en el tablero.
    c. Configurar el número de repeticiones a 2.
    d. Hacer click en starting para comenzar.
    e. Cuando termine la primera repetición, hacer click en iniciar siguiente repetición.
- Fin: recoger piezas cortadas y tirar los deshechos a la basura a menos que tengan suficiente área para ser usados en otro corte.

Nota: si nos pidieran introducir KEY, sacar y volver a meter el pendrive de la figura y reiniciar el programa.

ANEXO III: ACOTACIONES DEL CHASIS




19,00 mm
20,00 mm
75,00 mm
70,00 mm
60,00 mm
60,00 mm
20,50 mm
30,00 mm
30,00 mm
62,00 mm
110,00 mm
50,00 mm
6,00 mm
30,00 mm
5,00 mm
2,50 mm
37,00 mm
62,00 mm


90,00 mm
60,00 mm
44,00 mm
5,00 mm
120,00 mm
84,00 mm
20,00 mm
30,00 mm
102,00 mm
6,00 mm

REFERENCIAS

[1] Ignacio Alvarado Aldea, "Práctica control ventilador completa" .

[2] Ignacio Alvarado Aldea, "Práctica control ventilador sin registros".

[3] Jose Antonio Borja Conde, "Desarrollo de un robot autoequilibrado basado en Arduino con motores paso a paso", Trabajo de fin de Grado año 2018.

[4] Cecilia González González, "Mejora del software de un vehículo autoequilibrado de tipo péndulo invertido de bajo coste", Trabajo de fin de Grado año 2016.

[5] Anibal Ollero Baturone y Guillermo Heredia Benot "Apuntes de la asignatura Robótica Avanzada", curso 2017-2018.

[6] Joaquín Ferruz Melero, "Apuntes de la asignatura de Informática Industria', curso 2018-2019'

[7] Enrique Drake Moyano, "Apuntes de fundamentos físicos de la Ingenieria", asignatura Física I, curso 2014-2015.

[8] Manuel Toscano Jiménez, "Mecánica de Ingenieros Industriales", asignatura Ampliación de Física, curso 2015-2016.

[9] Daniel Rodriguez Ramirez y Carlos Bordons Alba, "Apuntes de Ingeniería de Control", del 04/05/2017.

[10] Luis Llamas, "Uso de la imu MPU6050', en su blog de Internet con enlace' https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/

[11] Configuración Bluetooch HC-05 mediante comandos AT, en la página https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html

[12] Configuración Bluetooch HC-06 mediante comandos AT, en la página https://naylampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html

[13] Luis Llamas, "Uso del joystik Numshuk de la Wii', en su blog de Internet con enlace' https://www.luisllamas.es/arduino-mando-wii-wiichuck/

[14] Páginas de Referencias de Arduino, con enlace https://www.arduino.cc/reference/en/

[15] Página web https://www.amazon.es/

[16] Datasheet del Atmega 328P

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[17] Datasheet del controlador V44 A3967

https://www.sparkfun.com/datasheets/Robotics/A3967.pdf

[18] Tutorial del prorama CorelDRAW en Youtube

https://www.youtube.com/watch?v=6uHRKhSa6Cw

[19] Ignacio Alvarado Aldea,''Identificación dinámica de un reactor con la función fmincon de Matlab''

[20] Teodoro Alamo Cantarero,''Control óptimo y satisfacción de restricciones''.Apuntes de Ingenieria de control 2017-2018.

[21] Ignacio Alvarado Aldea, David Muóz de la Peña y C. Gonzalez, ‘*’Low cost self balancing robot for control eduaction*''.

[22] Victor Manuel Villalar Lara, ‘Control predictivo de un vehículo tipo Segway'', trabajo de Fin de Grado 2017