

UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA
INSTITUTO DE ELECTRÓNICA Y MECATRÓNICA
INGENIERÍA EN MECATRÓNICA



"Desarrollo de un robot rueda con
movimiento en un solo plano"

Tesis

Para obtener el título de:

Ingeniero en Mecatrónica

Presenta:

Mario Enrique Herrera Cordero.

Director:

Dr. Manuel Arias Montiel.

Co-Directora:

Dra. Esther Lugo González.

Huajuapán de León, Oaxaca, México, Julio de 2020.

Índice

| | |
|--|-----------|
| Índice general | v |
| 1. Introducción | 1 |
| 1.1. Antecedentes | 2 |
| 1.1.1. Robot de una rueda | 2 |
| 1.1.2. Robot esfera | 5 |
| 1.1.3. Robot esfera inercial | 7 |
| 1.1.4. Configuraciones del robot esfera basado en péndulo | 7 |
| 1.1.5. Robot rueda inercial | 8 |
| 1.2. Planteamiento del problema | 9 |
| 1.3. Justificación | 9 |
| 1.4. Limitantes | 9 |
| 1.5. Hipótesis | 10 |
| 1.6. Objetivo | 10 |
| 1.6.1. Objetivo general | 10 |
| 1.6.2. Objetivos particulares | 10 |
| 1.7. Metodología | 11 |
| 1.8. Estructura de la tesis | 13 |
| 2. Desarrollo y modelo matemático del robot rueda | 15 |
| 2.1. Diseño del robot rueda | 15 |
| 2.2. Modelo matemático del robot rueda | 21 |
| 2.2.1. Energías cinéticas y potenciales | 22 |
| 2.2.2. Ecuaciones de Euler - Lagrange | 29 |
| 2.2.3. Análisis y validación del modelo dinámico | 33 |
| 3. Linealización de las ecuaciones dinámicas y control del robot rueda | 43 |
| 3.1. Linealización de las ecuaciones de la dinámica del robot rueda | 43 |
| 3.2. Función de transferencia | 50 |
| 3.3. Control de la velocidad del robot rueda | 52 |
| 3.4. Simulación del control de velocidad del robot en ADAMS View y filtro Kalman en Matlab | 62 |

| | |
|--|------------|
| 4. Manufactura e implementación del robot rueda | 69 |
| 4.1. Sistema electrónico | 69 |
| 4.1.1. Tareas necesarias a realizar y elección de dispositivos | 69 |
| 4.1.2. Conexión de dispositivos | 71 |
| 4.1.3. Programación del microcontrolador ATmega328p | 72 |
| 4.1.4. Pruebas de programación y simulación | 72 |
| 4.1.5. Construcción física del circuito | 72 |
| 4.2. Comunicación circuito del robot y Matlab Simulink | 73 |
| 4.2.1. Construcción y programación de diagrama a bloques de Arduino | 73 |
| 4.2.2. Pruebas | 74 |
| 4.3. Manufactura de elementos de PLA y acrílico | 77 |
| 5. Modelo matemático de la planta real e implementación del algoritmo de control. | 81 |
| 5.1. Modelo matemático | 81 |
| 5.1.1. Respuesta del robot con PWM máximo de entrada | 85 |
| 5.1.2. Estimación del par torsional máximo del sistema | 88 |
| 5.1.3. Simulación de la ecuación de fricción indefinida (fr) | 88 |
| 5.2. Diseño de control del sistema físico | 91 |
| 5.3. Pruebas del control en el sistema físico | 97 |
| 6. Conclusiones y trabajos a futuro | 105 |
| 6.1. Conclusiones | 105 |
| 6.2. Trabajos a futuro | 106 |
| A. Dibujos técnicos del robot rueda | 109 |
| B. Código de programación de microcontrolador de Proteus 8 | 119 |
| C. Código programación de microcontrolador con giroscopio | 123 |
| D. Instalación de paquetes complementarios en Matlab Simulink | 133 |
| D.1. Instrucciones de instalación de paquetes | 133 |
| D.2. Verificación de paquetes instalados | 147 |
| E. Código para decodificación de los datos de posición | 149 |
| F. Código para codificación de PWM | 151 |
| G. Configuración de los parámetros de simulación y simulación en tiempo real | 153 |
| H. Dibujos técnicos de implementación del robot rueda | 159 |
| I. Presencia del trabajo de tesis en congresos, memorias y artículos | 175 |

| | |
|--------------------|------------|
| J. Glosario | 177 |
| Referencias | 177 |

Capítulo 1

Introducción

Los robots han tenido una gran influencia en las situaciones o tareas que están fuera del alcance del ser humano, como aquellas en las que se pone en peligro la vida humana, las que representan tareas repetitivas y que requieren un gran esfuerzo físico o que requieren precisión y velocidad. Por lo que se han propuesto ideas para resolver estos problemas a los que se enfrenta el humano. Entre algunas de esas ideas está el poder hacer más eficiente el transporte y/o rescate de personas u objetos por medio del uso de robots.

Existen numerosas clasificaciones para los robots, en función de su generación de movimiento (1G, 2G, 3G, 4G, 5G), su nivel de inteligencia, su aplicación, su nivel de control y de acuerdo a su arquitectura [1]. Dentro de estas clasificaciones están los robots fijos, tales como los robots manipuladores, y los robots móviles (como los robots sobre ruedas, con patas, submarinos y aéreos)[2].

La mayoría de los robots móviles con ruedas usan más de dos ruedas, sin embargo, se ha realizado un gran esfuerzo para estabilizar robots con sólo dos ruedas debido a la versatilidad que tienen para operar en el medio ambiente donde cotidianamente se desenvuelve el hombre [3]. Este tipo de robots se denominan diferenciales ya que la dirección se logra a partir de la diferencia de velocidades entre ambas ruedas [4]. Sin embargo, debido a que los robots de 2 o más ruedas ocupan mucho espacio para desplazarse en comparación con el robot de una rueda delgada y liviana, éste último representa una alternativa para reemplazar a los robots de dos o más ruedas.

El principio de funcionamiento del robot rueda inercial se basa en el movimiento del péndulo, el cual genera la fuerza de inercia para el movimiento del robot. Por su principio de operación y su forma, el robot rueda inercial puede desplazarse por espacios más estrechos que otros robots convencionales de dos ruedas. Además, su estabilidad dinámica le permite mejorar su navegación en ambientes dinámicos y estrechos.

Desde otra perspectiva, el hecho de tener un robot compacto y que pueda realizar tareas en ambientes robustos, en donde se requiere que los componentes del robot se mantengan protegidos, genera la idea de un robot esfera, el cual es capaz de desplazarse en cualquier ambiente robusto, gracias a que el robot es un sistema holonómico [5]. Por otro lado, el tener una esfe-

ra para ciertos propósitos puede resultar problemático en los casos en los que los lugares de acceso son estrechos, de modo que este robot es eficiente en ciertas aplicaciones en las que el ambiente es robusto. Debido a que el robot rueda tiene una gran similitud con el robot esfera, se presenta el siguiente estudio del estado del arte.

1.1. Antecedentes

El péndulo simple ideal es un sistema conservativo en el cual la energía potencial asociada con el desplazamiento se retiene en el sistema cuando oscila, por lo que su modelación es de importancia ya que algunos sistemas se basan en el péndulo, sin embargo, con los péndulos dobles y triples puede inducirse un movimiento caótico que a su vez requiere matemáticas aún más sofisticadas para poder modelarse correctamente. Todo el sistema de péndulo se hace más complejo cuando el péndulo es accionado por un par variable en su punto de suspensión y por lo que se eliminan los límites de su amplitud. Entonces el comportamiento del péndulo se vuelve más complejo y consecuentemente más difícil de modelar matemáticamente. En las últimas décadas, matemáticos y físicos han trabajado conjuntamente en este problema [6]. El péndulo invertido, es un sistema bastante conocido dentro del área de ingeniería, esto debido a su naturaleza no lineal, inestable, fácilmente perturbable y el problema que lleva consigo el lograr estabilizarlo en su punto de equilibrio no estable. Además, ha despertado el interés de varios investigadores y aficionados al control y a la robótica.

En años recientes, los investigadores han aplicado la similitud del modelo dinámico de un péndulo invertido móvil a diversos problemas como el diseño de robots humanoides caminantes, sillas de ruedas robóticas y los sistemas personales de transporte [7]. Por ejemplo, el Segway; que se utiliza con fines comerciales y el prototipo del proyecto VECNA B.E.A.R; robot que se puede utilizar para operaciones militares de Gobierno [8]. Entre los trabajos reportados en la literatura existente, y que están relacionados con el péndulo invertido, pueden citarse los siguientes: El péndulo invertido sobre un carro [9], el péndulo con disco inercia [10], el péndulo esférico invertido [11], el péndulo de Furuta [12], el Pendubot [13], el Acrobot [14], el sistema bola-viga [15], el sistema de aterrizaje y despegue vertical de un avión [16], Ballbot [17], el sistema oscilador traslacional con actuador rotacional [18], el Robot esfera [5].

El robot rueda con locomoción inercial por péndulo tiene un principio de funcionamiento muy parecido al de un robot unicycle, por lo que es de importancia citarlo.

1.1.1. Robot de una rueda

Los robots móviles con ruedas pueden clasificarse según el número de ruedas, es decir, los robots monociclo o de una rueda, los robots móviles de dos ruedas y los robots multi-ruedas. El proyecto se centrará en robots de una rueda, en donde existen numerosas configuraciones de robots de este tipo, y las oscilaciones de los componentes son el problema a solucionar, de modo que el sistema se asemeja a un péndulo en donde hay que aplicar control ante las oscilaciones del péndulo. A continuación, se muestran algunos robots de una rueda.

Robot péndulo de una rueda

Este tipo de robot está conformado por la estabilización de un péndulo de manera giroscópica, en donde el momento del giroscopio actúa directamente sobre el pivote del péndulo, cuya magnitud está restringida por la precesión giroscópica [19].

Un girorotor en estado de precesión genera un momento de giro, el cual llega a ser igual al valor del producto de su velocidad de precesión y momento angular. Este fenómeno en la mecánica se conoce como **efecto giroscópico**. Las aplicaciones de giroscopios incluyen sistemas de navegación inercial, donde las brújulas magnéticas no funcionan o no serían lo suficientemente precisas y la estabilización de vehículos voladores, como helicópteros controlados por radio o vehículos aéreos no tripulados [20, 21, 22]. Un pequeño giroscopio puede incluso ser utilizado como un dispositivo de dirección en un robot de una sola rueda debido a la alta eficiencia de trabajo del momento giroscópico.

El equilibrado longitudinal es fácil de implementar para este tipo de robot porque el pivote de su cuerpo superior está fijado en el eje de accionamiento de la rueda, que proporciona directamente el par de control para el equilibrado longitudinal [23]. Por el contrario, el efecto del control de balanceo lateral es débil para su estructura bajo actuación; el par de control (par de inercia) para el equilibrado lateral se obtiene acelerando (o desacelerando) una rueda inercial instalada en la parte superior del robot [24]. Así, el movimiento de oscilación lateral exhibe el comportamiento del péndulo de la rueda de reacción [25].

El robot. al tratarse de una rueda tiene un principio de funcionamiento muy parecido al de un robot monociclo.

Robot monociclo

Los robots monociclo están equipados con una sola rueda, tienen tamaños considerablemente compactos. Además, los robots monociclo son más flexibles que otros robots con ruedas, porque el primero entra en contacto con el suelo en un solo punto. Por lo tanto, los robots monociclo pueden alcanzar un radio de rotación cero. De modo que, estos robots tienen aplicaciones potenciales, como el libre movimiento y el transporte de carga en espacios estrechos. [26]

Una de las cuestiones más importantes relativas a los robots monociclo es el problema del equilibrio lateral. Sin embargo, cuando se quiere controlar manualmente este dispositivo resulta ser algo fácil para el ser humano. Por el contrario, para los robots monociclo sin manipulación humana, los métodos existentes de balance lateral se basan en ruedas de inercia horizontal [27, 28], ruedas de inercia vertical [29] y giroscopios de alta velocidad [19]. Por lo tanto, el equilibrar de manera estable y autónoma es relativamente difícil. El uso de giroscopios de alta velocidad ha sido ampliamente adoptado en estudios previos. En las investigaciones recientes sobre los robots monociclo, los métodos de balance lateral se aplican en las máquinas de prototipado para resolver el problema del equilibrio lateral.

El sistema del robot monociclo basado en una rueda inercial horizontal es un sistema humanoide que imita la función de equilibrio de los seres humanos. Este sistema ha realizado el balance lateral de robots monociclo. Sin embargo, su algoritmo de control es relativamente complejo y su capacidad de recuperación rápida después de la perturbación necesita ser mejorada. [30, 31].

El modelado dinámico del robot monociclo basado en la rueda de inercia vertical es fácil de obtener. Además, el acoplamiento dinámico de la dinámica lateral y longitudinal es relativamente pequeño. Por lo tanto, el modelo de dinámica compleja podría ser simplificado [25, 32]. Sin embargo, los requisitos para el funcionamiento de los motores de accionamiento de la rueda de inercia vertical son estrictos, particularmente cuando la inercia de rotación es grande. Además, el motor necesita ser acelerado y alcanzar la saturación para continuar obteniendo un par de equilibrio [33, 34].

Robot de una sola rueda

El control de equilibrio de un robot móvil de una sola rueda se consigue mediante la fuerza giroscópica inducida por un sistema de cardán. Hay dos configuraciones del sistema cardán, configuración horizontal y vertical, dependiendo de cómo se genera la fuerza giroscópica. La configuración horizontal es una configuración convencional usada en sistemas de una rueda debido a la robustez. La configuración vertical del sistema de cardán tiene ventajas de menor consumo de energía pero desventajas de fácil inestabilidad con respecto a la configuración horizontal. Las características típicas de la configuración vertical se verifican a través de estudios experimentales de equilibrio de tareas de control [35].

En otros aspectos de la aplicación de control de equilibrio, el control de robots móviles con menos ruedas es bastante difícil y exigente. Los robots móviles son siempre estables cuando tienen tres o cuatro ruedas. Sin embargo, cuando el número de ruedas es inferior a tres, la técnica de control de equilibrado se convierte en la cuestión más importante a resolver. Segway es uno de los vehículos de dos ruedas mejor conocidos que requieren en mayor medida de las técnicas de equilibrado. El Segway de dos ruedas ha inspirado a muchos investigadores a usar el concepto de péndulo invertido para ayudar a las personas a desplazarse de un lugar a otro [35].

Desde un punto de vista de la investigación, Segway requiere esfuerzos extensos en el diseño, la detección y el control, de modo que muchos investigadores y educadores se han motivado por un largo tiempo para desarrollar sus propios robots móviles de dos ruedas con diferentes tamaños [36, 37, 38, 39, 40, 41]. Además, el número de ruedas de robot móvil se ha reducido a uno, cuyo control resulta más difícil [25, 29, 42, 43, 44, 45, 46, 47]. En los robots de una sola rueda se requiere que todo el hardware sea colocado dentro de la rueda y que ésta mantenga el equilibrio sin caerse. Las dificultades provienen del diseño, detección y control, en la estructura restringida de un robot móvil de una sola rueda.

Bajo la limitación antes mencionada, se han desarrollado pocos robots de una sola rueda. El accionamiento de las ruedas individuales incluye las ruedas de reacción, el momento de giro de control o los ventiladores acanalados. El propósito principal de estos actuadores es mantener la postura de equilibrio del robot. Gyrover ha sido desarrollado con el propósito de explorar

planetas en el universo. El diseño de sus actuadores y de su control se han mejorado en los años sucesivos [45, 46]. Gyrover puede mantener el equilibrio mientras conduce en su terreno. Un robot de una sola rueda debe ser autónomo para navegar o para realizar sus tareas de vigilancia. La navegación autónoma del robot de una sola rueda requiere resolver muchos problemas, tales como un sistema de alimentación de carga automática, un algoritmo de navegación autónoma, un sistema de autocontrol y un mecanismo de auto-movimiento. Entre ellos, el mecanismo de auto-montaje que permite al robot descansar y moverse es la capacidad más importante.

Para mantener a la rueda en su posición vertical, el control de la fuerza giroscópica juega un papel importante. Hay dos configuraciones en el diseño, configuraciones horizontales y verticales, dependiendo de la ubicación del sistema cardán que consiste en un volante, un motor de volante y un motor de inclinación. Puede considerarse como un problema de control no lineal del péndulo invertido. El péndulo invertido es un problema clásico en la teoría de control y se establece en la literatura técnica como una prueba práctica de algoritmos de control. Por lo tanto, el problema de equilibrio del robot de una sola rueda es un reto.

1.1.2. Robot esfera

El robot esfera es diseñado para mantener todo componente sellado dentro de la esfera, de tal modo que estará protegido de todo ambiente robusto en donde operará holonómicamente, así como también será capaz de rebotar ante colisiones de manera rápida y no destructiva [48].

Las esferas robóticas se pueden clasificar en 3 tipos, de acuerdo a su mecanismo interno de funcionamiento:

- 1.— **Barycenter Offset (BCO):** Las esferas robóticas con el principio de BCO tienen la peculiaridad de desfazar el centro de masa de la esfera para así generar una inercia que provoca que la esfera ruede, tratando de llegar nuevamente a su estado de equilibrio. Por lo que al cambiar constantemente de manera sincronizada el centro de gravedad de la esfera, se llega a lograr un movimiento uniforme del robot esfera. Sin embargo, este tipo de arquitectura está limitada al no poder desfazar su centro de masa más allá de la carcasa de la esfera, por lo que su potencia está limitada [5].
- 2.— **Shell Transformation:** Este tipo de robot tiene la peculiaridad de deformar su carcasa para producir su avance, de modo que, dependiendo de su diseño, este tipo de robot puede ser más versátil que un robot esfera basado en un BCO. Sin embargo, éste tiene instalado un sistema mecatrónico interno muy complicado para hacer posible la deformación de la esfera [5].
- 3.— **Conservations of Angular Momentum(COAM):** Debido a las limitaciones en cuanto a potencia de los sistemas BCO, se tiene que la investigación de los robots esfera tuvo que extenderse y esto conllevó al principio de gobernabilidad, en donde se tiene la aplicación del concepto *control de momento de giroscopios*(CMGs), esto es, al girar un gran

volante rápidamente alrededor de un eje, las leyes de conservación del momento angular pueden usarse para controlar el movimiento de la esfera. Haciendo uso de este método se relaciona el par de salida del mecanismo interno a la velocidad angular de los CMGs. Como la velocidad de los CMGs incrementa por lo tanto el par torsional de salida también incrementará. Este es un método más reciente para obtener un par torsional de salida más grande que los robots esferas basados en BCO [5].

El robot rueda se basa en el principio BCO, ya que el sistema cambia constantemente su centro de masa para así poder generar su movimiento, de modo que es de interés el conocer los tipos de mecanismos que se han usado para implementar la arquitectura BCO.

Robot esfera basado en una rueda

El primer robot móvil esférico fue desarrollado por Halme en 1996 [49]. La propulsión fue derivada de una rueda en contacto con el fondo de la esfera. Encima de la rueda está la Unidad de Accionamiento Interior (Inside Drive Unit - IDU, por sus siglas en inglés) con potencia y comunicación. La rueda y la IDU están integrados dentro de un solo eje soporte. En la contraparte del fin de la rueda de potencia está otra rueda de estabilización. Por la rueda motorizada de gobierno, la esfera tiene la habilidad de girar.

Este diseño fue también configurado con dos ruedas en contacto con el fondo de la superficie como es mostrado por una implementación de K.Husoy [50].

En 1997, A. Bicchi propuso un diseño que fue propulsado por un pequeño carro reposando en el fondo de la esfera [51]. El carro puede ser conducido para cambiar la dirección de la esfera. El robot fue modelado usando una combinación de la cinemática de monociclo y del sistema bola-viga.

Las ventajas de los diseños de esferas con ruedas son, una simple estructura mecánica y los motores pueden ser más pequeños ya que no requieren mucho par torsional. El sistema puede ser también holonómico o no holonómico dependiendo de la configuración de la rueda. Esto también ayuda a que el sistema pueda ser modelado por modelos matemáticos bien conocidos, haciendo el control más fácil.

Robot esfera basado en un péndulo de 2GDL

Un robot esférico simple puede ser desarrollado usando su diseño basado en un péndulo. El robot esfera llamado Rotundus [52] consiste de un motor unido al eje horizontal que pasa a través de la esfera, en el centro hay un péndulo que cae. Cuando el motor es activado, la esfera se moverá tanto como inercia tenga el peso del péndulo, aunque para el giro es más fácil mover el péndulo a la dirección deseada que mover la carcasa. El péndulo se puede mover a la izquierda y derecha, causando que el robot gire.

Debido a su peculiar forma de funcionamiento se tiene que este está disponible al mercado, en la página [http : //www.rotundus.se/](http://www.rotundus.se/)

Robot esfera basado en la reubicación del centro de masa

La idea detrás de este tipo de robot esférico de propulsión, es redistribuir masas en cierto orden para cambiar el centro de gravedad, continuamente redistribuyendo las masas en una cierta manera que permitirá a la esfera moverse en alguna dirección. R. Mukherjee en su diseño de robot esfera, propone un cuerpo central con pesos distribuidos radialmente a lo largo de radios fijos dentro de la superficie de la esfera [53]. Los pesos pueden ser independientes a los motores o los mismos motores de desplome, los cuales se moverán a lo largo de los ejes para cambiar el centro de masa. Otro diseño similar es propuesto por A.Javadi y P.Mojabi, y su robot fue llamado August [54]. En este diseño, cuatro ejes son montados en un patrón tetragonal. Los motores a pasos en el centro del robot hacen avanzar el peso en los ejes con un tornillo. Este diseño es más fácil para modelar de los robots esfera ya que sin pesos, Agust tiene un centro de masa localizado en el centro geométrico.

Una alternativa para masas cambiantes en barras rígidas, es la propuesta por Lux [50]. La carga útil para la inercia está dada por la unidad de control y el sensor, mediante el uso de malacates, los cables pueden ser retraídos y liberados para cambiar el centro de masa. El número de malacates pueden ser reducidos mediante resortes, de modo que los cables regresen sin la necesidad de malacates. Una ventaja de estos diseños es que pueden mover la masa para el centro y entonces solo se desliza, ahorrando energía cuando hay viento o está descendiendo una colina.

1.1.3. Robot esfera inercial

En años recientes, una nueva configuración de robot esférico ha logrado acaparar la atención de la comunidad científica. Se trata de una configuración de péndulo invertido conocida como esfera inercial. Es un robot que se adapta a diferentes entornos, y que además posee características que lo hacen robusto. Existen líneas de investigación relacionadas con este tipo de robot en las cuales se busca mejorar la estructura mecánica del robot y los esquemas de control aplicados al mismo, dejando abierta una amplia gama de aplicaciones [55].

En las siguientes subsecciones se mostrarán los tipos de robot de esfera basados en péndulo.

1.1.4. Configuraciones del robot esfera basado en péndulo

Esfera con eje fijo

La configuración más común de mecanismos internos de esferas robóticas es la denominada masa pendular con eje fijo (en inglés se identifica como "ballast mass with fixed axis").

Este sistema está compuesto por un eje fijo al cuerpo esférico, el cual atraviesa de extremo a extremo y pasa por su centro de gravedad (CG). Dicho eje sirve como base de un sistema pendular de dos grados de libertad (GDL), donde el primero de ellos gira alrededor del eje fijo (eje transversal) y el segundo en sentido perpendicular del primero (eje longitudinal). En el extremo del sistema pendular se tiene una masa que permite mover el CM fuera del CG de la esfera [55].

Esfera con eje móvil

Esta variante denominada masa pendular con eje móvil (en inglés se identifica como "ballast mass with moving axis") cuenta con un sistema pendular interno, al igual que el mecanismo anterior, pero en este caso posee un GDL adicional que permite al eje principal desplazarse. Rollo es el único robot encontrado en la revisión bibliográfica de este tipo, fue desarrollado por Ylikorpi [56]. Bajo un proyecto iniciado en el laboratorio de tecnología de automatización, de la Universidad de Tecnológica de Helsinki(HUT) y se presenta como resultado del refinamiento de las esferas tipo cuerpo central con muelle.

Sin embargo, aunque las posibilidades de movimiento de este tipo de sistema se aumentan con el GDL adicional, acercando sus prestaciones a las de un vehículo holonómico, la adición de este GDL incrementa no sólo las pérdidas energéticas por un actuador adicional, sino que además aumenta considerablemente la complejidad de construcción.

Esfera inercial con 4 péndulos

El robot de cuatro péndulos utiliza estos para cambiar el centro de masa del sistema de manera que el sistema pueda girar. Dichos cuatro péndulos rotan en los cuatro ejes tetrahédricos [57].

1.1.5. Robot rueda inercial

El péndulo invertido es un problema clásico en la teoría de control, ya que es un sistema inestable debido a que su centro de masa se encuentra por encima de su punto de pivote, por esta naturaleza es muy utilizado para pruebas de algoritmos de control. En el caso del robot rueda, el péndulo podrá oscilar de un punto de mayor energía potencial a uno de menor energía potencial y viceversa, ya que de esta manera se logrará obtener la aceleración en un sentido, sin embargo, dicha oscilación tiene que ser controlada para que el péndulo no llegue a pasar en el otro sentido de la rueda, ya que en caso contrario se producirán pérdidas, es decir, se genera una fuerza resultante en sentido opuesto al avance, por lo que es allí donde el control se vuelve de importancia en el desarrollo del robot [4].

El robot al tratarse de una sola rueda no puede ser tripulado, de manera que es necesario que sea autónomo o guiado a través de control remoto. El estudio de la navegación autónoma para este tipo de robots tiene su lugar en Japón donde en 1996 se construye un robot que realiza una navegación autónoma con seguimiento de trayectoria, donde se utilizan tres controladores independientes, uno para el balance y la velocidad lineal, otro para la dirección y por último uno para la navegación [4]. También se han desarrollado ciertos métodos de control especialmente para transporte de objetos, donde se toma en cuenta la carga, así como perturbaciones tales como el cambio de peso en los objetos. Más recientemente se proponen nuevos métodos de control para este tipo de robots utilizando técnicas de control invariante en el tiempo o control adaptativo difuso [4].

1.2. Planteamiento del problema

En este trabajo el problema presente es un conjunto de tareas como el diseño, el análisis y el control de un robot de una sola rueda con locomoción inercial mediante un péndulo de un sólo grado de libertad, de modo que, se abordará el tema del diseño mecánico de la estructura del robot y del sistema de transmisión de movimiento, así como del péndulo para la locomoción inercial, el desarrollo del modelo dinámico del robot, la construcción e instrumentación del prototipo físico que permita la implementación de algoritmos de control para su correcto funcionamiento y la solución de los problemas de comunicación entre el robot y la PC que fungirá como estación de control.

1.3. Justificación

Se propone el desarrollo de una alternativa a los robots esfera mencionados en el estado del arte para solventar algunas de sus desventajas en cuestión del espacio que necesitan para desplazarse. Por otra parte, este proyecto es de interés ya que el robot rueda con locomoción inercial mediante un péndulo representa un sistema no lineal con dinámica acoplada para el cuál se pueden diseñar e implementar diversos algoritmos de control. Además, al tratarse de una propuesta innovadora, se pueden desarrollar ecuaciones de diseño que permitan el desarrollo y la optimización de este tipo de sistemas en aplicaciones futuras.

Por otro lado, se tiene que la carrera de ingeniería en mecatrónica es un área multidisciplinaria, en donde es de importancia poner en práctica los conocimientos adquiridos durante la misma, dichos conocimientos aplicados a la presente propuesta comprenden las siguientes áreas:

- 1.- **Mecánica:** Se diseñará el mecanismo que mantiene el sistema estable de manera vertical.
- 2.- **Electrónica:** Se determina el uso de circuitos integrados tanto analógicos como digitales, sensores y actuadores. Además, también se implementará comunicación inalámbrica Bluetooth rueda-CPU.
- 3.- **Computación:** Se hará uso de código para programar los microcontroladores que rigen la comunicación, la lectura de los sensores y el accionamiento de los actuadores.
- 4.- **Control:** Se diseñarán e implementarán algoritmos de control para la velocidad y el sentido de movimiento del robot.

1.4. Limitantes

- * El robot estará limitado a sólo usar un grado de libertad del péndulo (la coordenada angular).

- * El robot será solamente capaz de desplazarse hacia adelante y hacia atrás sobre un plano horizontal.
- * La distancia máxima entre la estación de control y el robot será de 20 metros, dadas las limitantes de la comunicación mediante bluetooth.
- * El robot rueda tendrá aproximadamente un diámetro exterior entre 15 y 25 cm.

1.5. Hipótesis

Es posible desarrollar un robot móvil con locomoción inercial basada en un sistema de péndulo que represente una alternativa más a los robots rueda y esfera.

1.6. Objetivo

1.6.1. Objetivo general

Diseñar y construir el mecanismo de un robot rueda basado en la arquitectura BCO de un péndulo de 1 GDL y aplicar control para generar una inercia residual, que represente el sistema de locomoción del robot.

1.6.2. Objetivos particulares

- 1.- Diseñar la estructura mecánica del robot rueda inercial, incluyendo el mecanismo de locomoción inercial basado en péndulos.
- 2.- Obtener la dinámica del sistema y linealizarla alrededor de los puntos de operación.
- 3.- Diseñar algoritmos de control para el péndulo de 1GDL, para así controlar el desplazamiento del robot.
- 4.- Construir un prototipo de la estructura mecánica del robot rueda inercial.
- 5.- Instrumentar el prototipo con sensores y actuadores.
- 6.- Implementar la comunicación inalámbrica microcontrolador-CPU mediante bluetooth.
- 7.- Programar el control del robot microcontrolador y CPU-Matlab.
- 8.- Realizar pruebas en lazo cerrado del sistema completo.

1.7. Metodología

- 1.– Diseño del mecanismo rueda inercial mediante herramientas CAD/CAM.
 - * Se hará uso de la herramienta Solidworks para el diseño del mecanismo.
- 2.– Simulaciones con el Software ADAMS.
 - * Se hará uso de la herramienta ADAMS VIEW para la simulación.
- 3.– Obtener la dinámica del sistema mediante la metodología de Euler-Lagrange, validar las ecuaciones obtenidas mediante su solución numérica y simulaciones realizadas en SolidWorks y ADAMS.
- 4.– Linealizar el sistema en los puntos críticos en los que se desea que el péndulo del sistema sea estable y no oscile durante su funcionamiento.
- 5.– Aplicar una técnica de control al sistema para evitar las oscilaciones, dicha técnica se corroborará con los software de simulación ya sean Matlab (en el caso de las ecuaciones dinámicas) y SolidWorks o Adams (en el caso del sistema mecánico en simulación).
- 6.– Caracterización de los sensores y selección de actuadores.
 - 6.a) Seleccionar los motores que cumplan con la velocidad y el par de torsión para la respuesta del sistema.
 - 6.b) Caracterización del giroscopio MPU60-50.
 - * Uso de la comunicación TWI, analizar el diagramas de tiempos para la adquisición de datos.
 - * Uso de la comunicación alámbrica Microcontrolador – CPU para el almacenamiento de datos e interpretación de estos.
 - 6.c) Etapa de potencia de motores.
 - * Se implementará el circuito mostrado en la Figura 1.1, para la etapa de potencia, a partir del circuito integrado L293D.

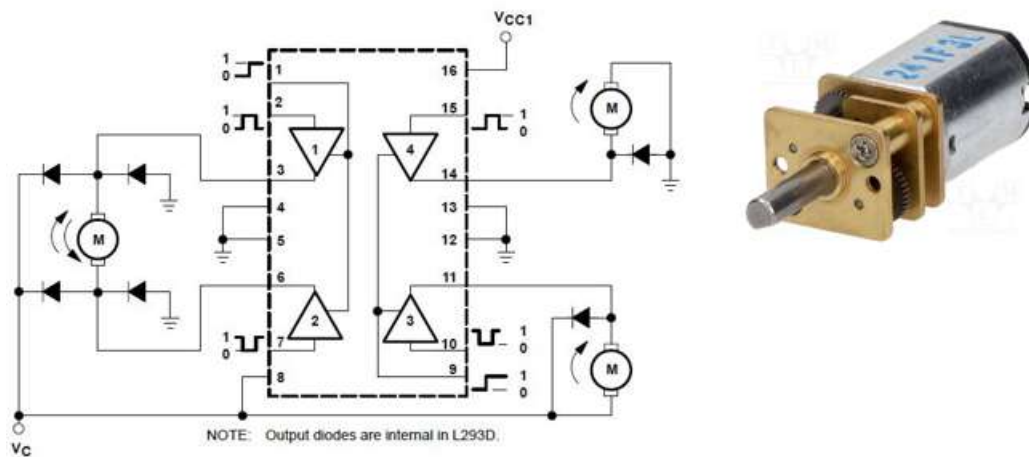


Figura 1.1: Circuito de potencia

- 7.— Resolver la problemática de la implementación de un giroscopio en el robot, posible modificación del diseño de la rueda.
 - * Puesto que un giroscopio está en la parte móvil del robot (el péndulo) se tiene que la comunicación entre el microcontrolador y el giroscopio puede representar ciertos problemas, ya que al haber un falso contacto es posible que se pierda la comunicación.
- 8.— Investigar la comunicación inalámbrica USART directa entre el microcontrolador de la rueda inercial y la CPU.
- 9.— Investigar la posibilidad de que la herramienta Matlab pueda adquirir los datos recibidos mediante el protocolo USART y hacer uso de ellos.
- 10.— Construir y ensamblar el robot rueda y realizar las conexiones correspondientes con los componentes electrónicos.
- 11.— Modificar la dinámica del sistema con las masas reales del sistema.
- 12.— Aplicar la técnica de control a la planta, en éste caso el controlador será analógico, aunque es de importancia señalar que las señales que envía el módulo de la rueda, son las posiciones en las que se encuentran los giroscopios y las que se reciben son valores de PWM para la actuación de los motores.
- 14.— Realizar iteraciones para el mejoramiento del control y ajustes del mismo.
- 15.— Pruebas finales y redacción del documento final.

1.8. Estructura de la tesis

La tesis consta de 7 capítulos, en donde el primer capítulo se basa en los antecedentes del proyecto de investigación, se establecen los objetivos, el planteamiento del problema y la justificación del trabajo, por otro lado, los consecuentes capítulos constarán de lo siguiente:

- * **Capítulo 2.** En este capítulo se propone el diseño conceptual del robot, se analiza la simetría de cada elemento del robot respecto al eje de rotación del mismo, para así poder estimar su energía tanto cinética y potencial, procediendo finalmente a la ecuación de Euler - Lagrange para hallar el modelo matemático del robot rueda.
- * **Capítulo 3.** Aquí se hace uso de la linealización por variables de estados alrededor de un punto de operación, de modo que, al tener dicha linealización se puede aplicar la metodología de asignación de polos y ceros mediante el Lugar Geométrico de las Raíces.
- * **Capítulo 4.** Se aborda la optimización, fabricación y ensamble de cada uno de los componentes que constituyen al robot rueda que propiamente es la parte mecánica y la parte electrónica.
- * **Capítulo 5.** En este capítulo se realizan pruebas experimentales del algoritmo de control en la plata física mediante Matlab y una tarjeta de adquisición de señales, se presentan también los resultados experimentales del sistema en lazo cerrado.
- * **Capítulo 6.** Finalmente se presentan las conclusiones y las propuestas de trabajos a futuro derivados del trabajo de tesis.

Capítulo 2

Desarrollo y modelo matemático del robot rueda

2.1. Diseño del robot rueda

El movimiento de los robots con locomoción inercial se basa en el cambio de su centro de masa, a este tipo de funcionamiento o arquitectura se le conoce como BCO (BarycenterOffset), por lo que de acuerdo al prototipo ideado con la arquitectura BCO se proponen los siguientes elementos:

- 1.— Una carcasa compuesta por dos partes individuales (ver Figura 2.1).

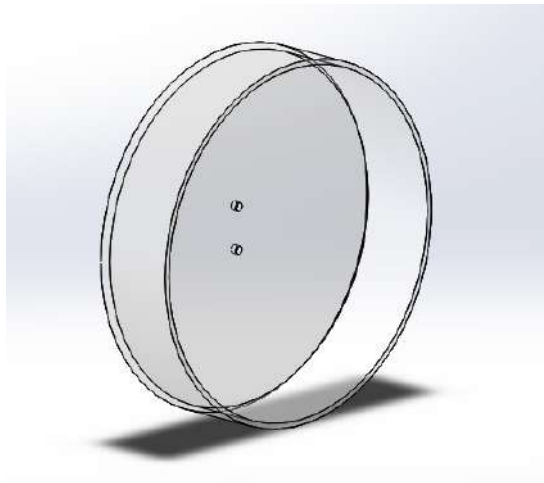


Figura 2.1: Carcasa de prototipo.

- 2.— Dos discos de base (ver Figura 2.2).

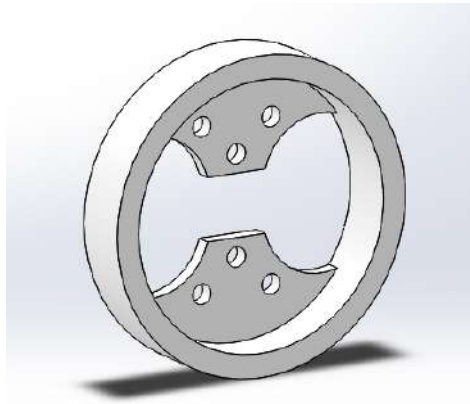


Figura 2.2: Disco base.

3.— Cuatro pernos para la unión entre la carcasa y los discos (ver Figura 2.3).

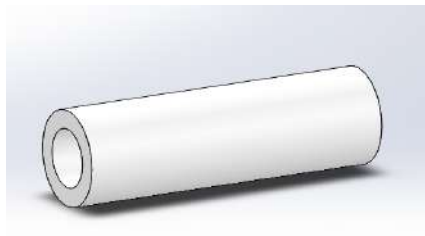


Figura 2.3: Perno.

4.— Una base para el péndulo (ver Figura 2.4).

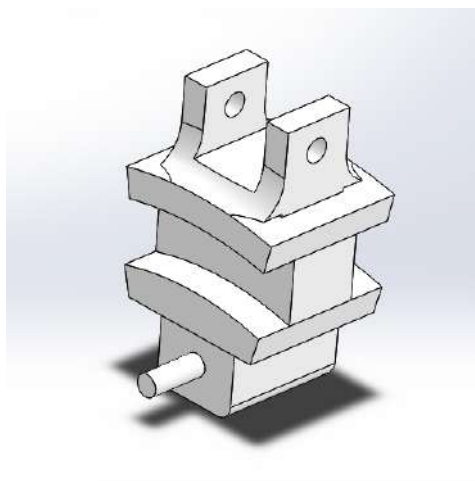


Figura 2.4: Péndulo base.

5.— Un péndulo (ver Figura 2.5).

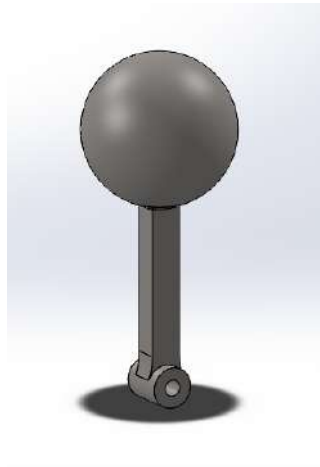


Figura 2.5: Péndulo.

6.— Un piñón (ver Figura 2.6).



Figura 2.6: Piñón (PD=0.5, número de dientes=30).

7.— Una corona (ver Figura 2.7).



Figura 2.7: Corona (PD=0.5, número de dientes=159).

Los materiales a utilizar se definen en el software Solidworks, donde la carcasa es de acrílico, el péndulo es de metal, y los elementos restantes son de PLA, el cual es el material que se usará en la impresión 3D.

Ahora bien, una vez obtenida la idea del principio del funcionamiento o arquitectura del robot, se tiene que optimizar el diseño del robot, ya que los elementos que lo componen no presentan simetría respecto al eje de rotación de la rueda, por lo que da como resultado un desbalance en el centro de masa e inercia de la rueda, siendo esto así, el robot solo estará en equilibrio en una posición específica, de modo que al no ser corregido este problema no se obtendrá una respuesta satisfactoria del control, así que el objetivo ahora es que todo elemento de la rueda presente simetría respecto a su eje de rotación.

La falta de simetría se encuentra en los Discos Base (ver Figura 2.2) de tal forma que se rediseña el disco base como se muestra en la Figura 2.8.

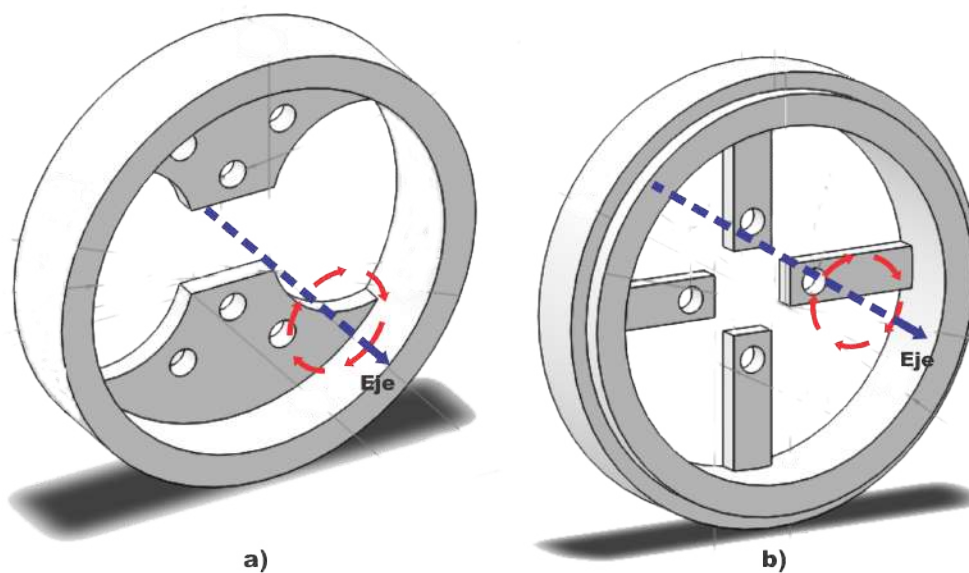


Figura 2.8: Comparación de Discos.

Otra problemática que presenta el primer diseño es que el motor solo genera fuerza de tracción en un lado del Robot Rueda, de modo que se genera un par torsional alrededor del eje del péndulo, como se puede ver en la Figura 2.9.

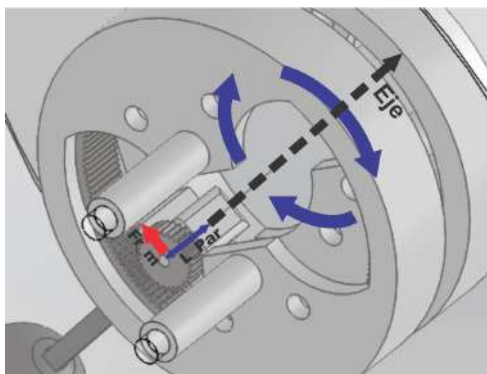


Figura 2.9: Par torsional interno.

Donde:

F_{r_m} = Fuerza resultante del motor.

L_{par} = Brazo de palanca.

La importancia en eliminar el par generado en el eje del péndulo radica en que se genera más fricción entre los Discos Base y el Péndulo Base, por lo que una manera sencilla de eliminar este par generado es extendiendo la flecha del péndulo base y colocar dos piñones, siendo esto así, los dos pares torsionales generados se cancelan (ver Figura 2.10).

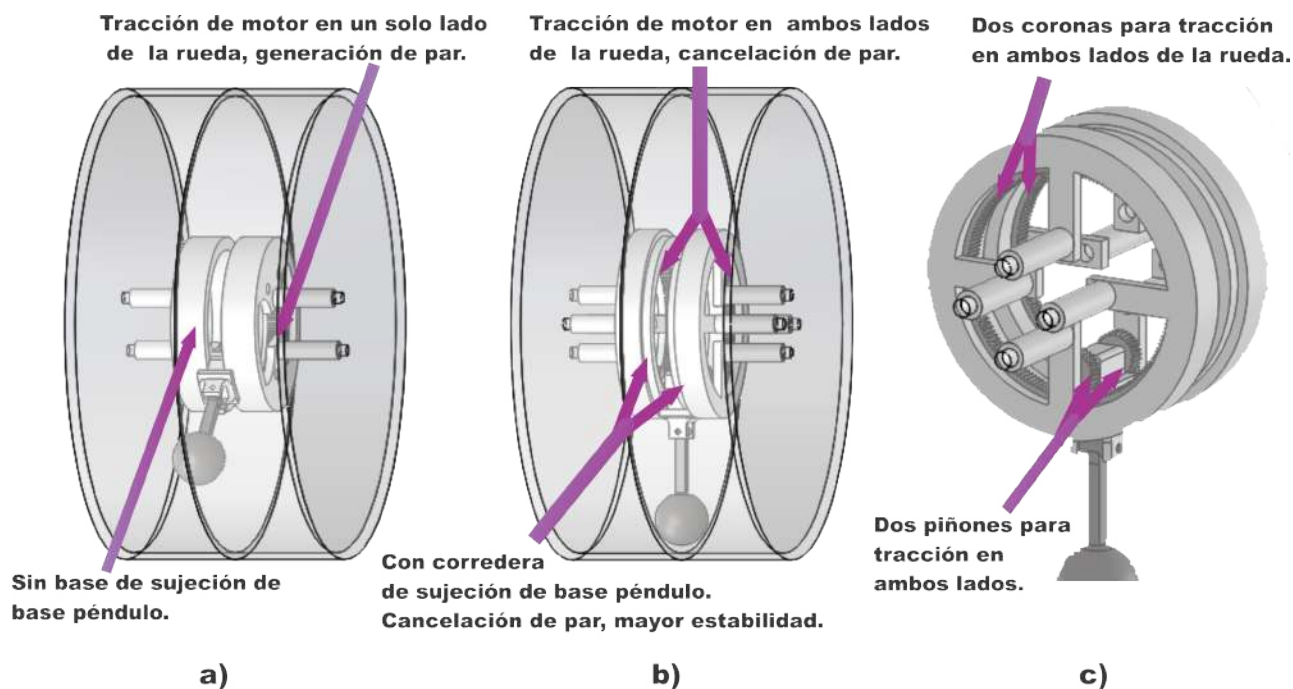


Figura 2.10: a)Diseño anterior idea del prototipo, b) y c) Diseño final para anulación de par de torsión.

Una vez expuesto todo lo anterior, se tiene que las diferencias entre los dos diseños son (ver Figura 2.11):

- 1.- Tracción en ambos lados de la rueda al colocar dos piñones a la flecha extendida del motor, que asegura la anulación de los pares torsionales generados.
- 2.- Carga equilibrada, la cual está relacionada con la forma simétrica del Disco Base respecto al eje de rotación de la rueda.
- 3.- Corredera del Péndulo Base, que asegura la anulación del par generado por el motor y que mejora la estabilidad de la rueda.

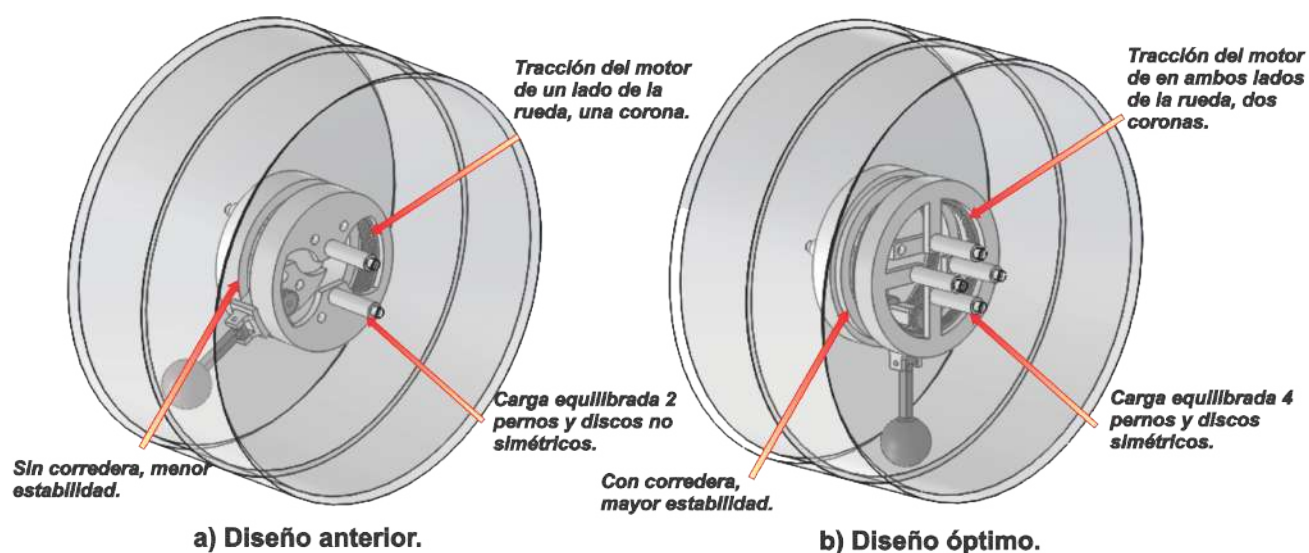


Figura 2.11: Comparación de diseño de idea y diseño de idea optimizada.

El diseño final de la rueda consta de los siguientes elementos:

- 1.- Una carcasa compuesta por dos partes individuales (ver Figura 2.1).
- 2.- Dos discos de base (ver Figura 2.8).
- 3.- Ocho pernos para la unión entre la carcasa y los discos (ver Figura 2.3).
- 4.- Una base para el péndulo (ver Figura 2.4).
- 5.- Un péndulo (ver Figura 2.5)
- 6.- Dos piñones (ver Figura 2.6).
- 7.- Dos coronas (ver Figura 2.7).

2.2. Modelo matemático del robot rueda

El funcionamiento del robot rueda está basado principalmente en la cambio de centro de masa o eje de inercia del mismo, mediante un péndulo interno de la rueda (ver Figura 2.12). Por lo que se harán las siguientes suposiciones para la obtención de su modelo dinámico:

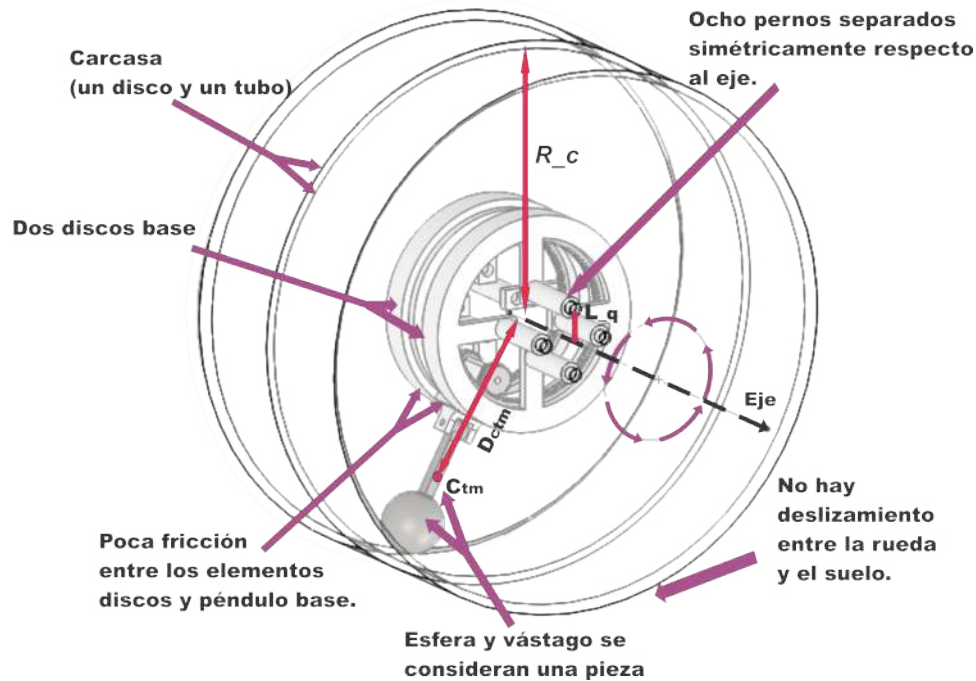


Figura 2.12: Consideraciones del sistema.

- 1.- No hay deslizamiento entre el suelo y la rueda.
- 2.- El péndulo consta de una barra y una esfera, de modo que se consideran como una sola pieza.
- 3.- Se consideran a la carcasa y a los Discos Base como un cilindro hueco y un disco respectivamente.
- 4.- Se considera que la fricción entre los elementos es poca.

Para la modelación del sistema se usará el método de **Euler-Lagrange**, el cual se basa en el cambio de la energía potencial y cinética.

En la modelación se consideran los siguientes elementos del diagrama de cuerpo libre (con base en la Figura 2.13) :

1. Las variables del robot son θ (desplazamiento angular de la rueda), α (desplazamiento angular del motor), f_c (coeficiente de fricción), $\beta = \theta - f_c * \alpha$, $\theta_1 = \theta$, $\theta_2 = \beta$

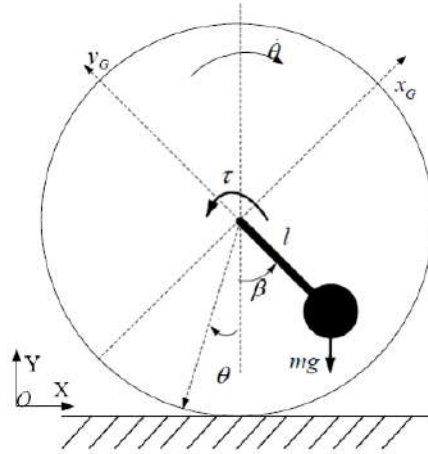


Figura 2.13: Diagrama de cuerpo libre.

2. Las constantes son m_c (masa de carcasa), m_d (masa de los dos discos base), m_{cor} (masa de las dos coronas), m_{per} (masa de un perno), m_p (masa péndulo), I_c (inercia de carcasa), I_d (inercia de los dos discos base), I_{cor} (inercia de las dos coronas), I_{per} (inercia de un perno), I_p (inercia péndulo), L_t (distancia del centro de la rueda al centro de inercia del péndulo), L_q (distancia del centro de la rueda al centro de un perno) y R_c (radio de la carcasa).

Una vez que se tienen esas consideraciones es posible calcular la energía cinética y potencial de cada elemento, esto se mostrará en las siguientes subsecciones.

2.2.1. Energías cinéticas y potenciales

Energía cinética y potencial de la carcasa

Es notorio que la carcasa, al tratarse del elemento que rodará en el suelo, tiene tanto desplazamiento lineal como rotacional, es decir, energía tanto cinética traslacional, rotacional y potencial.

El centro de masa de la carcasa coincide con el centro de la circunferencia, de modo que, el eje de rotación coincide con el eje que pasa por el centro de masa, por lo que su energía cinética rotacional está dada por la ecuación (2.2). En cuanto a la energía traslacional se tiene que la carcasa se desplazará en función de la velocidad angular dada por la ecuación (2.1), de modo que, la energía cinética traslacional está dada por la ecuación (2.3) y la energía potencial está dada por la altura de su centro de masa que es prácticamente el radio de la carcasa, lo cual se establece en la ecuación (2.4).

$$\dot{X} = R_c * \dot{\theta} \quad (2.1)$$

Donde: \dot{X} es el velocidad lineal de la rueda, $\dot{\theta}$ es la velocidad angular de la rueda y R_c es el radio de la carcasa.

$$T_{1R} = \frac{1}{2} m_c * \dot{\theta}^2 \quad (2.2)$$

$$T_{1T} = \frac{1}{2} m_c * R_c^2 * \dot{\theta}^2 \quad (2.3)$$

$$k_1 = m_c * g * R_c \quad (2.4)$$

Energía cinética y potencial de los Discos Base

Los Discos Base tienen energía tanto translacional y rotacional, así como también energía potencial. Los dos Discos Base son concéntricos con la carcasa de la rueda, por otra parte, la altura de un Disco Base está dada por la diferencia entre los radios del Disco Base y la carcasa. Ahora bien, con base en todo a lo anterior, la energía cinética rotacional está dada por la ecuación (2.5). En cuanto a la energía cinética translacional se tiene que los discos se desplazarán translacionalmente en función del ángulo θ de modo que, se hace uso de la relación (2.1) para obtener la energía cinética translacional en función de $\dot{\theta}$, definida por la ecuación (2.6) y por otra parte la energía potencial está dada por la ecuación (2.7).

$$T_{2R} = \frac{1}{2} m_d * \dot{\theta}^2 \quad (2.5)$$

$$T_{2T} = \frac{1}{2} m_d * R_c^2 * \dot{\theta}^2 \quad (2.6)$$

$$k_2 = m_d * g * (R_c - R_d) \quad (2.7)$$

Energía cinética y potencial de las Coronas

Las coronas del mecanismo interno del robot tienen la característica de ser concéntricas con la carcasa. Siendo esto así, se tiene que las coronas tienen energía cinética rotacional, translacional y potencial.

La energía cinética rotacional está dada por la ecuación (2.8). Y en cuanto a la energía cinética translacional se tiene que está en función de la velocidad lineal y que a su vez está relacionada con la relación \dot{X} y $\dot{\theta}$ (ecuación (2.1)), por lo que la energía cinética está definida por la ecuación (2.9), en el caso de la energía potencial se tiene que la altura que tiene la corona está definida por la diferencia entre los radios de la corona y el radio de la carcasa, por lo que, la ecuación que define la energía potencial es la (2.10).

$$T_{3R} = \frac{1}{2} m_{cor} * \dot{\theta}^2 \quad (2.8)$$

$$T_{3T} = \frac{1}{2} m_{cor} * R_c^2 * \dot{\theta}^2 \quad (2.9)$$

$$k_3 = m_c * g * (R_c - R_{cor}) \quad (2.10)$$

Energía cinética y potencial de los pernos

El robot rueda cuenta en total con 8 pernos, 4 de cada lado del robot distribuidos simétricamente respecto al eje de rotación de modo que se hará el siguiente análisis basado en la Figura 2.14, donde L_q es la distancia del centro del perno al eje de rotación del Robot Rueda. De la Figura 2.14 se puede apreciar que cada perno tiene tanto energía cinética rotacional como traslacional, de modo que la energía cinética total está dada por la ecuación (2.11).

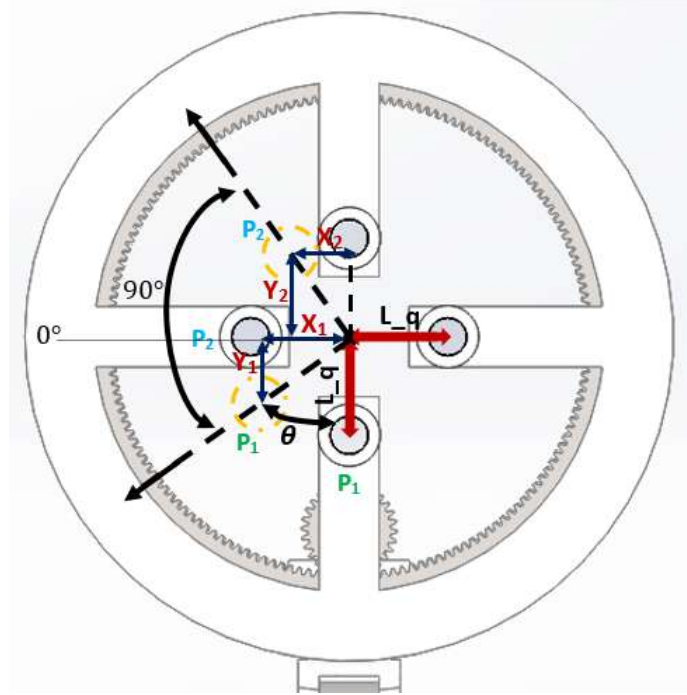


Figura 2.14: Distribución de pernos.

$$\begin{aligned}
 T_{4RotaTras} = & \frac{1}{2}m_{per1} * \dot{x}_1^2 + \frac{1}{2}m_{per1} * \dot{y}_1^2 + \frac{1}{2}m_{per2} * \dot{x}_2^2 + \frac{1}{2}m_{per2} * \dot{y}_2^2 \\
 & + \frac{1}{2}m_{per3} * \dot{x}_3^2 + \frac{1}{2}m_{per3} * \dot{y}_3^2 + \frac{1}{2}m_{per4} * \dot{x}_4^2 + \frac{1}{2}m_{per4} * \dot{y}_4^2
 \end{aligned} \quad (2.11)$$

Ahora bien, para poder sumar las energías dadas por la ecuación (2.11), es necesario poner las coordenadas \mathbf{X} y \mathbf{Y} en los mismos términos o variables, que en este caso es θ .

Donde las coordenadas de los pernos se definen de la siguiente manera:

$$X_1 = L_q * \cos(\theta) + R_c * \theta$$

$$Y_1 = L_q * (1 + \sen(\theta))$$

$$X_2 = L_q * \cos(\theta + 90^\circ) + R_c * \theta$$

$$Y_2 = L_q * (1 + \sen(\theta + 90^\circ))$$

$$X_3 = L_q * \cos(\theta + 180^\circ) + R_c * \theta$$

$$Y_3 = L_q * (1 + \text{sen}(\theta + 180^\circ))$$

$$X_4 = L_q * \cos(\theta + 270^\circ) + R_c * \theta$$

$$Y_4 = L_q * (1 + \text{sen}(\theta + 270^\circ))$$

Las correspondientes derivadas y términos cuadráticos se muestran a continuación:

$$\dot{X}_1 = -L_q * \text{sen}(\theta)\dot{\theta} + R_c * \dot{\theta}$$

$$\dot{Y}_1 = L_q * \cos(\theta) * \dot{\theta}$$

$$\dot{X}_1^2 = L_q * \text{sen}^2(\theta) * \dot{\theta}^2 - 2 * L_q * \text{sen}(\theta) * R_c * \dot{\theta}^2 + R_c^2 * \dot{\theta}^2$$

$$\dot{Y}_1^2 = L_q * \cos^2(\theta) * \dot{\theta}^2$$

$$\dot{X}_2 = -L_q * \text{sen}(\theta + 90^\circ)\dot{\theta} + R_c * \dot{\theta}$$

$$\dot{Y}_2 = L_q * \cos(\theta + 90^\circ) * \dot{\theta}$$

$$\dot{X}_2^2 = L_q * \text{sen}^2(\theta + 90^\circ) * \dot{\theta}^2 - 2 * L_q * \text{sen}(\theta + 90^\circ) * R_c * \dot{\theta}^2 + R_c^2 * \dot{\theta}^2$$

$$\dot{Y}_2^2 = L_q * \cos^2(\theta + 90^\circ) * \dot{\theta}^2$$

$$\dot{X}_3 = -L_q * \text{sen}(\theta + 180^\circ)\dot{\theta} + R_c * \dot{\theta}$$

$$\dot{Y}_3 = L_q * \cos(\theta + 180^\circ) * \dot{\theta}$$

$$\dot{X}_3^2 = L_q * \text{sen}^2(\theta + 180^\circ) * \dot{\theta}^2 - 2 * L_q * \text{sen}(\theta + 180^\circ) * R_c * \dot{\theta}^2 + R_c^2 * \dot{\theta}^2$$

$$\dot{Y}_3^2 = L_q * \cos^2(\theta + 180^\circ) * \dot{\theta}^2$$

$$\dot{X}_4 = -L_q * \text{sen}(\theta + 270^\circ)\dot{\theta} + R_c * \dot{\theta}$$

$$\dot{Y}_4 = L_q * \cos(\theta + 270^\circ) * \dot{\theta}$$

$$\dot{X}_4^2 = L_q * \text{sen}^2(\theta + 270^\circ) * \dot{\theta}^2 - 2 * L_q * \text{sen}(\theta + 270^\circ) * R_c * \dot{\theta}^2 + R_c^2 * \dot{\theta}^2$$

$$\dot{Y}_4^2 = L_q * \cos^2(\theta + 270^\circ) * \dot{\theta}^2$$

Por las equivalencias anteriores es importante notar que el desplazamiento horizontal del perno está dado por el desplazamiento de la rueda y el movimiento interno del perno que va de $\pm L_q$, es por ello que se suman los desplazamientos para obtener el desplazamiento neto.

Sustituyendo las expresiones anteriores a sus correspondientes variables de la ecuación (2.11) se obtiene que la energía cinética de los cuatro pernos está dada por la ecuación (2.12).

$$\begin{aligned}
T_{4RotaTras} = & \frac{1}{2}m_{per1} * L_q^2\dot{\theta}^2 - m_{per1} * L_q * R_c * \text{sen}(\theta)\dot{\theta}^2 + \frac{1}{2}m_{per1} * R_c^2\dot{\theta}^2 + \frac{1}{2}m_{per2} * L_q^2\dot{\theta}^2 \\
& - m_{per2} * L_q * R_c * \text{sen}(\theta + 90^\circ)\dot{\theta}^2 + \frac{1}{2}m_{per2} * R_c^2\dot{\theta}^2 + \frac{1}{2}m_{per3} * L_q^2\dot{\theta}^2 \\
& - m_{per3} * L_q * R_c * \text{sen}(\theta + 180^\circ)\dot{\theta}^2 + \frac{1}{2}m_{per3} * R_c^2\dot{\theta}^2 + \frac{1}{2}m_{per4} * L_q^2\dot{\theta}^2 \\
& - m_{per4} * L_q * R_c * \text{sen}(\theta + 270^\circ)\dot{\theta}^2 + \frac{1}{2}m_{per4} * R_c^2\dot{\theta}^2
\end{aligned} \tag{2.12}$$

Para simplificar la expresión anterior se tiene que cada perno tiene la misma masa, y por identidades trigonométricas en suma de ángulos en senos, se procede a eliminar los términos que involucran senos, quedando la expresión de la energía cinética reducida por la ecuación (2.13)

$$T_{4RotaTras} = \underbrace{\frac{1}{2}(4 * m_{per}) * L_q^2 * \dot{\theta}^2}_{1.-Rotacional} + \underbrace{\frac{1}{2}(4 * m_{per}) * R_c^2\dot{\theta}^2}_{2.-Traslacional} \tag{2.13}$$

Es de importancia señalar que durante el desarrollo de la ecuación (2.12) a la ecuación (2.13) cada perno fue considerado como carga puntual, de modo que, se debe hacer un ajuste en la ecuación (2.13), el primer término de la ecuación (2.13) es simplemente el segundo término del teorema de ejes paralelos (ver ecuación (2.14)) respecto al eje de rotación de la rueda.

$$I_{neta} = \underbrace{I_{per}}_{Inercia_{CM}} + \underbrace{m_{per} * (L_q^2)}_{Inercia_{EjeParalelo}} \tag{2.14}$$

donde, L_q es la distancia del eje del perno al eje de rotación de la rueda.

$$T_{inercial} = \frac{1}{2} * I_{per} * \dot{\theta}^2 \tag{2.15}$$

Por todo lo analizado anteriormente, se tiene que agregar la energía producida por la inercia, la cual se muestra en la ecuación (2.15) y debido a que son cuatro pernos, se considera cuatro veces la energía producida por la inercia en su eje de rotación, de modo que la energía cinética queda definida por la ecuación (2.16)

$$\begin{aligned}
T_{Enec4pernos} = & \underbrace{\frac{1}{2}(4 * m_{per}) * L_q^2 * \dot{\theta}^2}_{Rotacional_{EjeParalelo}} + \underbrace{\frac{1}{2}(4 * m_{per}) * R_c^2\dot{\theta}^2}_{Ec_{Traslacional}} + \underbrace{\frac{1}{2}(4 * I_{per}) * \dot{\theta}^2}_{Rotacional_{EjePerno}} \\
= & \frac{1}{2}((4 * m_{per}) * L_q^2 + 4 * I_{per} + (4 * m_{per}) * R_c^2) * \dot{\theta}^2 \\
= & \frac{1}{2}(4 * m_{per} * (L_q^2 + R_c^2) + 4 * I_{per}) * \dot{\theta}^2 \\
= & 2(m_{per} * (L_q^2 + R_c^2) + I_{per}) * \dot{\theta}^2
\end{aligned} \tag{2.16}$$

Es importante hacer notar que la ecuación (2.16) solo es para cuatro pernos, de modo que para los ocho pernos la expresión de la energía cinética es el doble y está dada por la ecuación (2.17)

$$T_{EneC8pernos} = 4(m_{per} * (L_q^2 + R_c^2) + I_{per}) * \dot{\theta}^2 \quad (2.17)$$

Para el caso de la energía potencial se tiene que para cada perno está dada por la diferencia de altura, donde la altura está definida en función del ángulo (ver ecuación (2.18)), de modo que por la analogía anterior la energía potencial total de 4 pernos está dada por la ecuación (2.19).

$$H_{altura} = L_q * (1 + \text{sen}(\theta)) \quad (2.18)$$

$$\begin{aligned} K_{4pernos} &= m_{per1} * g * L_q * (1 + \text{sen}(\theta)) + m_{per2} * g * L_q * (1 + \text{sen}(\theta + 90^\circ)) \\ &\quad + m_{per3} * g * L_q * (1 + \text{sen}(\theta + 180^\circ)) + m_{per4} * g * L_q * (1 + \text{sen}(\theta + 270^\circ)) \\ &= 4 * m_{per} * g * L_q \end{aligned} \quad (2.19)$$

Debido a que el robot rueda cuenta con ocho pernos, la energía considerada en la ecuación (2.19) se duplica por lo que la energía potencial de los ocho pernos queda expresada por la ecuación (2.20).

$$K_{8pernos} = 8 * m_{per} * g * L_q \quad (2.20)$$

Energía cinética y potencial del Péndulo Base y Péndulo

El Péndulo del robot rueda en general está compuesto por dos elementos, Péndulo Base y el Péndulo, como se aprecia en la Figura 2.15, por lo que se tiene que buscar el centro de masa del elemento compuesto, así como también la inercia total del elemento compuesto respecto a su centro de masa.

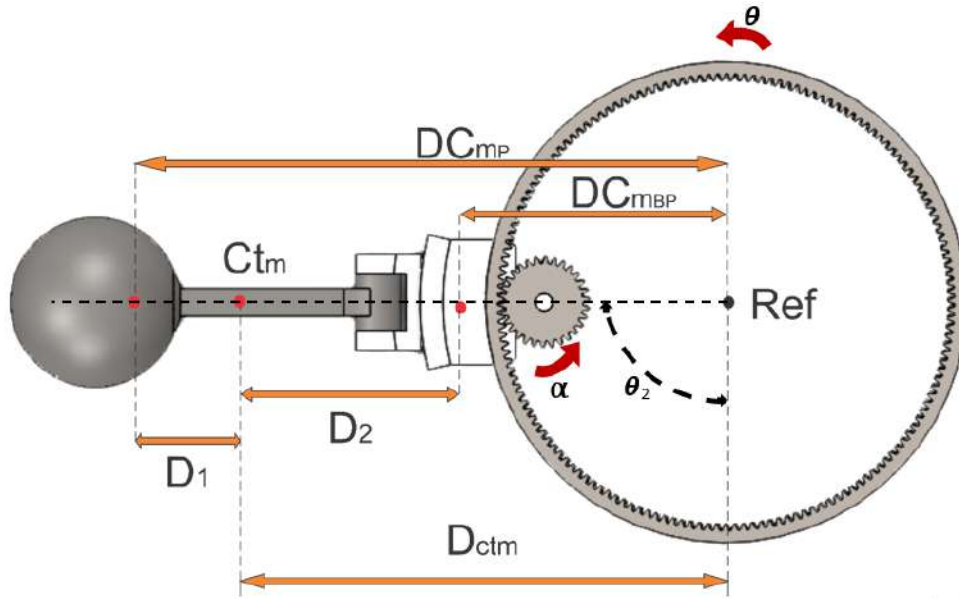


Figura 2.15: Péndulo compuesto.

De la Figura 2.15 se proponen las definiciones como θ es el ángulo de avance de la rueda, α es el ángulo de giro del piñón, θ_2 es el ángulo de inclinación del péndulo, $DCmp$ es la distancia del eje de rotación de la Rueda al centro de masa del elemento Péndulo, $DCmbp$ es la distancia del eje de rotación de la Rueda al centro de masa del elemento Base Péndulo, $Dctm$ es la distancia del centro de masa del elemento compuesto (Base Péndulo y Péndulo) al eje de rotación de la Rueda, D_1 es la distancia del centro de masa del Péndulo al centro de masa del elemento compuesto y D_2 es la distancia del centro de masa de la Base Péndulo al centro de masa del elemento compuesto.

El motivo por el que se calcula la inercia respecto al centro de masa es debido a que dicho elemento tiene dos tipos de energía, energía cinética rotacional y traslacional, de modo que en la energía cinética rotacional el elemento compuesto tiene como eje de rotación al eje de rotación de la rueda, por lo que se podría pensar en aplicar el teorema de ejes paralelos (ver ecuación (2.14)). Sin embargo, el teorema de ejes paralelos no se puede aplicar de manera directa, ya que la inercia del sistema está en función de un ángulo que es propiamente θ_2 , el cual puede ser interpretado por la siguiente relación:

$$\theta_2 = \frac{R_{Pinon}}{R_{Corona}} * \alpha - \theta \quad (2.21)$$

Ahora bien, se tiene que la energía cinética tanto rotacional como traslacional está dada por la ecuación (2.22), la cual surge de la analogía de la Figura 2.16.

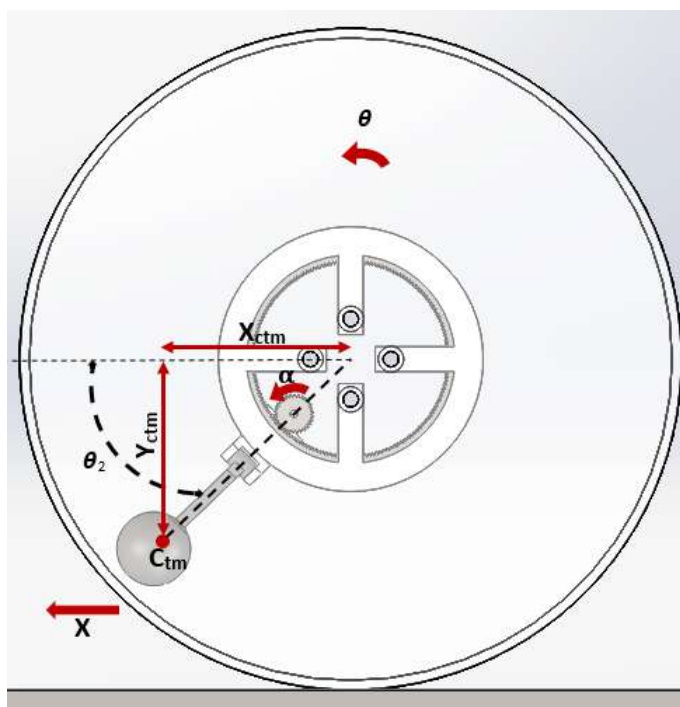


Figura 2.16: Péndulo en coordenadas.

$$\begin{aligned}
T_{ElemCompuesto} &= \frac{1}{2}m_{comp} * \dot{X}_{cmt} + X^2 + \frac{1}{2}m_{comp} * \dot{Y}_{cmt}^2 + \frac{1}{2}I_{ctm} * \dot{\theta}_2^2 \\
&= \frac{1}{2}m_{comp} * \dot{X}_T^2 + \frac{1}{2}m_{comp} * \dot{Y}_{cmt}^2 + \frac{1}{2}I_{ctm} * \dot{\theta}_2^2
\end{aligned} \tag{2.22}$$

Donde las coordenadas que definen al péndulo son los siguientes de la siguiente manera:

$$X_T = D_{ctm} * \text{sen}(\theta_2) + X$$

$$Y_{ctm} = D_{ctm} * \text{cos}(\theta_2)$$

Sus correspondientes derivadas y términos cuadráticos en coordenadas son las siguientes:

$$\dot{X}_T = \dot{X} + D_{ctm} * \text{cos}(\theta_2) * \dot{\theta}_2$$

$$\dot{Y}_{ctm} = -D_{ctm} * \text{sen}(\theta_2) * \dot{\theta}_2$$

$$\dot{X}_T^2 = \dot{X}^2 + 2 * \dot{X} * D_{ctm} * \text{cos}(\theta_2) * \dot{\theta}_2 + D_{ctm}^2 * \text{cos}^2(\theta_2) * \dot{\theta}_2^2$$

$$\dot{Y}_{ctm}^2 = D_{ctm}^2 * \text{sen}^2(\theta_2) * \dot{\theta}_2^2$$

Ahora bien sustituyendo las expresiones anteriores se obtiene que la energía cinética está dada por la ecuación (2.23).

$$\begin{aligned}
T_{ElemCompuesto} &= \frac{1}{2}m_{comp} * (\dot{X}^2 + 2\dot{X} * D_{ctm}\text{cos}(\theta_2) * \dot{\theta}_2 + D_{ctm}^2 * \text{cos}^2(\theta_2) * \dot{\theta}_2^2) \\
&\quad + \frac{1}{2}m_{comp} * (D_{ctm}^2 * \text{sen}^2(\theta_2) * \dot{\theta}_2^2) + \frac{1}{2}I_{ctm} * \dot{\theta}_2^2 \\
&= \frac{1}{2}m_p\dot{X}^2 + m_{comp} * \dot{X} * D_{ctm} * \text{cos}(\theta_2) * \dot{\theta}_2 + \frac{1}{2}m_{comp} * D_{ctm}^2 * \dot{\theta}_2^2 + \frac{1}{2}I_{ctm}\dot{\theta}_2^2 \\
&= \frac{1}{2}m_{comp}R_c\dot{\theta}^2 + m_{comp}\dot{X} * D_{ctm}\text{cos}(\theta_2) * \dot{\theta}_2 + \frac{1}{2}m_{comp} * D_{ctm}^2 * \dot{\theta}_2^2 + \frac{1}{2}I_{ctm}\dot{\theta}_2^2
\end{aligned} \tag{2.23}$$

Para el caso de la energía potencial se tiene que está dada por la ecuación (2.24).

$$K = m_{comp} * g * D_{ctm} * (1 - \text{cos}(\theta_2)) \tag{2.24}$$

2.2.2. Ecuaciones de Euler - Lagrange

Una vez que se han obtenido las energías cinéticas y potenciales, es posible proceder a modelar con Euler - Lagrange, donde la diferencia de energía produce la aceleración del robot rueda.

La ecuación (2.25) modela la energía cinética total del robot rueda.

$$\begin{aligned}
T_{total} &= \underbrace{\frac{1}{2}I_c * \dot{\theta}^2 + \frac{1}{2}m_c * R_c^2 * \dot{\theta}^2}_{Energía_{Carcasa}} + \underbrace{\frac{1}{2}I_d * \dot{\theta}^2 + \frac{1}{2}m_d * R_c^2 * \dot{\theta}^2}_{Energía_{Discos}} + \underbrace{\frac{1}{2}I_{cor} * \dot{\theta}^2 + \frac{1}{2}m_{cor} * R_c^2 * \dot{\theta}^2}_{Energía_{Coronas}} \\
&+ \underbrace{\frac{1}{2}m_{comp}R_c\dot{\theta}^2 + m_{comp} * R_c * \dot{\theta} * D_{ctm}\cos(\theta_2) * \dot{\theta}_2 + \frac{1}{2}m_{comp} * D_{ctm}^2 * \dot{\theta}_2^2 + \frac{1}{2}I_{ctm}\dot{\theta}_2^2}_{Energías_{PnduloCompuesto}} \\
&+ \underbrace{4(m_{per} * (L_q^2 + R_c^2) + I_{per}) * \dot{\theta}^2}_{Energía_{Pernos}} \\
&= \frac{1}{2} * [m_c * R_c^2 + m_d * R_c^2 + m_{cor} * R_c^2 + 4(m_{per} * (L_q^2 + R_c^2) + I_c + I_d + I_{cor} + 8 * I_{per}) * \dot{\theta}^2 \\
&+ m_{comp} * D_{ctm} * R_c * \cos(\theta_2)\dot{\theta}\dot{\theta}_2 + \frac{1}{2}I_{comp}\dot{\theta}_2^2 + \frac{1}{2} * m_{comp} * D_{ctm}^2 * \dot{\theta}^2 \\
&= \frac{1}{2}[e] * \dot{\theta}_2^2 + m_{comp}D_{ctm}R_c * \cos(\theta_2)\dot{\theta}\dot{\theta}_2 + \frac{1}{2}I_{comp}\dot{\theta}_2^2 + \frac{1}{2} * m_{comp} * D_{ctm}^2 * \dot{\theta}_2^2
\end{aligned} \tag{2.25}$$

Donde:

$$e = m_c * R_c^2 + m_d * R_c^2 + m_{cor} * R_c^2 + 4(m_{per} * (L_q^2 + R_c^2) + I_c + I_d + I_{cor} + 8 * I_{per})$$

En el caso de la energía potencial se tiene que está dada por la ecuación (2.26).

$$K = m_c * g * R_c + m_d * g * (R_c - R_d) + m_c * g * (R_c - R_{cor}) + 8 * m_{per} * g * L_q + m_{comp} * g * D_{ctm} * (1 - \cos(\theta_2)) \tag{2.26}$$

Al tener simplificadas las energías tanto cinética como potencial se puede proceder a calcular el Lagrangiano del sistema.

$$\begin{aligned}
L &= \frac{1}{2}[e] * \dot{\theta}^2 + m_{comp} * D_{ctm} * R_c * \cos(\theta_2)\dot{\theta} * \dot{\theta}_2 + \frac{1}{2}I_{comp} * \dot{\theta}_2^2 + \frac{1}{2} * m_{comp} * D_{ctm}^2 * \dot{\theta}_2^2 \\
&- (m_c * g * R_c + m_d * g * (R_c - R_d) + m_c * g * (R_c - R_{cor}) + 8 * m_{per} * g * L_q \\
&+ m_{comp} * g * D_{ctm} * (1 - \cos(\theta_2)))
\end{aligned} \tag{2.27}$$

Es de importancia notar que las variables involucradas en la ecuación (2.27) son θ y θ_2 , en donde θ_2 es una variable auxiliar que engloba dos variables α y θ como se explicó en la Sección 2.2, θ_2 está representada por la relación dada por la ecuación (2.28). Por otra parte, para que la ecuación (2.27) sea manipulable en cuanto a las variables de entrada y salida es necesario sustituir las variables auxiliares.

$$\begin{aligned}
\theta_2 &= \frac{R_{pinon}}{R_{corona}} * \alpha - \theta \\
&= b * \alpha - \theta
\end{aligned} \tag{2.28}$$

Donde: $b = \frac{R_{pinon}}{R_{corona}}$

Por todo lo anterior, la ecuación (2.27) se puede reescribir como:

$$\begin{aligned}
L = & \frac{1}{2}[e] * \dot{\theta}^2 + m_{comp} * D_{ctm} * R_c * \cos(b * \alpha - \theta) * \dot{\theta} * (b * \dot{\alpha} - \dot{\theta}) + \frac{1}{2}I_{comp} * (b * \dot{\alpha} - \dot{\theta})^2 \\
& + \frac{1}{2} * m_{comp} * D_{ctm}^2 * (b * \dot{\alpha} - \dot{\theta})^2 - (m_c g R_c + m_d g * (R_c - R_d) + m_c g * (R_c - R_{cor})) \\
& + 8m_{per} g * L_q + m_{comp} g * D_{ctm} * (1 - \cos(b * \alpha - \theta))
\end{aligned} \tag{2.29}$$

Al tener la diferencia entre la energía cinética y potencial, es posible aplicar la ecuación (2.30) para lo cual, es necesario asignar las variables de estado.

$$\frac{1}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_1} \right) - \frac{\partial L}{\partial q_1} = 0 \tag{2.30}$$

Dichas variables definidas son:

$$\theta = q_1, \dot{\theta} = \dot{q}_1, \alpha = q_2, \dot{\alpha} = \dot{q}_2$$

Una vez asignadas las variables de estado es posible desarrollar las ecuaciones que rigen la dinámica del robot rueda.

$$\begin{aligned}
\frac{1}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_1} \right) = & e\ddot{\theta} - m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta})^2 + m_{comp} D_{ctm} R_c \cos(b * \alpha - \theta) (b\ddot{\alpha} - \ddot{\theta}) \\
& + m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta}) \dot{\theta} - m_{comp} D_{ctm} R_c \cos(b * \alpha - \theta) \dot{\theta}^2 \\
& - I_{comp} * (b * \ddot{\alpha} - \ddot{\theta}) - m_{comp} * D_{comp}^2 * (b * \ddot{\alpha} - \ddot{\theta})
\end{aligned} \tag{2.31}$$

$$\frac{\partial L}{\partial q_1} = m_{comp} * D_{comp} * R_c * \text{sen}(b * \alpha - \theta) * (b * \dot{\alpha} - \dot{\theta}) * \dot{\theta} + m_{comp} * g * D_{comp} * \text{sen}(b * \alpha - \theta) \tag{2.32}$$

$$\frac{1}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_1} \right) - \frac{\partial L}{\partial q_1} = 0$$

$$\begin{aligned}
& e\ddot{\theta} - m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta})^2 + m_{comp} D_{ctm} R_c \cos(b * \alpha - \theta) (b\ddot{\alpha} - \ddot{\theta}) \\
& + m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta}) \dot{\theta} - m_{comp} D_{ctm} R_c \cos(b * \alpha - \theta) \dot{\theta}^2 - I_{comp} (b * \ddot{\alpha} - \ddot{\theta}) \\
& - m_{comp} * D_{comp}^2 * (b * \ddot{\alpha} - \ddot{\theta}) - (m_{comp} * D_{comp} * R_c * \text{sen}(b * \alpha - \theta) * (b * \dot{\alpha} - \dot{\theta}) * \dot{\theta} \\
& + m_{comp} * g * D_{comp} * \text{sen}(b * \alpha - \theta)) = 0
\end{aligned}$$

$$\begin{aligned}
& [e - 2m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) + I_{comp} + m_{comp} * D_{ctm}^2] \ddot{\theta} \\
& + [m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2] b \ddot{\alpha} \\
& - m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta})^2 - m_{comp} * g * D_{comp} \text{sen}(b * \alpha - \theta) = 0
\end{aligned} \tag{2.33}$$

$$\begin{aligned} \frac{1}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_2} \right) &= -m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta}) \dot{\theta} * b + m_{comp} D_{ctm} R_c \text{cos}(b * \alpha - \theta) \ddot{\theta} b \\ &\quad + I_{comp} (b \ddot{\alpha} - \ddot{\theta}) b + m_{comp} * D_{ctm}^2 (b * \ddot{\alpha} - \ddot{\theta}) b \end{aligned} \quad (2.34)$$

$$\frac{\partial L}{\partial q_2} = -m_{comp} * D_{ctm} * R_c \text{sen}(b * \alpha - \theta) (b \dot{\alpha} - \dot{\theta}) b \theta - m_{comp} * g * D_{comp} \text{sen}(b * \alpha - \theta) b \quad (2.35)$$

$$\frac{1}{\partial t} \left(\frac{\partial L}{\partial \dot{q}_2} \right) - \frac{\partial L}{\partial q_2} = I_{entrada}$$

$$\begin{aligned} &- m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} - \dot{\theta}) b \dot{\theta} + m_{comp} D_{ctm} R_c \text{cos}(b * \alpha - \theta) \ddot{\theta} b \\ &+ I_{comp} (b \ddot{\alpha} - \ddot{\theta}) b + m_{comp} * D_{ctm}^2 (b * \ddot{\alpha} - \ddot{\theta}) b - (-m_{comp} * D_{ctm} * R_c \text{sen}(b * \alpha - \theta) (b \dot{\alpha} - \dot{\theta}) b \theta \\ &- m_{comp} * g * D_{comp} \text{sen}(b * \alpha - \theta) b) = I_{ParTorsional} \end{aligned}$$

$$\begin{aligned} &[m_{comp} * D_{ctm} * R_c \text{cos}(b * \alpha - \theta) b - I_p * b - m_{comp} * D_{ctm}^2 * b] \ddot{\theta} + [I_{comp} * b + m_{comp} * D_{ctm}^2 * b] b \ddot{\alpha} \\ &+ m_{comp} * g * D_{ctm} \text{sen}(b \alpha - \theta) b = I_{ParTorsional} \end{aligned}$$

$$\begin{aligned} &[m_{comp} * D_{ctm} * R_c \text{cos}(b * \alpha - \theta) - I_p - m_{comp} * D_{ctm}^2] \ddot{\theta} + [I_{comp} + m_{comp} * D_{ctm}^2] b \ddot{\alpha} \\ &+ m_{comp} * g * D_{ctm} \text{sen}(b \alpha - \theta) = \frac{I_{ParTorsional}}{b} \end{aligned} \quad (2.36)$$

Al obtener las ecuaciones que modelan el comportamiento del sistema es posible despejar las variables de mayor orden para poder simular en el software (MATLAB), así como también agregar un coeficiente de fricción entre la rueda y el suelo la cual está dada por la relación $Fr = c * \dot{\theta}$ en la ecuación (2.33), de modo que las ecuaciones ya despejadas y modificadas quedan dadas por las ecuaciones.

Ecuación 1

$$\begin{aligned} \ddot{\theta} &= [m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta) (b * \dot{\alpha} + \dot{\theta})^2 + m_{comp} * g * D_{comp} \text{sen}(b * \alpha - \theta) \\ &\quad - [m_{comp} * D_{ctm} * R_c \text{cos}(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2] b \ddot{\alpha} - \underbrace{c * \dot{\theta}}_{\text{Fricción suelo}}] \frac{1}{W} \end{aligned} \quad (2.37)$$

Donde: $W = e - 2m_{comp} * D_{ctm} * R_c \text{cos}(b * \alpha - \theta) + I_{comp} + m_{comp} * D_{ctm}^2] \ddot{\theta}$ y $c =$ Coeficiente de fricción.

Ecuación 2

$$\begin{aligned} \ddot{\alpha} &= \left[\frac{I_{ParTorsional}}{b} - [m_{comp} D_{ctm} R_c \text{cos}(b \alpha - \theta) - I_p - m_{comp} D_{ctm}^2] \ddot{\theta} - m_{comp} g D_{ctm} \text{sen}(b \alpha - \theta) \right] \\ &\quad * \frac{1}{[I_{comp} + m_{comp} D_{ctm}^2] b} \end{aligned} \quad (2.38)$$

2.2.3. Análisis y validación del modelo dinámico

Una vez que se tienen las ecuaciones que modelan el comportamiento del sistema, es de importancia validarlas, es por ello que se recurre a utilizar nuevamente el software de diseño (SolidWorks) donde a cada elemento que compone al robot se le asignan las propiedades del material, en donde se tiene que la Carcasa es de acrílico, el Péndulo es acero, y los demás componentes son de PLA (material para la impresión 3D), por otra parte, se asigna la fricción entre los componentes internos del robot.

Ahora bien, una vez definidas las propiedades de los elementos del Robot Rueda, se procede a asignar en el estudio de movimiento en SolidWorks (ver Figura 2.17) las propiedades del robot y su entorno, las cuales son:

1. La gravedad.
2. La fricción entre la rueda y el suelo (establecidas por el software).
3. La fricción viscosa entre el aire y los lados del robot rueda (establecidas por requerimientos).

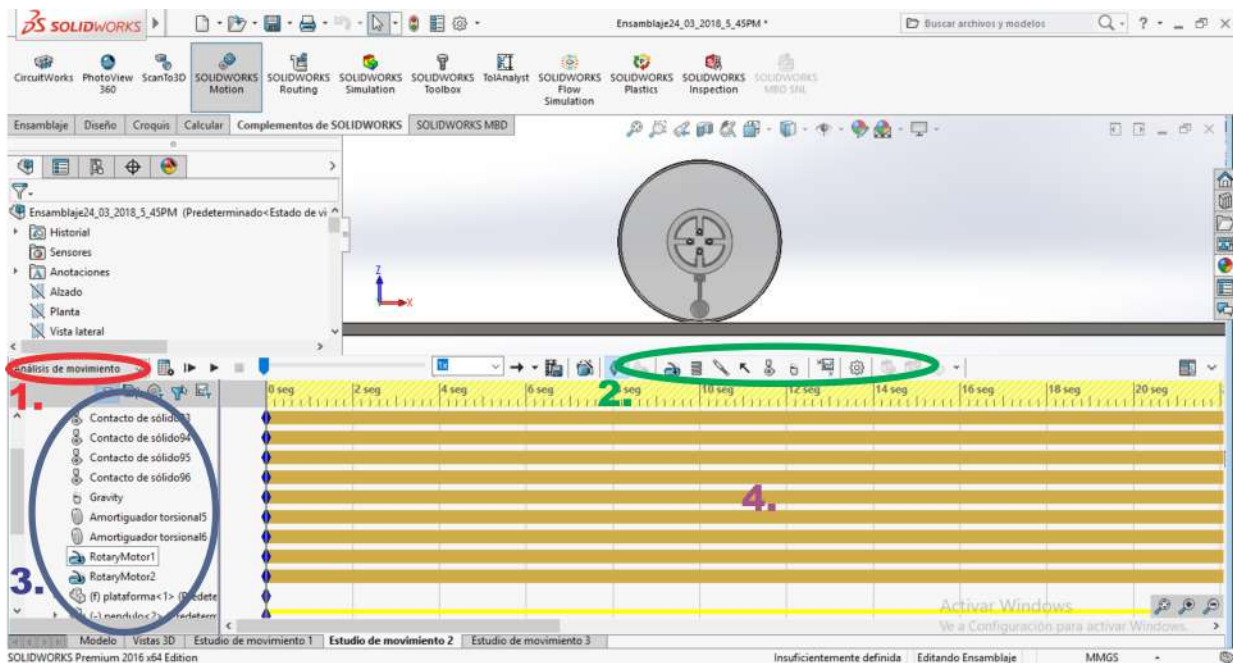


Figura 2.17: Estudio de movimiento en SolidWorks 1. Análisis de movimiento 2. Herramientas de propiedades 3. Características del Robot (Gravedad, fricción, sensores y actuadores) 4. Tiempo de simulación

Es de importancia señalar que la fricción entre el suelo y la rueda son establecidas por el software de acuerdo al material. Por otra parte, la fricción viscosa entre el aire y la rueda dependen de $\dot{\theta}$, en donde dicho valor de fricción es lineal y se propone por los requerimientos. Los requerimientos que se usan para que definir el coeficiente la fricción viscosa son los siguientes:

1. – El robot rueda alcance una velocidad de establecimiento de $49^\circ/\text{s}$.
2. – La velocidad de entrada a los motores es de 43.5 RPM

Por los requerimientos anteriores se tiene que el robot deberá tener una fricción viscosa en las caras de $0.10 \text{ N}\cdot\text{mm}/(^{\circ}/\text{s})$, esto se consigue mediante iteraciones proponiendo un coeficiente de fricción y aumentándolo o decreméntándolo según se requiera.

Una vez establecidos los materiales de cada elemento del robot, la gravedad, los coeficientes de fricción y el tiempo de simulación, la velocidad que adquiere el robot está dada por la gráfica de la Figura 2.18.

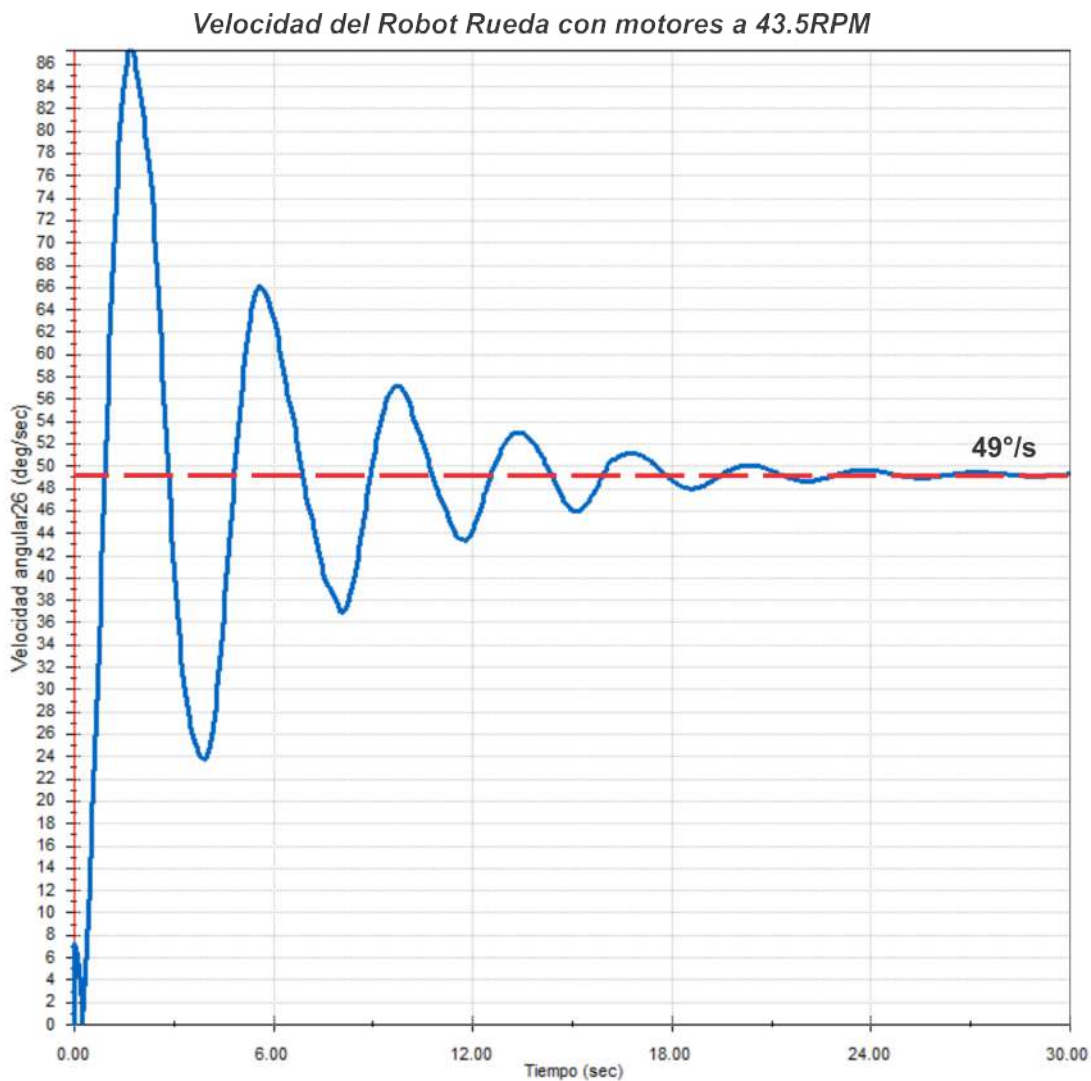


Figura 2.18: Velocidad del Robot Rueda ($^\circ/\text{s}$)

Es de importancia observar de la Figura 2.18 que el robot rueda presenta oscilaciones en la velocidad ante una entrada de velocidad constante en los motores, y que propiamente después se estabiliza debido a la fricción viscosa en $49^\circ/\text{s}$, esto desde el punto de vista cinemático, donde

solo se consideran velocidades. Sin embargo, no se considera la magnitud del par torsional requerido en los motores para que tengan la velocidad de 43.5RPM en $t=0$ (ver Figura 2.19) y que además permanezca constante en $t>0$, siendo así se procede a medir mediante software SolidWorks el par torsional requerido en los motores (ver Figura 2.20).

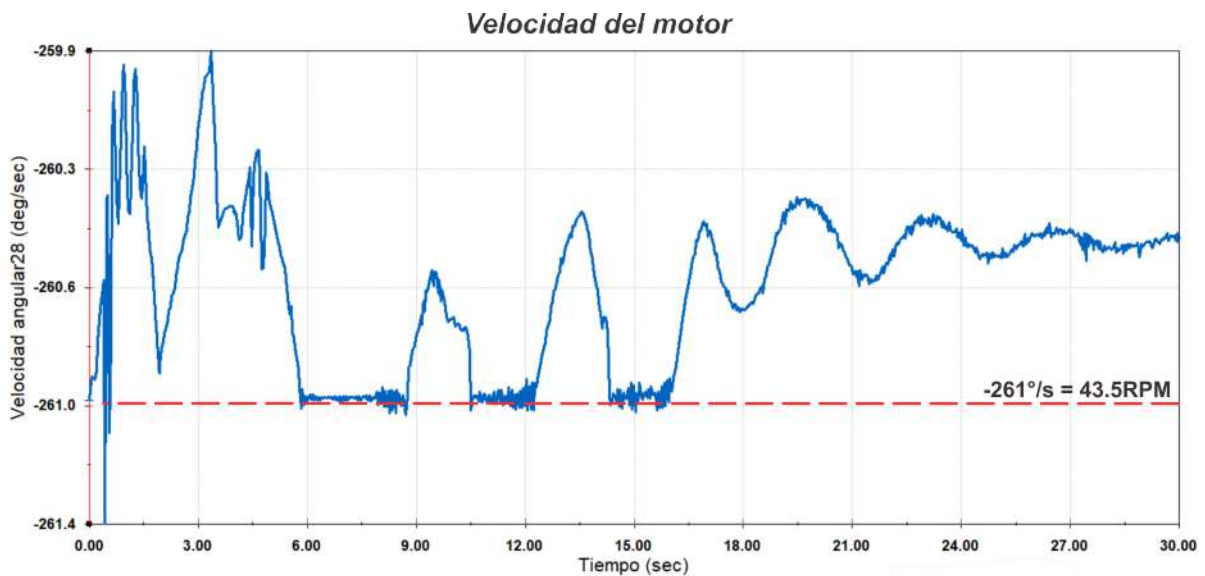


Figura 2.19: Velocidad de un motor.

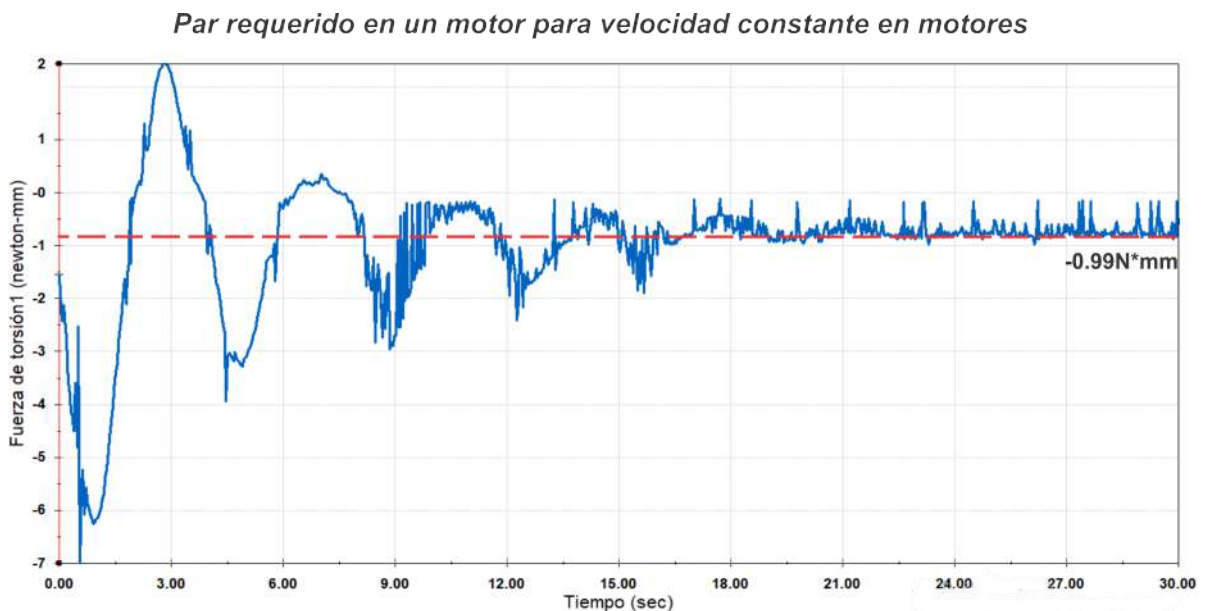


Figura 2.20: Par requerido en un motor para mantener velocidad constante.

Se puede notar de la Figura 2.20 que el par requerido solo es para un motor, por lo que, el par que se requiere para que los motores tengan una velocidad constante de 43.5RPM es de

1.98N*mm, distribuidos en 0.99N*mm por motor.

Ahora bien, como se puede apreciar en la Figura 2.20 el par requerido es 5 veces mayor para mantener la velocidad constante desde $t=0$, y conforme pasa el tiempo el par decremента hasta un 1N*mm aproximadamente. Además, la velocidad que adquiere la rueda presenta comportamiento subamortiguado, de modo que es algo indeseado para su comportamiento.

Al tener en cuenta que el análisis cinemático (la velocidad de los motores son la entrada de control en lazo abierto del robot) no es lo ideal para el robot, ya que la velocidad oscila, se realiza un análisis dinámico, donde ahora la entrada de control en lazo abierto es el par torsional en los motores, por lo que, el par que se elegirá como entrada será el de establecimiento de velocidad constante 1N*mm para cada motor. Siendo esto así, se procede a cambiar la entrada de velocidad a par torsional, por lo que los resultados son mostrados por la Figura 2.21.

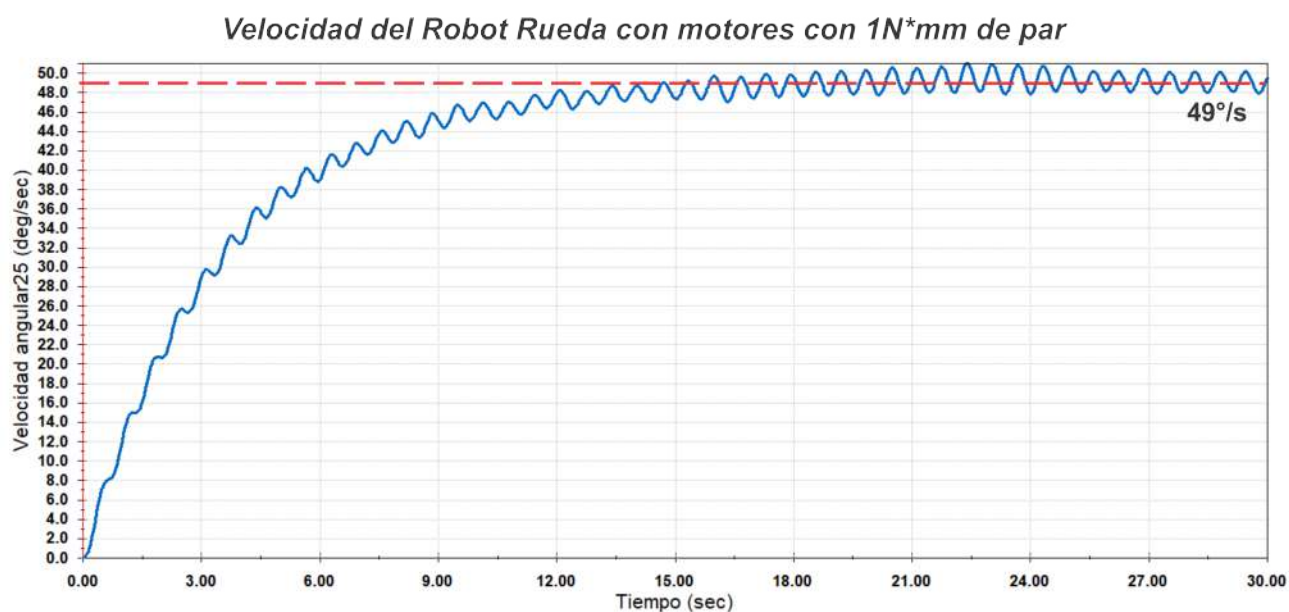


Figura 2.21: Velocidad del Robot Rueda ante una entrada de par torsional.

De la Figura 2.21 se tiene que al aplicar un par torsional constante al robot, éste se comporta como un sistema amortiguado en donde la velocidad no oscila como en el caso del análisis cinemático. Sin embargo, no hay gran cambio en cuanto al tiempo de asentamiento. Por lo que, resulta ser mas ventajoso aplicar control en lazo cerrado mediante la variación de par torsional de los motores.

Una vez que se ha observado que el análisis dinámico es el más conveniente, se procede a validar las ecuaciones que rigen la dinámica del robot, de modo que se hace uso de la herramienta Matlab **Simulink** en donde mediante bloques se resuelven las ecuaciones de la dinámica del robot (ver ecuaciones (2.39) y (2.40)).

$$\ddot{\theta} = \underbrace{[m_{comp} D_{ctm} R_c \text{sen}(b * \alpha - \theta)(b * \dot{\alpha} + \dot{\theta})^2]}_{\text{PrimerTermino}} + \underbrace{m_{comp} * g * D_{comp} \text{sen}(b * \alpha - \theta)}_{\text{SegundoTermino}} - \underbrace{[m_{comp} * D_{ctm} * R_c \text{cos}(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2]}_{\text{TercerTermino}} b \ddot{\alpha} - \underbrace{c * \dot{\theta}}_{\text{CuartoTermino}} \Big] \frac{1}{W} \quad (2.39)$$

$$\ddot{\alpha} = \left[\underbrace{\frac{I_{ParTorsional}}{b}}_{\text{QuintoTermino}} - \underbrace{[m_{comp} D_{ctm} R_c \text{cos}(b\alpha - \theta) - I_p - m_{comp} D_{ctm}^2]}_{\text{SextoTermino}} \ddot{\theta} - \underbrace{m_{comp} g D_{ctm} \text{sen}(b\alpha - \theta)}_{\text{SeptimoTermino}} \right] \frac{1}{[I_{comp} + m_{comp} D_{ctm}^2] b} \quad (2.40)$$

Al tener construido el diagrama a bloques en Matlab se procede a sustituir los valores de las propiedades de los materiales como masas, inercias y distancias, dichas propiedades se muestran en la siguiente tabla:

| Elemento | Masa(gr) | Inercia _{cm} (gr * cm ²) | Radio o D _{cm} |
|-------------------------|----------|---|-------------------------|
| Caracasa | 583.44 | 73884.32 | 13.5 |
| Disco Base | 70.42 | 1548.73 | N.A |
| Perno | 1.87 | 0.35 | 1.64 |
| Pendulo _{Comp} | 122.36 | 586.611 | 10.34411 |
| Piñon | N.A | N.A | 0.75 |
| Corona | 4.199 | 7112.644 | 3.975 |

Teniendo esto en cuenta se es posible simular el diagrama a bloques del sistema, en donde el problema es encontrar el coeficiente de fricción que realmente tiene el sistema, que es el cuarto término de la ecuación (2.39), de modo que hay que iterar la simulación ajustando el coeficiente de fricción de modo que la velocidad del sistema se estabilice en 49°/s al tener como entrada 2N*mm, por lo que se llega a que el coeficiente de fricción adecuado es 0.124. El resultado de la iteración queda dado por la Figura 2.22.

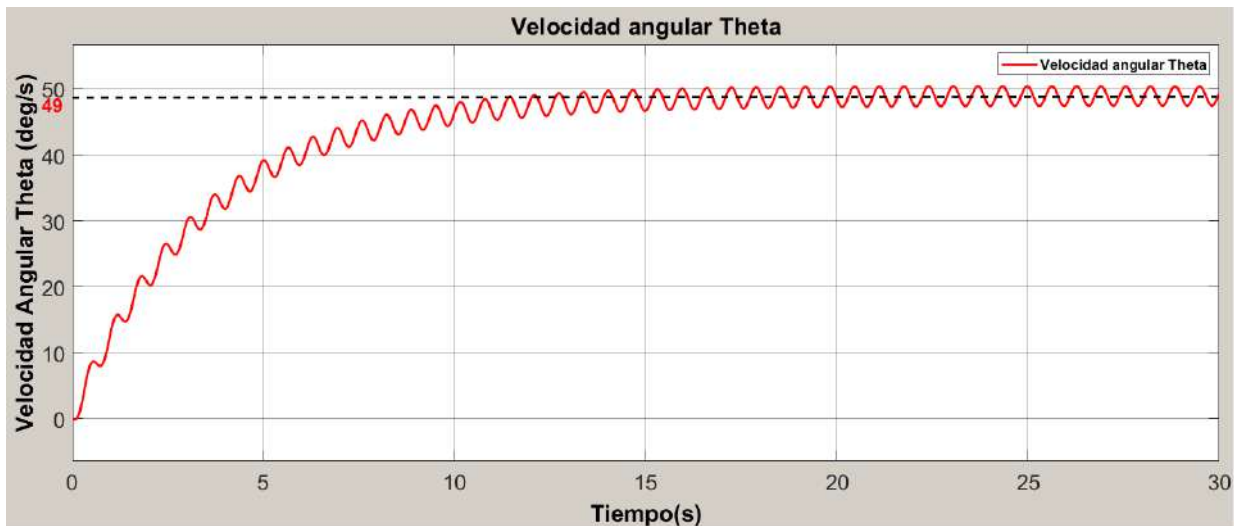


Figura 2.22: Velocidad del robot rueda con simulación en Matlab Simulink.

Para una comparación más cercana, los datos SolidWorks son exportados y graficados en Matlab junto con el resultado de las ecuaciones de la dinámica del robot en Matlab Simulink. Para ello se tienen que exportar los datos de la gráfica de simulación de solidworks mediante un archivo CSV.

Una vez que se tienen los datos es posible comparar las gráficas como se muestra en la Figura 2.23, de modo que se puede observar las gráficas son casi idénticas en cuanto a la frecuencia de oscilación, así como también el tiempo de establecimiento, sin embargo, para poder asegurar aún más que las ecuaciones modelan de manera correcta al robot, se procede a medir otro parámetro que es el desplazamiento lineal del robot. Por lo que las ecuaciones que modelan al sistema dan como resultado la Figura 2.25 que muestra el desplazamiento del robot, y en cuanto a la simulación en SolidWorks se tiene la gráfica dada por la Figura 2.24.

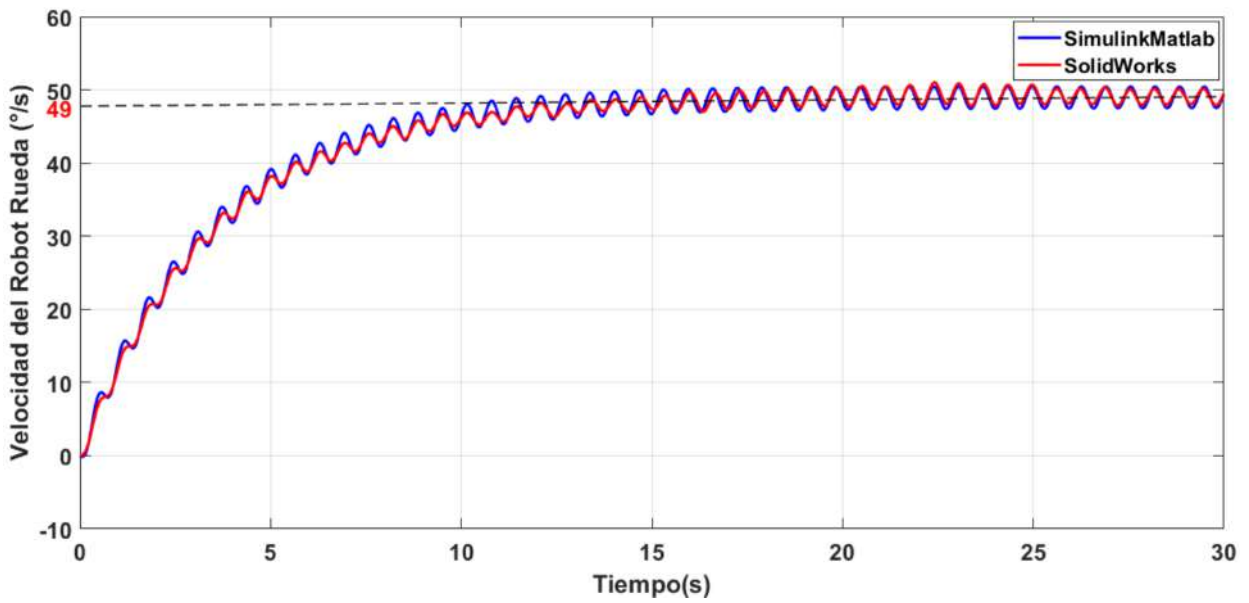


Figura 2.23: Comparación de simulaciones de SolidWorks y Matlab Simulink.

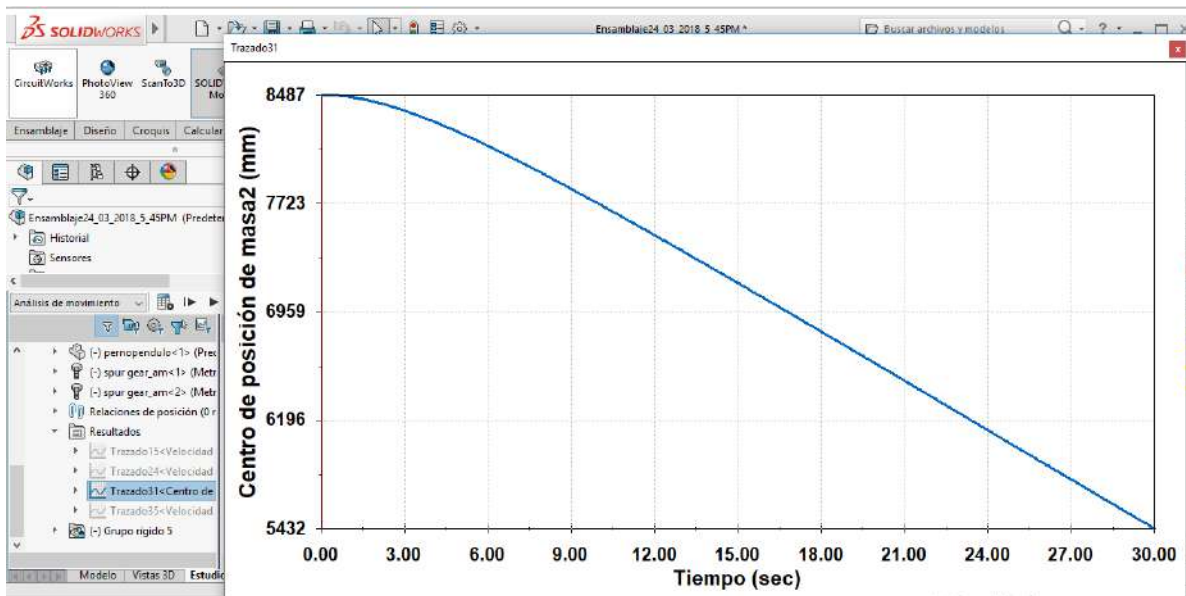


Figura 2.24: Posición del robot mediante la simulación en SolidWorks.

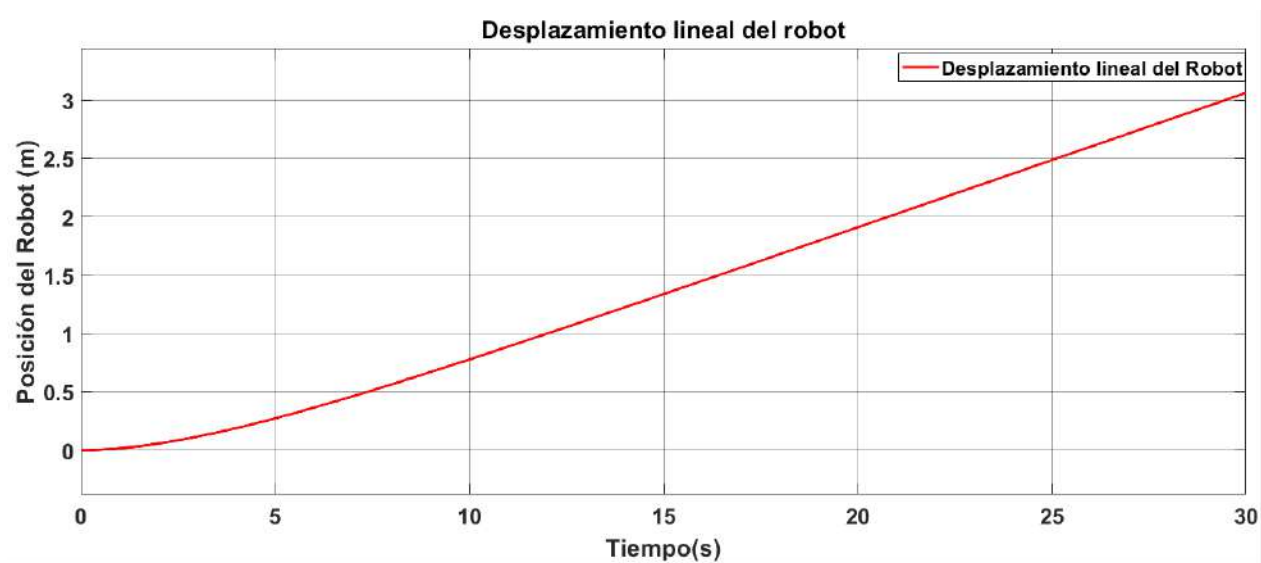


Figura 2.25: Posición del robot con la simulación de las ecuaciones en Matlab Simulink.

Es de importancia notar que en la Figura 2.24 tiene un offset de desplazamiento y esto se debe a que el robot rueda está posicionada a 8487mm de la referencia en SolidWorks, por lo que, para una comparación directa se desfase la gráfica.

Al realizar el desfase y por todo lo anterior expuesto es posible comparar las dos gráficas, como se muestra en la Figura 2.26.

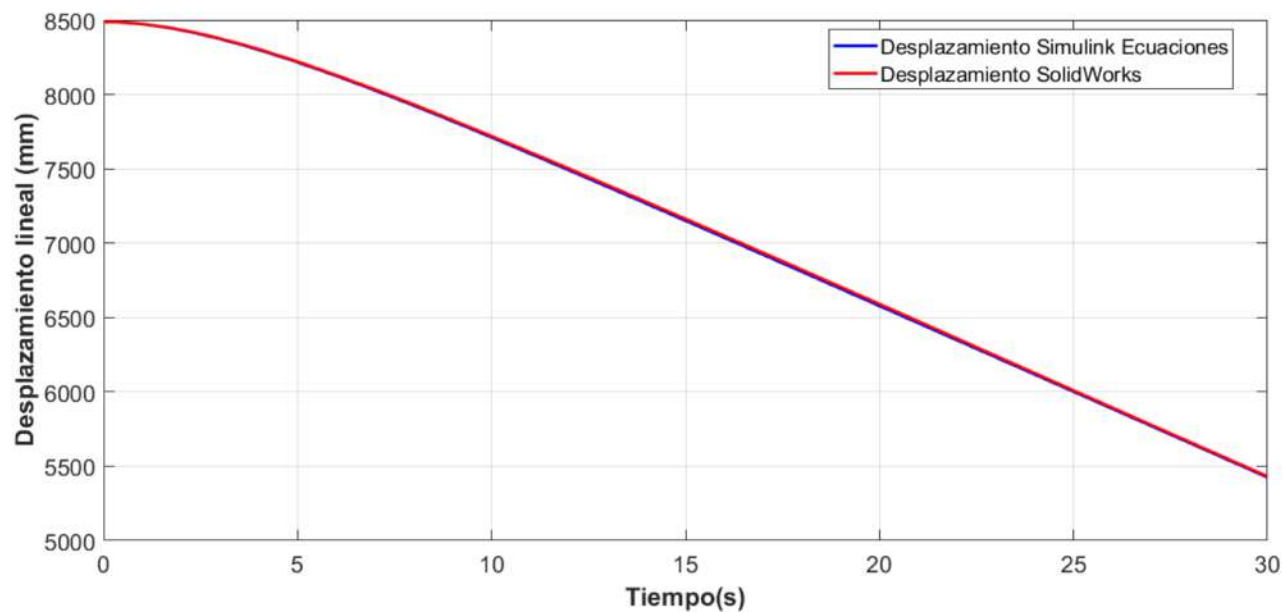


Figura 2.26: Comparación de los desplazamientos tanto de Matlab y SolidWorks.

Por todo lo anterior se puede concluir que el modelo matemático propuesto modela correctamente el comportamiento del robot rueda, ya que las gráficas coinciden a la perfección.

Por lo que, se puede concluir que las consideraciones que se hicieron respecto al robot rueda para calcular las ecuaciones fueron acertadas.

Capítulo 3

Linealización de las ecuaciones dinámicas y control del robot rueda

3.1. Linealización de las ecuaciones de la dinámica del robot rueda

En el capítulo anterior se logró notar, que las ecuaciones que modelan el robot son válidas para todo dominio de θ , sin embargo, como ya se mencionó en dicha sección el análisis dinámico ofrece resultados adecuados en términos de estabilidad para aplicar control en donde la entrada es un par torsional.

Es de importancia notar que las ecuaciones que rigen el comportamiento del sistema son no lineales y que además, las variables de grado superior están acopladas, es decir, no se puede dejar una ecuación con una variable de mayor orden despejada sin que haya otra variable del mismo orden involucrada, siendo esto así se recurre a expresar las dos ecuaciones de la dinámica del robot como una ecuación matricial, en la que las variables de mayor orden quedan como un vector, como se muestra en la ecuación (3.1) que es el modelo ideal del arreglo matricial, por lo que se tienen que identificar y renombrar los términos de las ecuaciones (2.37) y (2.38) que involucran cada término de la matriz o vector de la ecuación (3.1).

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} In_1 \\ In_2 \end{bmatrix} \quad (3.1)$$

Para acoplar las ecuaciones (2.33) y (2.36) en la ecuación matricial (3.1) es necesario hacer un reordenamiento en las variables de la ecuación (2.33) que da como resultado la ecuación

(3.2).

$$\begin{aligned} & [e - 2m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) + I_{comp} + m_{comp} * D_{ctm}^2] \ddot{\theta} \\ & + [m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2] b \ddot{\alpha} \\ & - \underbrace{m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta)}_{V_{Aux}} (b * \dot{\alpha} - \dot{\theta})^2 + F_c \dot{\theta} - m_{comp} * g * D_{comp} \operatorname{sen}(b * \alpha - \theta) = 0 \end{aligned}$$

$$\begin{aligned} & [e - 2m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) + I_{comp} + m_{comp} * D_{ctm}^2] \ddot{\theta} \\ & + [m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2] b \ddot{\alpha} \\ & - m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) (b^2 \dot{\alpha}^2 - 2b \dot{\alpha} \dot{\theta} + \dot{\theta}^2) + F_c \dot{\theta} - m_{comp} * g * D_{comp} \operatorname{sen}(b * \alpha - \theta) = 0 \end{aligned}$$

$$\begin{aligned} & [e - 2m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) + I_{comp} + m_{comp} * D_{ctm}^2] \ddot{\theta} \\ & + [m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2] b \ddot{\alpha} \\ & - m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) (b^2 \dot{\alpha}^2) + m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) (2b \dot{\alpha} \dot{\theta}) \\ & - m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) (\dot{\theta}^2) + F_c \dot{\theta} - m_{comp} * g * D_{comp} \operatorname{sen}(b * \alpha - \theta) = 0 \end{aligned}$$

$$\begin{aligned} & \underbrace{[e - 2m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) + I_{comp} + m_{comp} * D_{ctm}^2]}_{A_{11}} \ddot{\theta} \\ & + \underbrace{[m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) - I_{comp} - m_{comp} * D_{comp}^2]}_{A_{12}} b \ddot{\alpha} \\ & + \underbrace{[-m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) (b^2 \dot{\alpha}^2) + 2m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) \dot{\theta}] b \dot{\alpha}}_{B_{12}} \\ & - \underbrace{[m_{comp} D_{ctm} R_c \operatorname{sen}(b * \alpha - \theta) \dot{\theta}]}_{B_{11}} + \underbrace{F_c \dot{\theta} - m_{comp} * g * D_{comp} \operatorname{sen}(b * \alpha - \theta)}_{C_1} = \underbrace{0}_{I_{n_1}} \end{aligned} \quad (3.2)$$

$$\begin{aligned} & \underbrace{[m_{comp} * D_{ctm} * R_c \cos(b * \alpha - \theta) - I_p - m_{comp} * D_{ctm}^2]}_{A_{21}} \ddot{\theta} + \underbrace{[I_{comp} + m_{comp} * D_{ctm}^2]}_{A_{22}} b \ddot{\alpha} \\ & + \underbrace{m_{comp} * g * D_{ctm} \operatorname{sen}(b \alpha - \theta)}_{C_2} = \underbrace{\frac{I_{ParTorsional}}{b}}_{T_{in}} \end{aligned} \quad (3.3)$$

Una vez que se localizaron los términos correspondientes la ecuación matricial, ésta queda dada por la ecuación (3.4).

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ T_{in} \end{bmatrix} \quad (3.4)$$

Al tener las ecuaciones de la dinámica del sistema en forma matricial, es posible hacer un despeje de las variables de mayor orden, quedando como se muestra en la ecuación (3.5).

$$\begin{aligned}
A \begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} + B \begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + C &= \begin{bmatrix} 0 \\ T_{in} \end{bmatrix} \\
\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} &= A^{-1}(IN - (B \begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + C)) \\
\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} &= \frac{1}{\det_A} \begin{bmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{bmatrix} \left(\begin{bmatrix} 0 \\ T_{in} \end{bmatrix} - \left(\begin{bmatrix} B_{11} & B_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \right) \right) \\
\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} &= \frac{1}{\det_A} \begin{bmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{bmatrix} \begin{bmatrix} -(B_{11}\dot{\theta} + B_{12}\dot{\alpha} + C_1) \\ T_{in} - C_2 \end{bmatrix} \\
\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} &= \frac{1}{\det_A} \begin{bmatrix} -(B_{11}\dot{\theta} + B_{12}\dot{\alpha} + C_1)A_{22} - (T_{in} - C_2)A_{12} \\ (B_{11}\dot{\theta} + B_{12}\dot{\alpha} + C_1)A_{21} + (T_{in} - C_2)A_{11} \end{bmatrix} \\
\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} &= \frac{1}{\det_A} \begin{bmatrix} -(V_{aux}(b\dot{\alpha} - \dot{\theta})^2 + F_c\dot{\theta} + C_1)A_{22} - (T_{in} - C_2)A_{12} \\ (V_{aux}(b\dot{\alpha} - \dot{\theta})^2 + F_c\dot{\theta} + C_1)A_{21} + (T_{in} - C_2)A_{11} \end{bmatrix}
\end{aligned} \tag{3.5}$$

Una vez que se tiene la ecuación matricial en donde las variables de mayor orden están despejadas en forma de vector, de modo que es posible poner cada variable de mayor orden con su correspondiente igualdad, es decir las ecuaciones (3.6) y (3.7).

$$\ddot{\theta} = \frac{1}{\det_A} [-(V_{aux}(b\dot{\alpha} - \dot{\theta})^2 + F_c\dot{\theta} + C_1)A_{22} - (T_{in} - C_2)A_{12}] \tag{3.6}$$

$$\ddot{\alpha} = \frac{1}{\det_A} [(V_{aux}(b\dot{\alpha} - \dot{\theta})^2 + F_c\dot{\theta} + C_1)A_{21} + (T_{in} - C_2)A_{11}] \tag{3.7}$$

Es posible notar que las variables de mayor orden ya están despejadas de modo que, es posible asignar variables de estado al sistema, las cuales son:

$$x_1 = \theta, x_2 = \dot{\theta}, x_3 = \alpha \text{ y } x_4 = \dot{\alpha}.$$

Teniendo en cuenta las variables de estado de las ecuaciones del robot, es posible aplicar la técnica de linealización alrededor de un punto de operación o equilibrio, que en este caso resultan ser lo mismo, y esto es cuando $\theta_2 = b\alpha - \theta = 0^\circ$ y $\dot{\theta}_2 = b\dot{\alpha} - \dot{\theta} = 0^\circ/s$, y esto sucede ante una entrada que da como resultado el equilibrio *Inequilibrio*

Para aplicar esta técnica, las variables de mayor orden deben estar despejadas y que las variables independientes sean de menor orden que la variable despejada, una vez cumpliendo con lo anterior se tiene que hacer lo siguiente:

1. Crear la matriz de funciones con las variables de estado.
2. Derivar cada función de la matriz de funciones con respecto a cada variable de estado.
3. Evaluar el punto de equilibrio, la entrada de equilibrio y los valores de las constantes, todo esto genera la matriz A que se multiplica por el vector de estados.
4. Derivar cada función de la matriz de funciones por la entrada.

5. Evaluar el punto de equilibrio, la entrada de equilibrio y los valores de las constantes, todo esto genera la matriz B que se multiplica por la entrada del sistema.
6. Sumar los resultados del punto 3 y 5 e igualarlos con el vector de razón de cambio respecto al tiempo del vector de estados (ver ecuación (3.8)).

$$\begin{bmatrix} \dot{x}_{1\delta} \\ \dot{x}_{2\delta} \\ \dot{x}_{3\delta} \\ \dot{x}_{4\delta} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{f_1(\alpha,\theta)}{\partial x_1} & \frac{f_1(\alpha,\theta)}{\partial x_2} & \frac{f_1(\alpha,\theta)}{\partial x_3} & \frac{f_1(\alpha,\theta)}{\partial x_4} \\ \frac{f_2(\alpha,\theta)}{\partial x_1} & \frac{f_2(\alpha,\theta)}{\partial x_2} & \frac{f_2(\alpha,\theta)}{\partial x_3} & \frac{f_2(\alpha,\theta)}{\partial x_4} \\ \frac{f_3(\alpha,\theta)}{\partial x_1} & \frac{f_3(\alpha,\theta)}{\partial x_2} & \frac{f_3(\alpha,\theta)}{\partial x_3} & \frac{f_3(\alpha,\theta)}{\partial x_4} \\ \frac{f_4(\alpha,\theta)}{\partial x_1} & \frac{f_4(\alpha,\theta)}{\partial x_2} & \frac{f_4(\alpha,\theta)}{\partial x_3} & \frac{f_4(\alpha,\theta)}{\partial x_4} \end{bmatrix}}_{\text{Matriz}_A} \begin{bmatrix} x_{1\delta} \\ x_{2\delta} \\ x_{3\delta} \\ x_{4\delta} \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{f_1(\alpha,\theta)}{\partial In_2} \\ \frac{f_2(\alpha,\theta)}{\partial In_2} \\ \frac{f_3(\alpha,\theta)}{\partial In_2} \\ \frac{f_4(\alpha,\theta)}{\partial In_2} \end{bmatrix}}_{\text{Matriz}_B} [In_{2\delta}] \quad (3.8)$$

Donde:

$$\begin{aligned} x_\delta &= x(t) - x_{operacion} \\ In_\delta &= In(t) - In_{operacion} \end{aligned}$$

Ahora bien, para saber el estado del robot en algún momento, se tiene que generar una salida, lo cual se realiza de la siguiente manera:

1. Se eligen las variables de salida y se acomodan en forma vectorial y se derivan parcialmente con respecto a las variables de estados, con esto se crea la matriz C.
2. La matriz C, es multiplicada por el vector de estados.
3. La misma matriz de salida es derivada ahora por la entrada, con esto se crea la matriz D.
4. La matriz D, es multiplicada por la entrada.
5. Las salidas Y, están dadas por la suma del resultado del punto 2 y 4. (ver ecuación (3.8))

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{x_1}{\partial x_1} & \frac{x_1}{\partial x_2} & \frac{x_1}{\partial x_3} & \frac{x_1}{\partial x_4} \\ \frac{x_2}{\partial x_1} & \frac{x_2}{\partial x_2} & \frac{x_2}{\partial x_3} & \frac{x_2}{\partial x_4} \\ \frac{x_3}{\partial x_1} & \frac{x_3}{\partial x_2} & \frac{x_3}{\partial x_3} & \frac{x_3}{\partial x_4} \\ \frac{x_4}{\partial x_1} & \frac{x_4}{\partial x_2} & \frac{x_4}{\partial x_3} & \frac{x_4}{\partial x_4} \end{bmatrix}}_{\text{Matriz}_C} \begin{bmatrix} x_{1\delta} \\ x_{2\delta} \\ x_{3\delta} \\ x_{4\delta} \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{x_1}{\partial In} \\ \frac{x_2}{\partial In} \\ \frac{x_3}{\partial In} \\ \frac{x_4}{\partial In} \end{bmatrix}}_{\text{Matriz}_D} [In_\delta] \quad (3.9)$$

Ahora se procede a calcular los puntos de equilibrio y la entrada de equilibrio, en donde se imponen las condiciones de equilibrio y se estipula que el péndulo debe estar siempre verticalmente hacia abajo, es decir $\theta_2 = b * \alpha - \theta = 0$, así como también su velocidad $\dot{\theta}_2 = 0$ y en el caso del sistema en general, la aceleración $\ddot{\theta} = 0$ y en el caso del motor este también debe tener la aceleración $\ddot{\alpha} = 0$, es decir se sustituyen todas esas condiciones en las ecuaciones

(3.6) y (3.7), en donde la entrada de equilibrio $T_{in} = 0$. Teniendo todo lo anterior en cuenta es posible crear la matriz de funciones (ver ecuación (3.10)).

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{\det A} [-(V_{aux}(bx_4 - x_2)^2 + F_c \dot{\theta} + C_1)A_{22} - (T_{in} - C_2)A_{12}] \\ x_4 \\ \frac{1}{\det A} [(V_{aux}(bx_4 - x_2)^2 + F_c \dot{\theta} + C_1)A_{21} + (T_{in} - C_2)A_{11}] \end{bmatrix} \quad (3.10)$$

Teniendo la función matricial (3.10) se pueden calcular las matrices A, B, C y D. De modo que es de importancia señalar que en el caso de la matriz C se establecen todas las variables de estado de x_1 a x_4 . Siendo esto así, se hace uso del Software Matlab, en donde se hace la programación correspondiente para encontrar la matriz A,B,C y D, dicho código de la programación se muestra a continuación:

```

1 clear all
2 clc
3 syms S e mp lt rc b x1 x2 x3 x4 Ip g tau x2_p x4_p
4 g1=9.81
5 % masas
6 M_carcaza=0.58344*2
7 M_perno=1.87e-3
8 M_pendolo=0.12236
9 M_corona=(4.199e-3)*2
10 M_base=(70.42e-3)*2
11 % Inercias
12 I_carcaza=(7.3884e-3)*2
13 I_perno=0.35e-7
14 I_pendolo=5.866111e-5
15 I_corona=(7.112644e-4)*2
16 I_base=(1.54873e-4)*2
17 % logitudes
18 rc1=0.27/2
19 r_pinon=0.75e-2
20 r_corona=3.975e-2
21 lt1=0.1034411
22 lq=0.0164
23 b1=r_pinon/r_corona
24
25 % Valores a sustituir
26 e1=(M_carcaza + M_base + M_corona + M_pendolo)*(rc1^2) + I_carcaza + I_base +
    I_corona + 8*I_perno + (M_perno)*(8)*(lq^2 + rc1^2)
27 mp1=M_pendolo
28 lt1=0.1034411
29 rc1=0.27/2
30 b1=r_pinon/r_corona
31 Ip1=I_pendolo
32 g1=9.81
33 % valor de la friccion de contacto con el suelo
34 fr=0.0124
35 % coeficientes de ecuaciones para linealizacion
36 p= e - 2*mp*lt*rc*cos(b*x3 - x1) + Ip + mp*(lt^2)

```

```

37 q= mp*lt*rc*cos(b*x3 - x1)- Ip - mp*(lt^2)
38 r= mp*lt*rc*sin(b*x3 - x1)*((b*x4 - x2)^2)
39 t= q
40 u= Ip + mp*(lt^2)
41 s= mp*g*lt*sin(b*x3 - x1)
42 w= tau/b
43 f_r=fr*x2
44
45 % calculo del determinante para la linealizacion
46
47 ecua_deter=[p, q*b; q,u*b]
48
49 d_ec= det(ecua_deter)
50
51 d_ec_num=subs(d_ec,{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,0,0,0,e1,mp1,lt1,
rc1,b1,Ip1,g1})
52
53 % declaracion de theta1 dos puntos y alfa dos puntos
54 x2_p=(1/d_ec_num)*[(r+s-f_r)*u*b - (q)*b*(w-s)]
55
56 x4_p=(1/d_ec_num)*[-1*t*(r+s-f_r) + (p)*(w-s)]
57
58 % proceso de linealizacion donde x1 = theta1 y x2 = theta1 punto, X3 = alfa y
59 % x4 = alfa punto. donde la diferencia de bx3 - x1 = 0 , bx4 - x2 = 0
60 a11=subs(diff(x2,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
61 a12=subs(diff(x2,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
62 a13=subs(diff(x2,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
63 a14=subs(diff(x2,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
64 a21=subs(diff(x2_p,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
mp1,lt1,rc1,b1,Ip1,g1})
65 a22=subs(diff(x2_p,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
mp1,lt1,rc1,b1,Ip1,g1})
66 a23=subs(diff(x2_p,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
mp1,lt1,rc1,b1,Ip1,g1})
67 a24=subs(diff(x2_p,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
mp1,lt1,rc1,b1,Ip1,g1})
68 a31=subs(diff(x4,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
69 a32=subs(diff(x4,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
70 a33=subs(diff(x4,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
71 a34=subs(diff(x4,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
,lt1,rc1,b1,Ip1,g1})
72 a41=subs(diff(x4_p,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
mp1,lt1,rc1,b1,Ip1,g1})
73 a42=subs(diff(x4_p,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
mp1,lt1,rc1,b1,Ip1,g1})
74 a43=subs(diff(x4_p,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,

```

```

75 mp1,lt1 ,rc1 ,b1 ,Ip1 ,g1 })
a44=subs (diff(x4_p,x4),{x1,x2,x3,x4,e, mp, lt , rc , b, Ip , g},{0,x2,0,x2/b1,e1 ,
mp1,lt1 ,rc1 ,b1 ,Ip1 ,g1 })
76
77 A=[a11 , a12 , a13 , a14 ; a21 , a22 , a23 , a24 ; a31 , a32 , a33 , a34 ; a41 , a42 , a43 ,
a44 ]
78
79 sistem_linealizado_en_B=[diff(x2,tau);diff(x2_p,tau);diff(x4,tau); diff(x4_p,
tau)]
80 B=[subs(sistem_linealizado_en_B,{x1,x3,e, mp, lt , rc , b, Ip , g},{0,0,e1,mp1,lt1
,rc1 ,b1 ,Ip1 ,g1 })]
81
82 % busqueda del U_eq se iguala theta dos puntos y alfa dos punto a cero.
83
84 x2_psub=subs(x2_p,{x1,x2,x3,x4, mp, lt , rc , b, Ip , g},{0,0,0,0,mp1,lt1 ,rc1 ,b1 ,
Ip1 ,g1 })
85
86 x2_psub1=subs(x2_p,{x1,x2,x3,x4},{0,0,0,0})
87
88 x4_psub=subs(x4_p,{x1,x2,x3,x4, e, mp, lt , rc , b, Ip , g},{0,0,0,0,e1,mp1,lt1 ,
rc1 ,b1 ,Ip1 ,g1 })
89
90 x4_psub1=subs(x4_p,{x1,x2,x3,x4},{0,0,0,0})
91
92 % del desarrollo se tiene que U_eq=0
93
94 C=[diff(x1,x1) diff(x1,x2) diff(x1,x3) diff(x1,x4);diff(x2,x1) diff(x2,x2) diff
(x2,x3) diff(x2,x4);diff(x3,x1) diff(x3,x2) diff(x3,x3) diff(x3,x4);diff(x4,
x1) diff(x4,x2) diff(x4,x3) diff(x4,x4)]
95
96 D=[diff(0,tau)]

```

El resultado de dicha programación da como resultado las siguientes matrices acomodadas con sus respectivas variables de estado.

$$\begin{bmatrix} \dot{x}_{1\delta} \\ \dot{x}_{2\delta} \\ \dot{x}_{3\delta} \\ \dot{x}_{4\delta} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ -3,7951 & -0,3034 & 0,7161 & 0 \\ 0 & 0 & 0 & 1 \\ 486,0901 & 0,4006 & -91,7151 & 0 \end{bmatrix}}_{Matriz_A} \begin{bmatrix} x_{1\delta} \\ x_{2\delta} \\ x_{3\delta} \\ x_{4\delta} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -32,3079 \\ 0 \\ 2,0577e+04 \end{bmatrix}}_{Matriz_B} [T_{in}] \quad (3.11)$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{Matriz_C} \begin{bmatrix} x_{1\delta} \\ x_{2\delta} \\ x_{3\delta} \\ x_{4\delta} \end{bmatrix} + \underbrace{0}_{Matriz_D} [T_{in}] \quad (3.12)$$

Una vez que se tienen las matrices que son propiamente la linealización del sistema, se procede a simularlas en Simulink con un diagrama a bloques.

Una vez que se tienen los diagramas a bloques se puede simular y ver las diferencias entre el modelo lineal y no lineal, esto se muestra en la Figura 3.1.

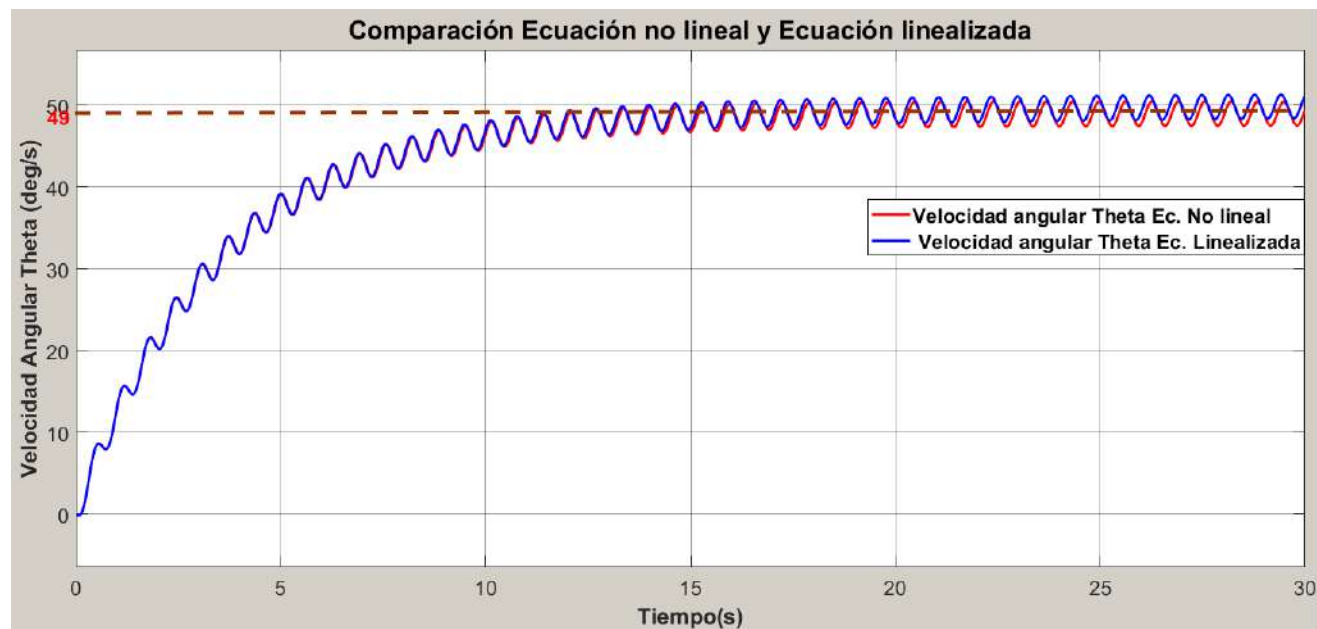


Figura 3.1: Comparación entre el modelo no lineal y lineal.

De la Figura 3.1 se tiene que el modelo linealizado describe la dinámica del sistema no lineal sin error aparente, de modo que se puede calcular la función de transferencia.

3.2. Función de transferencia

Para calcular la función de transferencia se tiene que aplicar la ecuación (3.13) que muestra la forma en la que se calcula dicha función de transferencia con las matrices (A, B, C y D) de la ecuación linealizada.

$$G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + D \quad (3.13)$$

Al aplicar la ecuación (3.13) se tiene que la función de transferencia del sistema queda dada por:

$$G(s) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [s * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 0 \\ -3,7951 & -0,3034 & 0,7161 & 0 \\ 0 & 0 & 0 & 1 \\ 486,0901 & 0,4006 & -91,7151 & 0 \end{bmatrix}]^{-1} \begin{bmatrix} 0 \\ -32,3079 \\ 0 \\ 2,0577e + 04 \end{bmatrix} + D \quad (3.14)$$

Dando como resultado:

$$G(s) = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} \frac{-32,3079s^2+1,1772e+04}{s^4+0,3034s^3+95,5102s^2+27,5410s} \\ \frac{-32,3079s^2+1,1772e+04}{s^3+0,3034s^2+95,5102s+27,5410} \\ \frac{2,0577e+04s^2+6,2302e+03s62387}{s^4+0,3034s^3+95,5102s^2+27,5410s} \\ \frac{2,0577e+04s^2+6,2302e+03s62387}{s^3+0,3034s^2+95,5102s+27,5410} \end{bmatrix} \quad (3.15)$$

Como se puede observar de la ecuación (3.15), el primer elemento de la matriz es la función de transferencia de la posición θ , el segundo elemento es la función transferencia de la velocidad de la rueda $\dot{\theta}$, el tercer término de la matriz es la función de transferencia de la posición angular del motor α y el último elemento es la función de transferencia de la velocidad angular del motor $\dot{\alpha}$.

Debido a que se quiere controlar la velocidad de la rueda a través de una entrada de par torsional en los motores, es necesario usar la segunda función de transferencia de la matriz de la ecuación (3.15), es decir la ecuación (3.16), la cual se evaluará ante una entrada escalón de 0.002N*m y se comparará con la respuestas de las ecuaciones no lineales del robot (ver Figura 3.2), dicho resultado de la comparación se puede ver en la Figura 3.3.

$$G(s_{\dot{\theta}}) = \frac{-32,3079s^2 + 1,1772e + 04}{s^3 + 0,3034s^2 + 95,5102s + 27,5410} \quad (3.16)$$

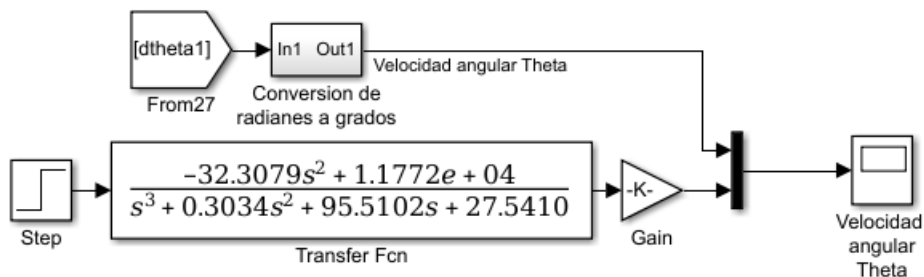


Figura 3.2: Diagrama a bloques de la función de transferencia y comparación con las Ec. de dinámica.

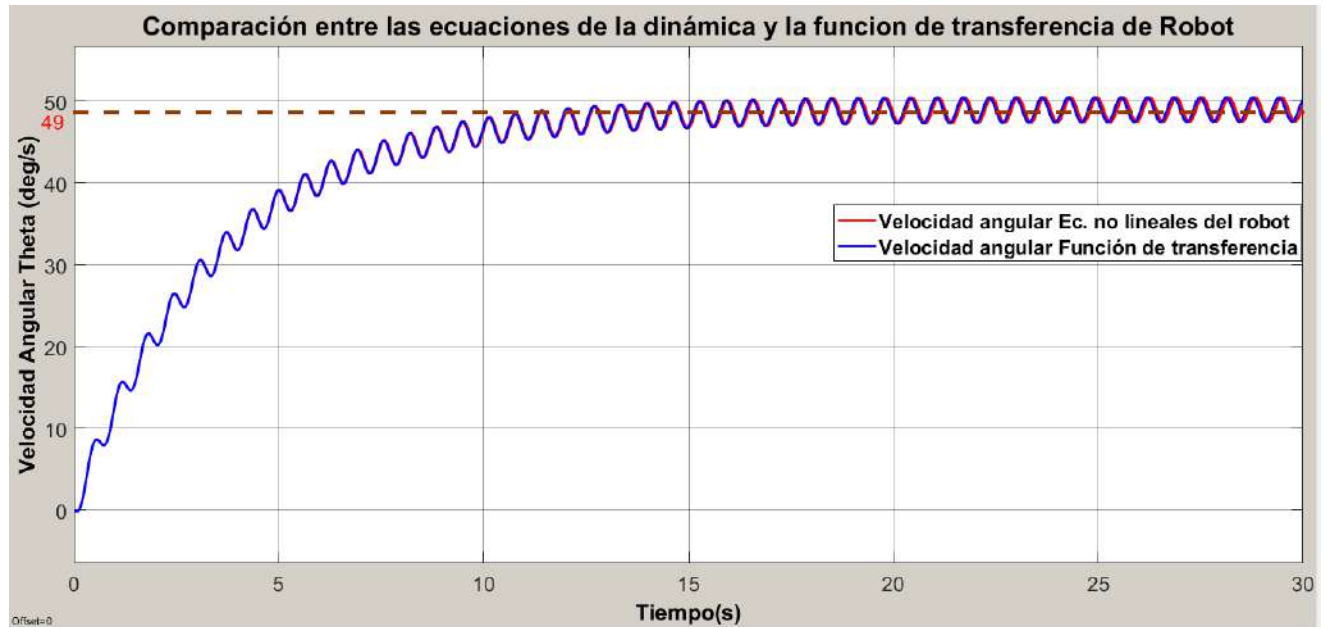


Figura 3.3: Comparación entre el modelo no lineal y la función de transferencia.

Como se puede apreciar en la Figura 3.3 la respuesta de la función de transferencia resulta ser prácticamente idéntica a la del sistema no lineal, de modo que se puede concluir que la función de transferencia es válida.

3.3. Control de la velocidad del robot rueda

Una vez que se tienen tanto las matrices que definen a el sistema como la función de transferencia de la velocidad del robot, se analiza la contrabilidad y la observabilidad del sistema, esto es debido a que se le aplicará control a dicho sistema y que a su vez se hará uso de un observador para minimizar el ruido de la señal de algún sensor mediante el filtro Kalman.

Controlabilidad Un sistema es totalmente controlable si cumple con la condición dada por la ecuación (3.17), por lo que se aplica a las matrices del sistema dando como resultado la ecuación (3.18), de modo que la matriz es no singular, ya que su determinante no es cero, por lo que se puede concluir que el sistema es completamente controlable.

$$\det[B|AB \cdots |A^{n-1}B] \neq 0 \quad (3.17)$$

$$\det \begin{bmatrix} 0 & -32,3079 & 9,8028 & 1,4854e + 04 \\ -32,3079 & 9,8028 & 1,4854e + 04 & -4,5535e + 03 \\ 0 & 2,0577e + 04 & -12,9431 & -1,9030e + 06 \\ 2,0577e + 04 & -12,9431 & -1,9030e + 06 & 1,1903e + 04 \end{bmatrix} \neq 0 \quad (3.18)$$

$$5,9641e + 16 \neq 0$$

Es de importancia señalar que por la ecuación (3.18) el sistema es completamente controlable, sin embargo, esto no asegura que una salida sea completamente controlable, de modo que hace falta ver si las salidas de todo el sistema lo son. Para probar si una salida es completamente controlable se tiene que cumplir la condición dada por la ecuación (3.19), por lo que al aplicar dicha ecuación se tiene que el determinante es diferente de cero, de modo que el sistema tiene salidas totalmente controlables (ver ecuación (3.20)).

$$\det[CB|CAB|CA^2B|\dots|CA^{n-1}B] \neq 0 \quad (3.19)$$

$$\det \begin{bmatrix} 0 & -32,3079 & 9,8028 & 1,4854e + 04 \\ -32,3079 & 9,8028 & 1,4854e + 04 & -4,5535e + 03 \\ 0 & 2,0577e + 04 & -12,9431 & -1,9030e + 06 \\ 2,0577e + 04 & -12,9431 & -1,9030e + 06 & 1,1903e + 04 \end{bmatrix} \neq 0 \quad (3.20)$$

$$5,9641e + 16 \neq 0$$

Observabilidad Para que un sistema sea observable, éste tiene que cumplir con la condición dada por la ecuación (3.21), en donde el determinante debe ser diferente de cero, siendo esto así, se procede a aplicar dicha condición a las matrices del sistema dando como resultado la desigualdad (3.22), por lo que se puede decir que el sistema es completamente observable.

$$\det[C^*|A^*C^*|\dots|(A^*)^{n-1}C^*] \neq 0 \quad (3.21)$$

$$\det \begin{bmatrix} 0 & -3,7951 & 1,1515 & 362,1227 \\ 1,0000 & -0,3034 & -3,7031 & 2,5619 \\ 0 & 0,7161 & -0,2173 & -68,3250 \\ 0 & 0 & 0,7161 & -0,2173 \end{bmatrix} \neq 0 \quad (3.22)$$

$$3,8641e - 14 \neq 0$$

Al tener en cuenta los criterios de controlabilidad y observabilidad se procede a analizar la estabilidad del sistema. Para ello, se le someterá a la planta a las entradas escalón y rampa unitaria, para así, saber su comportamiento, tanto en lazo abierto como en lazo cerrado.

Como se ha abordado, la entrada de la planta es un par torsional en los motores, siendo esto así, la respuesta de la planta en lazo abierto (L.A.) ante una entrada escalón unitario está dada por la Figura 3.4, en donde es posible notar que la ganancia de la planta es de 427.43546.

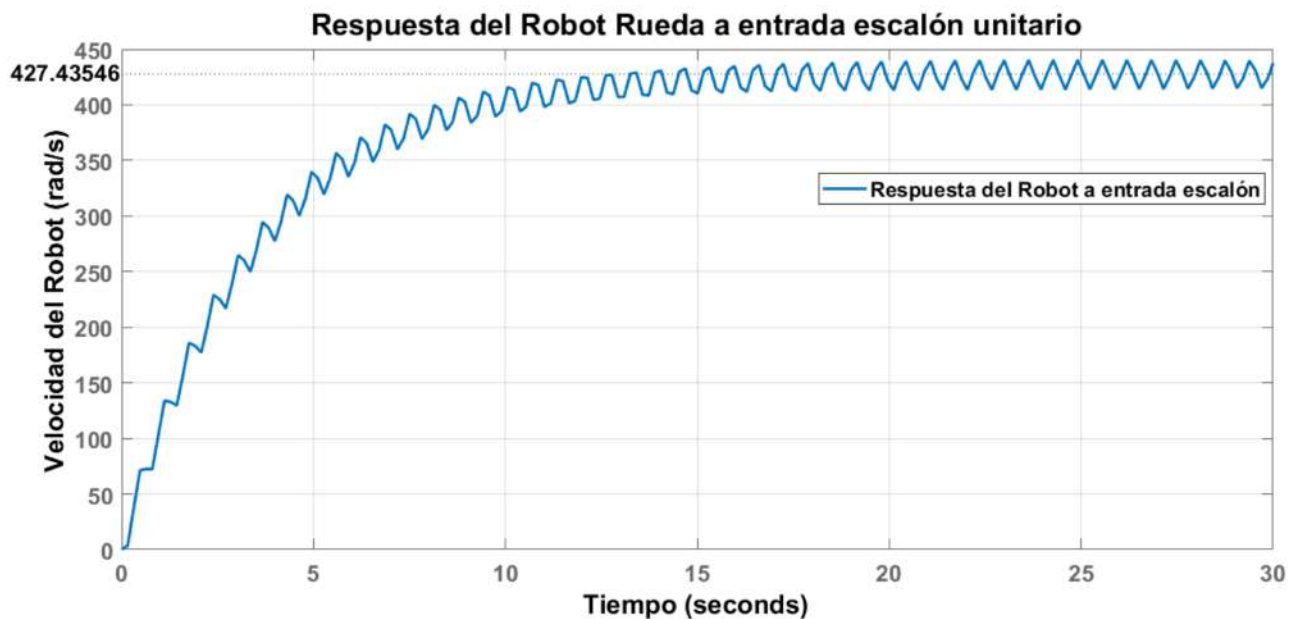


Figura 3.4: Respuesta del Robot ante una entrada escalón unitario.

Es de importancia eliminar la ganancia del sistema para saber la respuesta con una entrada rampa unitaria, de modo que, las respuestas tanto de escalón como de entrada rampa están dadas por las Figuras 3.5 y 3.6.

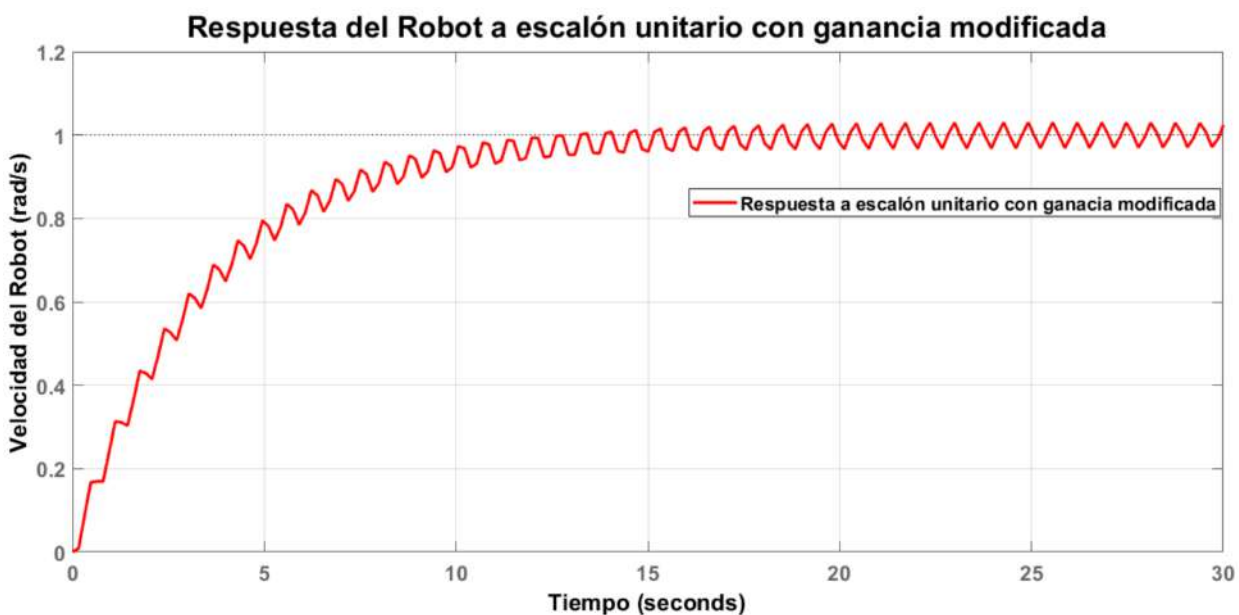


Figura 3.5: Respuesta del sistema con ganancia suprimida ante una entrada escalón.

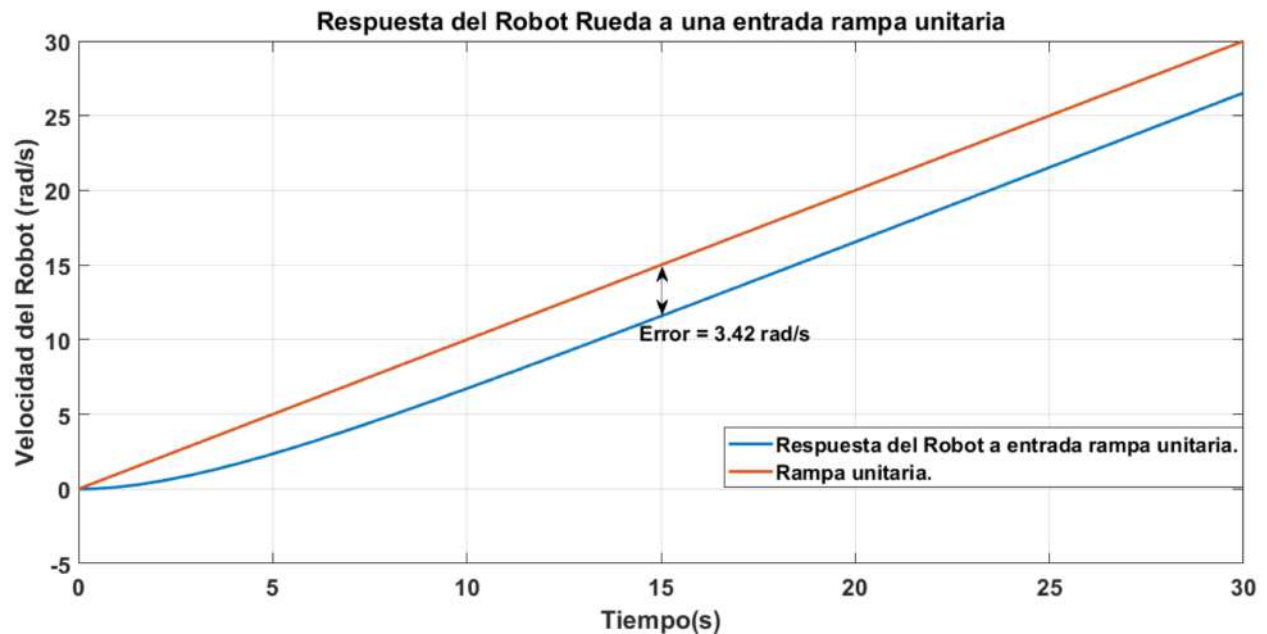


Figura 3.6: Respuesta del sistema con ganancia suprimida ante una entrada rampa.

Como se muestra en la Figura 3.6, el sistema presenta un error en estado estacionario ante una entrada rampa. Por lo que es necesario analizar la estabilidad del sistema en L.A, lo cual es posible mediante el trazado de su lugar geométrico de las raíces (LGR) en L.A (ver Figura 3.7).

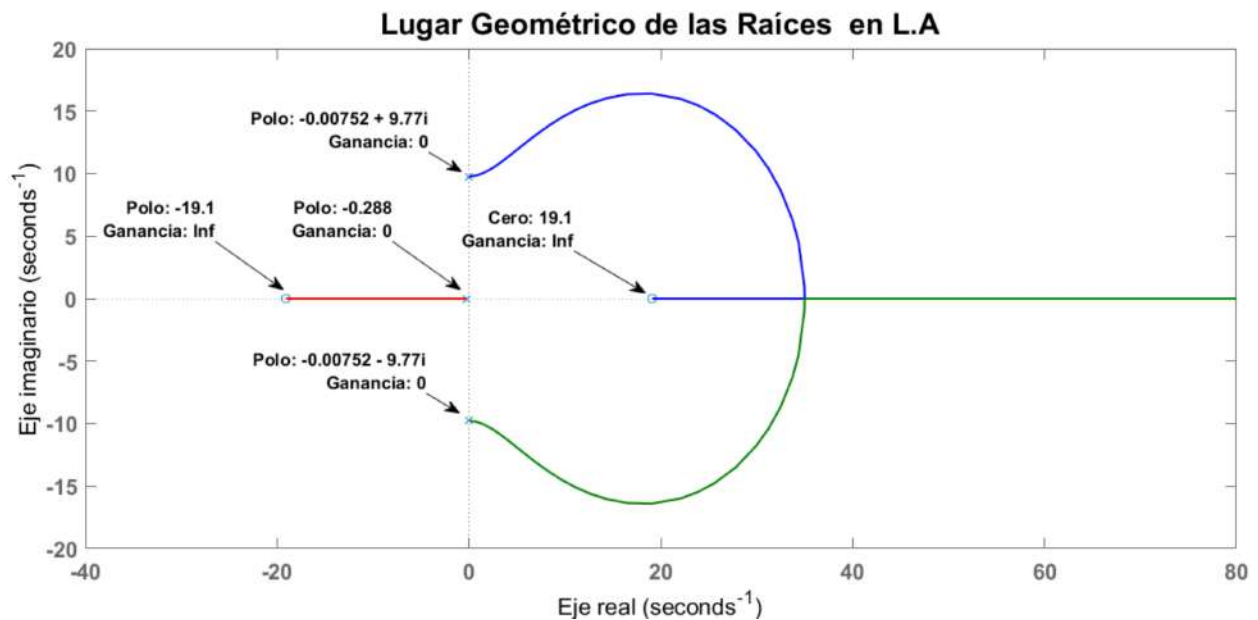


Figura 3.7: Lugar geométrico de las raíces del sistema en L.A.

Como se puede apreciar en la Figura 3.7, el sistema es estable en presencia de todos

los polos en el semiplano izquierdo, en donde además se tiene el par de polos dominantes en $S = -0,00752 \pm 9,77i$, los cuales están en el semiplano izquierdo por lo que el sistema es estable, sin embargo, para el control es necesario cerrar el lazo, en donde se compara la salida con una entrada de referencia, como se muestra en la Figura 3.8, cuya ecuación (3.23) describe el diagrama a bloques.

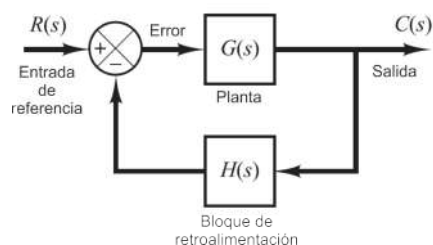


Figura 3.8: Diagrama de sistema de control en lazo cerrado.

$$\begin{aligned} \frac{C(s)}{R(s)} &= \frac{G(s)}{1 + G(s)H(s)} \\ &= \frac{-32,31s^2 + 11772}{s^3 - 32s^2 + 95,51s + 11800} \end{aligned} \quad (3.23)$$

Ahora bien, teniendo la función de transferencia en lazo cerrado es posible evaluar su estabilidad, para ello primero se sugiere una entrada escalón unitario dicho resultado se aprecia en la Figura 3.9.

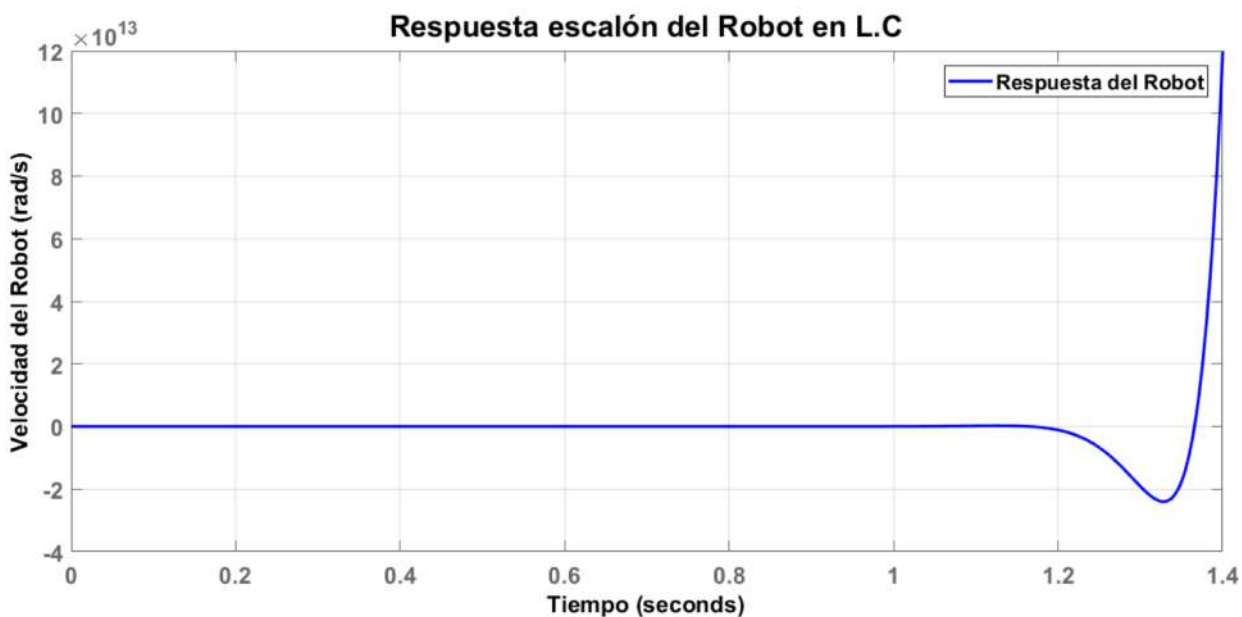


Figura 3.9: Respuesta del sistema ante una entrada escalón en L.C.

Como se puede apreciar, el sistema es inestable, ya que la respuesta del sistema crece de manera exponencial, por lo que es de importancia ver el lugar geométrico de las raíces, como se muestra en la Figura 3.10.

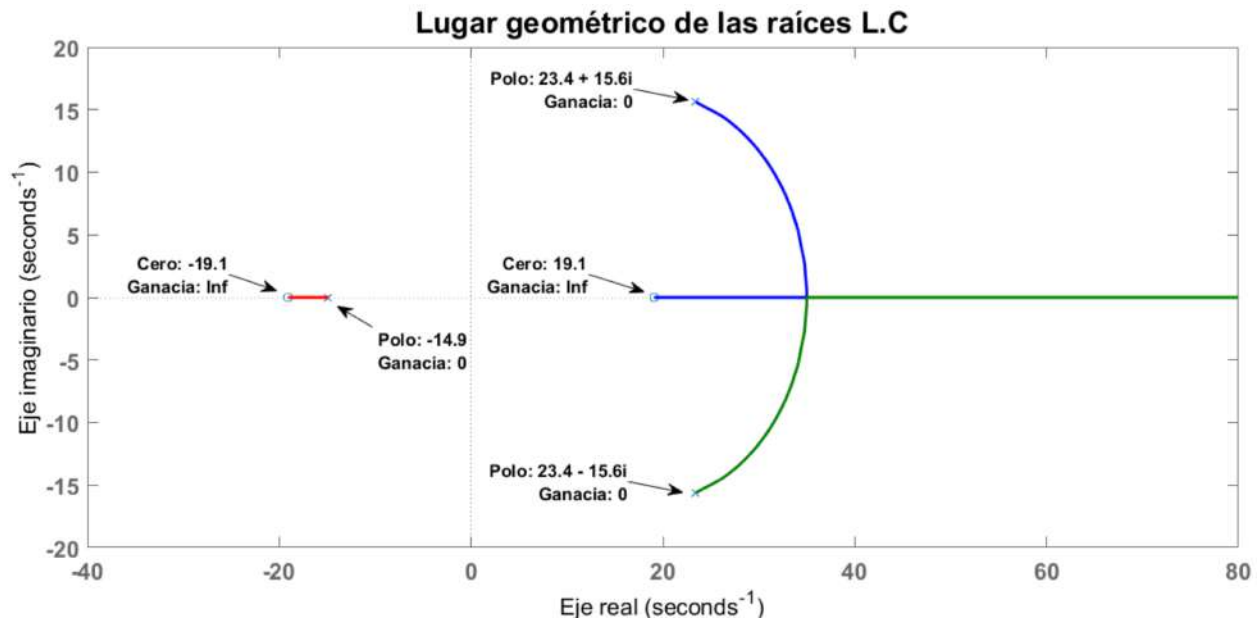


Figura 3.10: Lugar geométrico de las raíces en L.C.

Dicho lugar geométrico revela que hay un par de polos conjugados en el semiplano derecho, de modo que el sistema es inestable, aunque se ajuste la ganancia el sistema será inestable, y esto se puede corroborar con el criterio de estabilidad de Ruth Hurwitz, mediante el uso del polinomio característico, dicho polinomio está dado por la ecuación (3.24).

$$Pol_{caracteristico} = s^3 - 32s^2 + 95,51s + 11800 \quad (3.24)$$

Al tener el polinomio característico se puede ver que, ante la presencia de al menos un coeficiente positivo, hay raíces imaginarias que tienen partes reales positivas. Por lo que al aplicar la metodología de Ruth, como se muestra en la ecuación (3.25) se observan dos cambios de signo, del primer al segundo renglón y del segundo al tercer renglón. Esto significa que existen dos raíces con partes reales positivas, de modo que se llega a la misma conjetura del lugar geométrico de las raíces, en donde se tienen polos en $s = -14,8793599$ y $s = 23,4396799 \pm 15,60853i$, por el par de polos conjugados en el semiplano derecho, el sistema es inestable.

$$\begin{array}{r} s^3 \\ s^2 \\ s^1 \\ s^0 \end{array} \begin{array}{r} 1 \\ -32 \\ 95,51 = \frac{95,51*(-32)-11800}{-32} \\ 11800 \end{array} \begin{array}{r} 95,51 \\ 11800 \\ 0 \end{array} \quad (3.25)$$

Como los polos conjugados son los que dominan al sistema, es necesario saber sus propiedades como es el factor de amortiguamiento y la frecuencia de oscilación, las cuales son 0.83234 y

28.1605 rad/s.

Teniendo en cuenta lo anterior, es posible aplicar una técnica de control para el sistema, dicha técnica a seleccionar es la de diseño de control mediante el lugar geométrico de las raíces, la cual impone las condiciones del sistema como es el coeficiente de amortiguamiento $\zeta = 0,5$ y la frecuencia de oscilación $\omega_n = 1\text{rad/s}$, por lo que, los polos dominantes están definidos por $s_{1,2} = \zeta\omega_n \pm \omega_n\sqrt{1 - \zeta^2}i$, de modo que los polos deseados son $s_{1,2} = -0,5 \pm 0,866i$.

Ahora bien, una vez que se conocen los polos dominantes es de importancia ver la manera en la que se agregarán, es por ello que se hará un análisis del LGR tanto en L.A como en L.C.

En el caso del LGR en L.A (ver Figura 3.7) se tiene que los polos dominantes están en el semiplano izquierdo de modo que la planta es estable, sin embargo cuando se cierra el lazo, el LGR (ver Figura 3.10) indica que la planta se vuelve inestable, y eso es debido a que los polos dominantes pasan al semiplano derecho, en donde un polo dominante busca al cero del semiplano derecho por lo que se cancela con él, y el otro polo dominante se va al infinito, siendo esto así, se tiene un criterio de la causa de la inestabilidad, esto prácticamente proviene de los polos conjugados en el semiplano derecho y el cero ($s = -19,088454$) en el semiplano izquierdo, ya que cancela el único polo presente.

Por lo anterior, se tiene que es de importancia eliminar ese par de polos conjugados y el cero de la planta en L.A, es decir agregar un compensador directo con la planta ($G(s)_{supr}$), por lo que la función de transferencia queda de la siguiente manera:

$$\begin{aligned}
 G(s) * G(s)_{supr} &= \frac{-32,3079s^2 + 1,1772e + 04}{s^3 + 0,3034s^2 + 95,5102s + 27,5410} * \left(\frac{s^2 + 0,01503027s + 95,49204}{(s + 19,088454)(s)} \right) \\
 &= \frac{-32,3079(s + 19,088454)(s - 19,088454)}{(s + 0,288)(s^2 + 0,01503027s + 95,49204)} * \left(\frac{s^2 + 0,01503027s + 95,49204}{(s + 19,088454)(s)} \right) \\
 &= \frac{-32,3079(s - 19,088454)}{(s + 0,288)(s)}
 \end{aligned} \tag{3.26}$$

Ahora bien, es importante notar que la planta queda simplificada, sin embargo, no simplemente se le pueden hacer cambios a la función de transferencia, sino que se tiene que compensar ese cambio con la adición de un compensador de adelanto que cumpla con los requerimientos del funcionamiento de la planta, el cual tiene que compensar el ángulo de deficiencia dada por los polos conjugados deseados en L.C de la planta en L.A.

La ecuación (5.7) cuantifica el ángulo compensar.

$$\begin{aligned}
 Angulo_{comp} &= 180^\circ + AngCeros_{s=-0,5+0,866i} - AngPolos_{s=-0,5+0,866i} \\
 &= 180^\circ - 2,53138^\circ - 103,755^\circ - 120^\circ = -46,28638^\circ
 \end{aligned} \tag{3.27}$$

Sabiendo el ángulo a compensar, se puede proceder mediante el método gráfico a calcular el compensador de adelanto, como se muestra en la Figura 3.11 en donde el polo está dado en -1.65520103 y el cero en -0.604129, siendo esto así, la función de transferencia del sistema compensado en L.A queda dado por la ecuación (3.28).

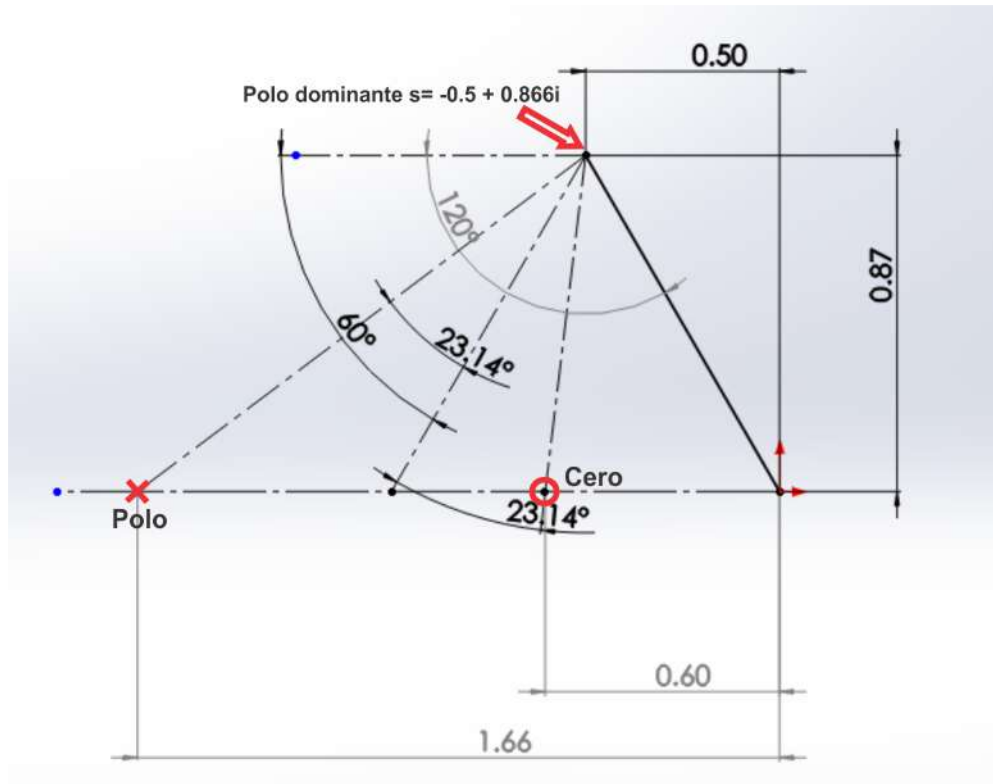


Figura 3.11: Metodo grafico para calcular el compensador de adelanto.

$$G_T = G(s) * G(s)_{comp} = \frac{-32,3079(s - 19,088454)}{(s + 0,288)(s)} * \frac{K(s + 0,604129)}{s + 1,65520103} \quad (3.28)$$

Ahora, solo falta calcular la ganancia \mathbf{K} del sistema para que tenga los polos dominantes en el lugar deseado, por lo que se debe de cumplir la condición de magnitud y ángulo, en donde la magnitud debe ser 1 y ángulo de 180° en el polo deseado. La compensación de ángulo está dada por el polo y el cero del compensador respecto a los polo dominantes, por otro lado, la condición de magnitud se puede encontrar al sustituir el polo dominante deseado $s = -0,5 + 0,866i$ en la función de transferencia (3.28) e igualar a -1.

$$\begin{aligned} \frac{-32,3079(-0,5 + 0,866i - 19,088454)}{(-0,5 + 0,866i + 0,288)(-0,5 + 0,866i)} * \frac{K(-0,5 + 0,866i + 0,604129)}{-0,5 + 0,866i + 1,65520103} &= -1 \\ (-429,2451 - 0,16203i)K &= -1 \\ 429,2452K_{\angle -179,97^\circ} &= -1 \\ K &= |2,32967e - 03| \end{aligned} \quad (3.29)$$

Al tener la ganancia \mathbf{K} , es posible establecer la función de transferencia en lazo abierto como se muestra a continuación:

$$G_T = \frac{-32,3079(s - 19,088454)}{(s + 0,288)(s)} * \frac{(2,32967e - 03)(s + 0,604129)}{s + 1,65520103} \quad (3.30)$$

Para probar si el diseño cumple con las especificaciones, se procede a cerrar el lazo y se grafica su lugar geométrico de las raíces, como se muestra en la Figura 3.12.

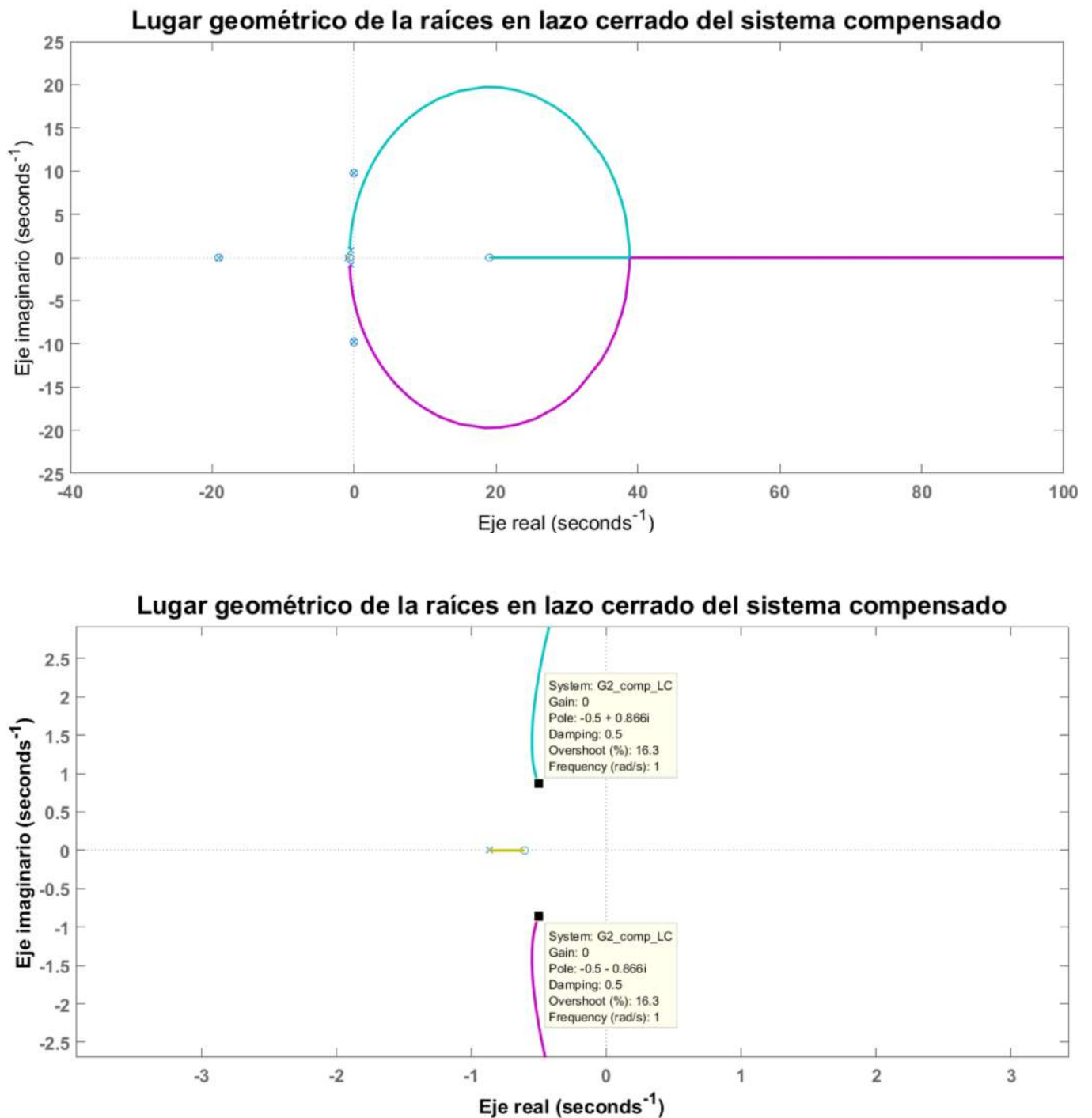


Figura 3.12: Lugar geométrico de las raíces del sistema compensado.

Ahora bien, como se puede apreciar en la Figura 3.12, el sistema cumple con las características deseadas, por lo que ahora se simulará en Matlab simulink (ver Figura 3.13) para ver

la respuesta del sistema.

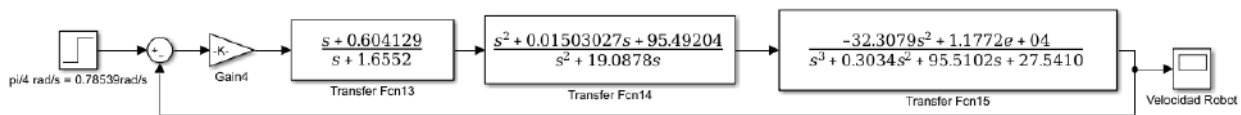


Figura 3.13: Diagrama a bloques del sistema controlado.

El resultado del control se muestra en la Figura 3.14 en donde se comparan las respuestas del sistema tanto en lazo abierto con par torsional constante y lazo cerrado con control, de modo que se puede observar que el control tiene una gran eficiencia, basada en los parámetros de sobre elongación (1.25) y tiempo de asentamiento (5% de tolerancia del valor final de asentamiento). En cuanto al tiempo de asentamiento, que es prácticamente 205% más rápido que el sistema controlado con par torsional constante.

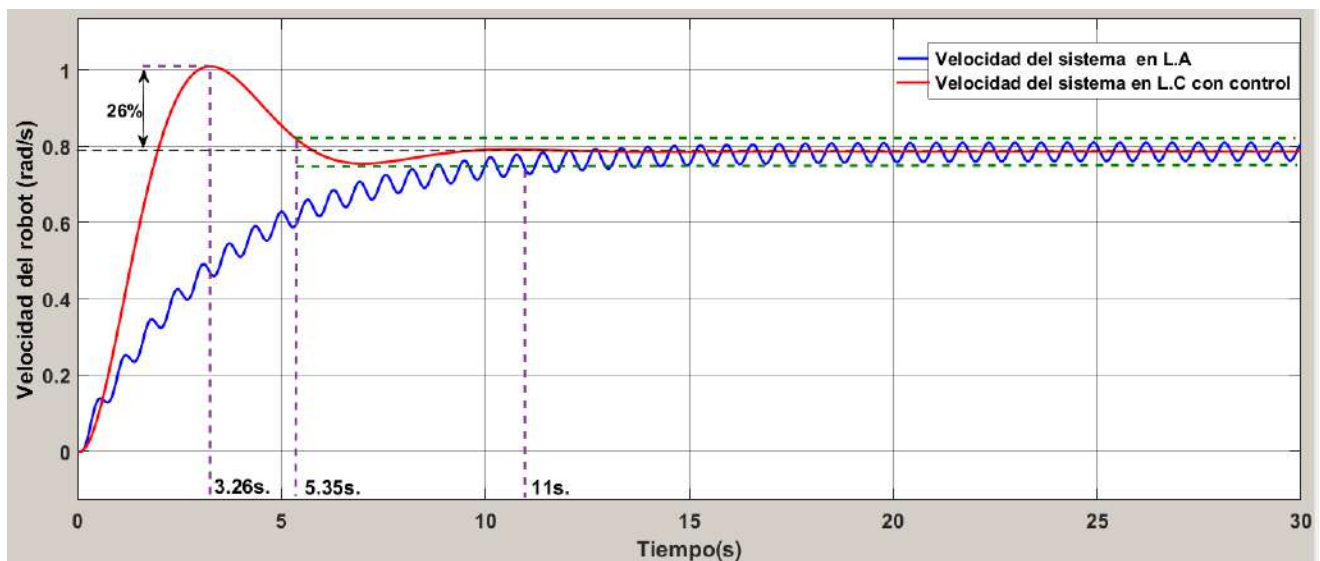


Figura 3.14: Comparación de las respuestas del sistema con control y en L.A.

Es de importancia notar que la respuesta del sistema podría mejorarse aun más modificando los requerimientos del control, sin embargo, se tienen que considerar las características del actuador para realizar dicho control, por lo que hay que observar que el par requerido para que el sistema pueda comportarse de la manera deseada.

Mediante el diagrama a bloques se adquiere la señal del par de control, la cual se muestra en la Figura 3.15 en donde se tiene que el par máximo requerido es de 5.325 N*mm por lo que es ideal para el motor pololu de 107.91 N*mm que se tiene planeado utilizar, así que el control se puede hacer más eficiente o incrementar la velocidad de la rueda, que en este caso la velocidad deseada fue de $\frac{1}{8}$ rev/s o 45°/s, la cual es sumamente lenta, por lo que incrementar este valor sería una opción.

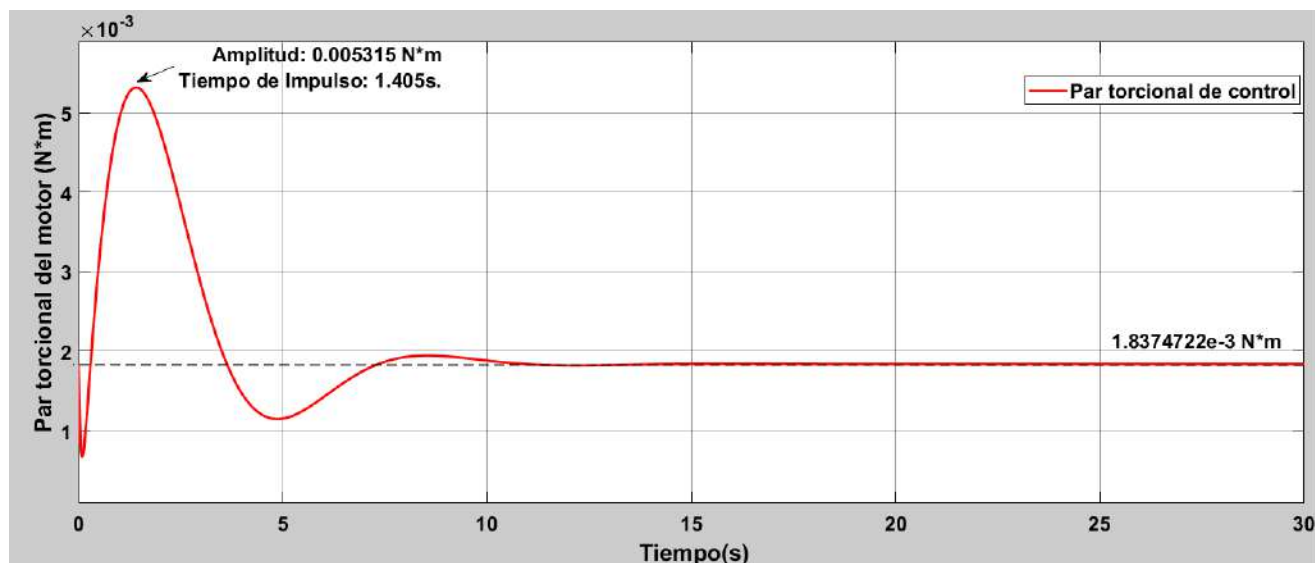


Figura 3.15: Par torsional requerido para el control de la planta.

De los resultados numéricos obtenidos, se puede concluir que el diseño del controlador cumple con el objetivo de estabilizar el sistema y de inducir la dinámica deseada, ya que se sabe que polos y ceros de la planta se tienen que cancelar mediante un compensador, y así lograr que el sistema sea estable en lazo cerrado.

3.4. Simulación del control de velocidad del robot en ADAMS View y filtro Kalman en Matlab

En esta sección se presenta la validación del controlador diseñado mediante la técnica de co-simulación. Esto es posible mediante ADAMS View en donde se exporta la planta con todas sus características dinámicas, en un bloque para que pueda ser utilizado en Matlab Simulink en donde se implementa el control de par torsional.

Para realizar lo anterior, es necesario exportar el diseño de SolidWorks a ADAMS View, esto es mediante la exportación de cada elemento del Robot Rueda como archivo en parasolid ($.x_t$), y posteriormente realizar el ensamble en ADAMS, Dicho ensamble se muestra en la Figura 3.16, el cual contiene las mismas propiedades que SolidWorks, por lo que, al robot se le somete una entrada de par torsional constante de $2 \text{ N}^* \text{ mm}$, obteniendo la respuesta mostrada en la Figura 3.17, la cual se asemeja bastante a la obtenida tanto en SolidWoks como en Simulink a través de la solución de las ecuaciones dinámicas.

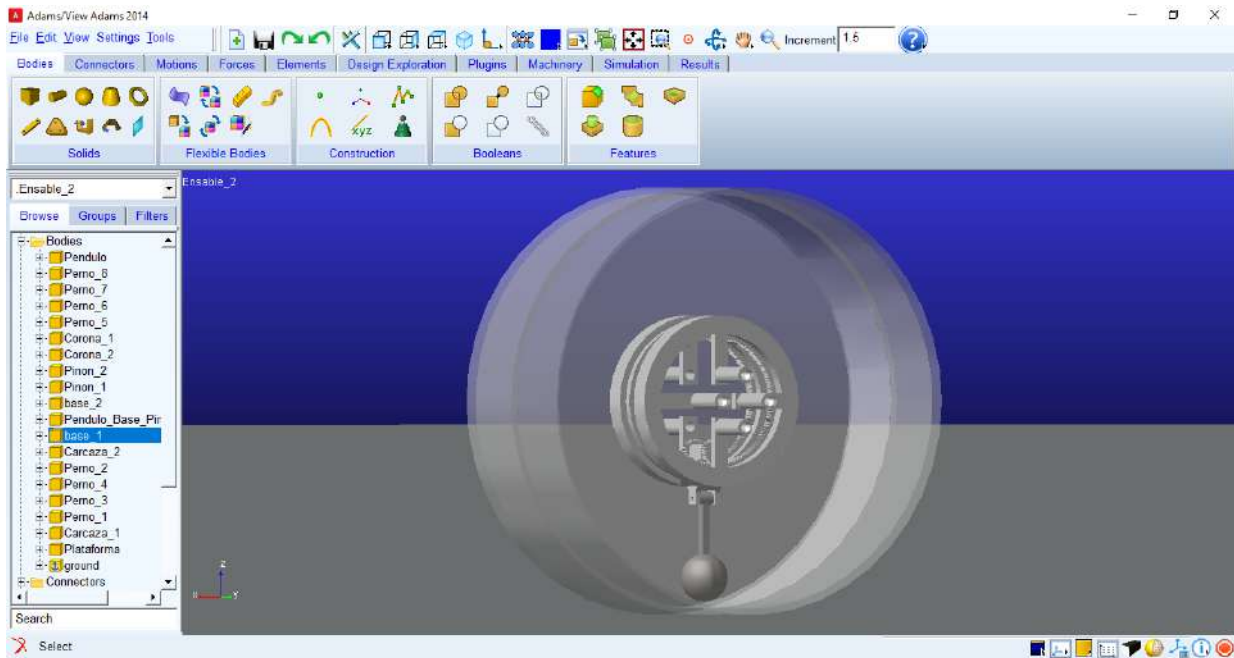


Figura 3.16: Diseño del Robot Rueda en Adams view.

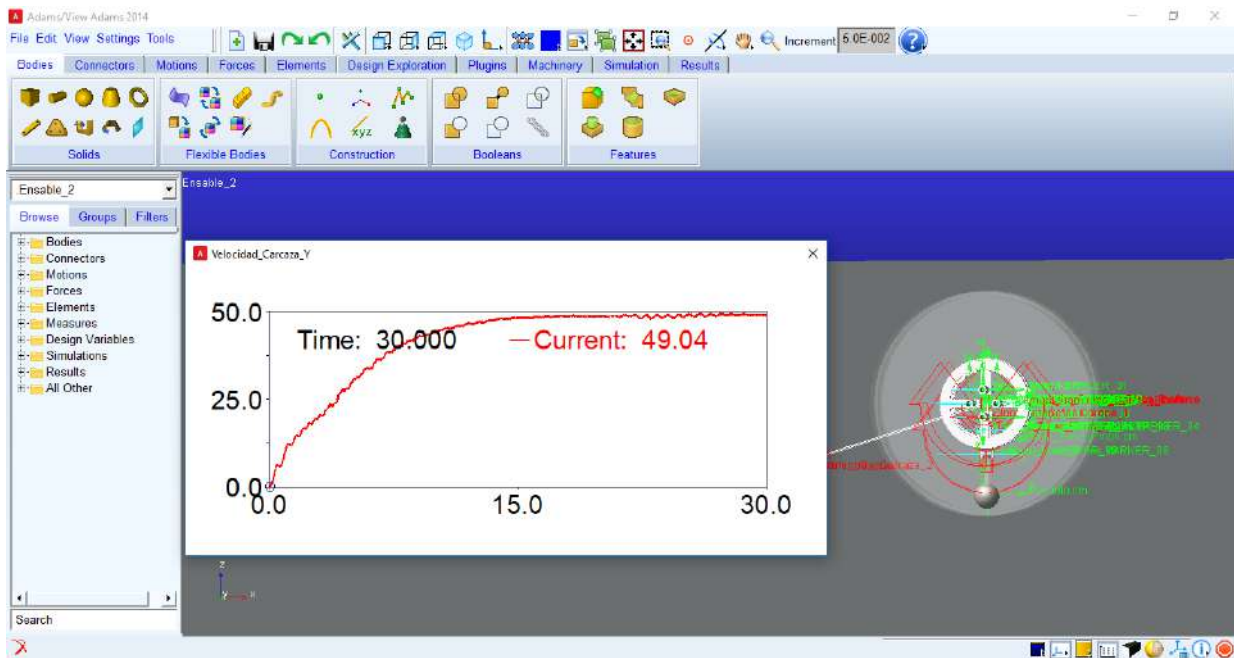


Figura 3.17: Velocidad del Robot Rueda ante una entrada de par constante(0.002Nm).

Una vez que se tiene el diseño realizado y probado en Adams View es posible aplicar control a dicha planta, para ello se tiene que exportar a Matlab en donde se genera un bloque de la planta en Matlab Simulink (ver Figura 3.18).

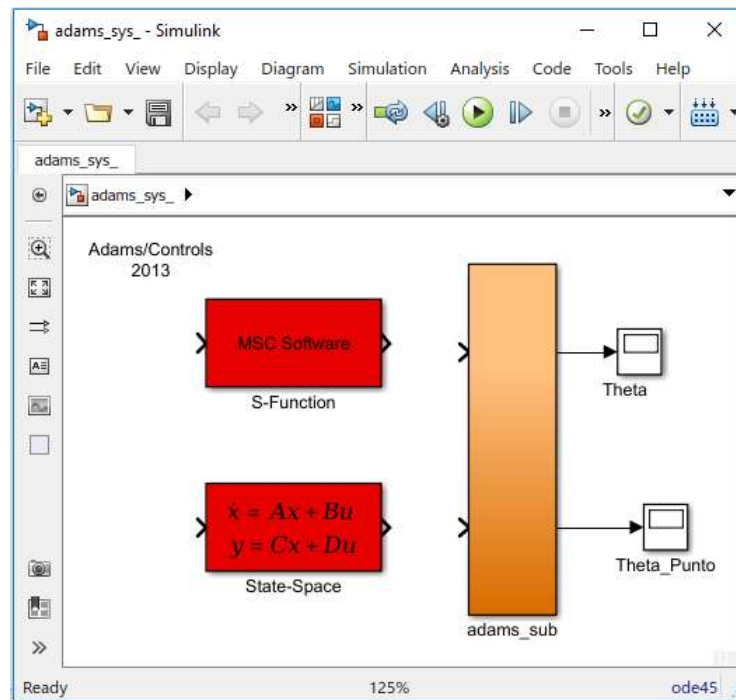


Figura 3.18: Planta exportada de Adams View a Matlab Simulink.

En la Figura 3.19 se muestra la implementación del sistema en lazo cerrado para la co-simulación ADAMS-Matlab Simulink. La respuesta de la co-simulación se compara con la simulación de las ecuaciones del sistema en lazo cerrado, como se observa en la Figura 3.20.

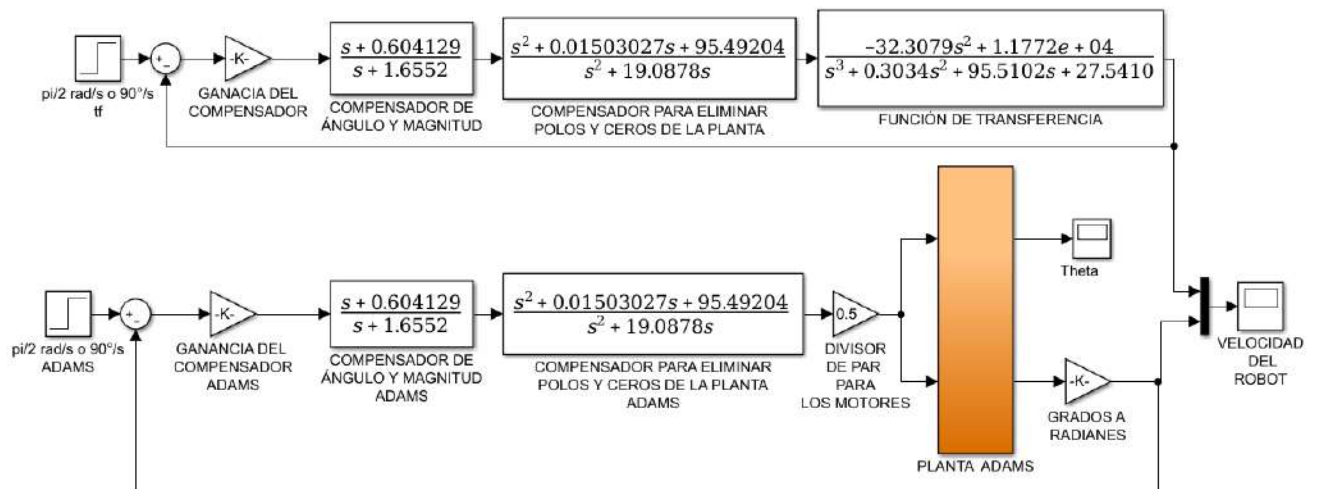


Figura 3.19: Diagrama a bloques del control de la planta de Adams View.

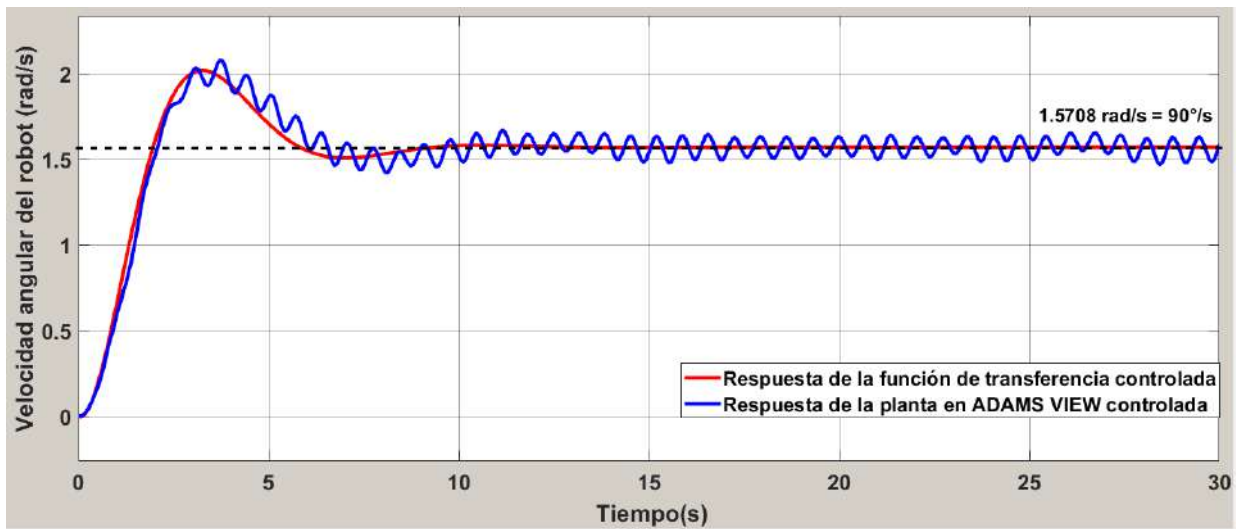


Figura 3.20: Comparación de las respuestas de control tanto de Adams View y la función de transferencia.

Para eliminar las oscilaciones de la respuesta del sistema en lazo cerrado en la co-simulación que se aprecian en la Figura 3.20, se propone el uso de un filtro Kalman, cuya implementación en la co-simulación se muestra en la Figura 3.21. En la Figura 3.22 se muestran los parámetros de dicho filtro.

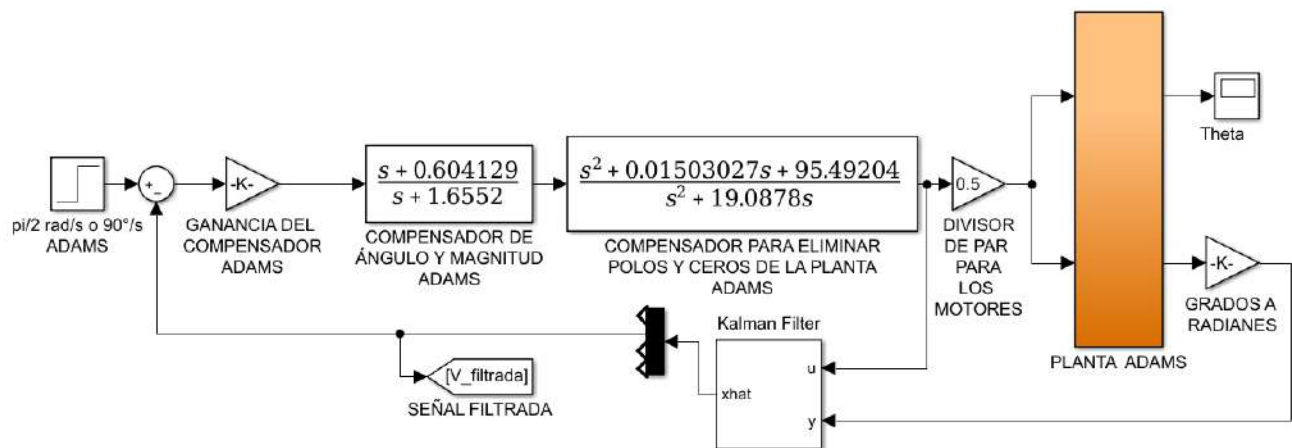


Figura 3.21: Diagrama a bloques del sistema con filtro Kalman.

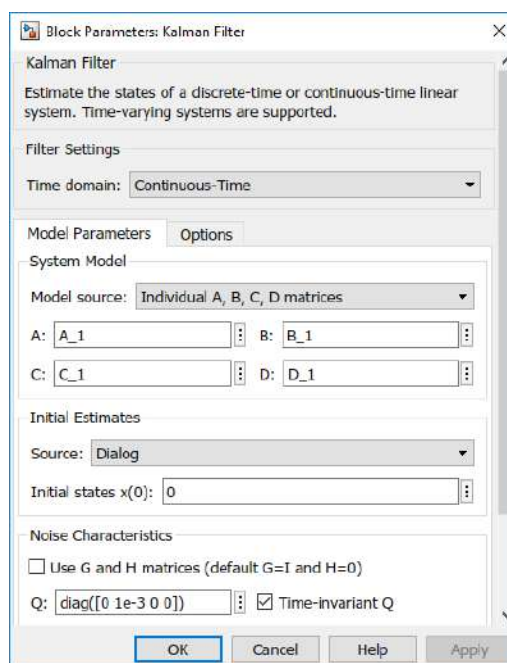


Figura 3.22: Parámetros del filtro Kalman.

Como se puede observar, se tiene que el filtro Kalman requiere prácticamente de las matrices de estado de la planta, el cual se transforma en un observador. Y por la entrada de control y la lectura del sensor, se predice el velocidad del robot mediante una convolución de ambas señales.

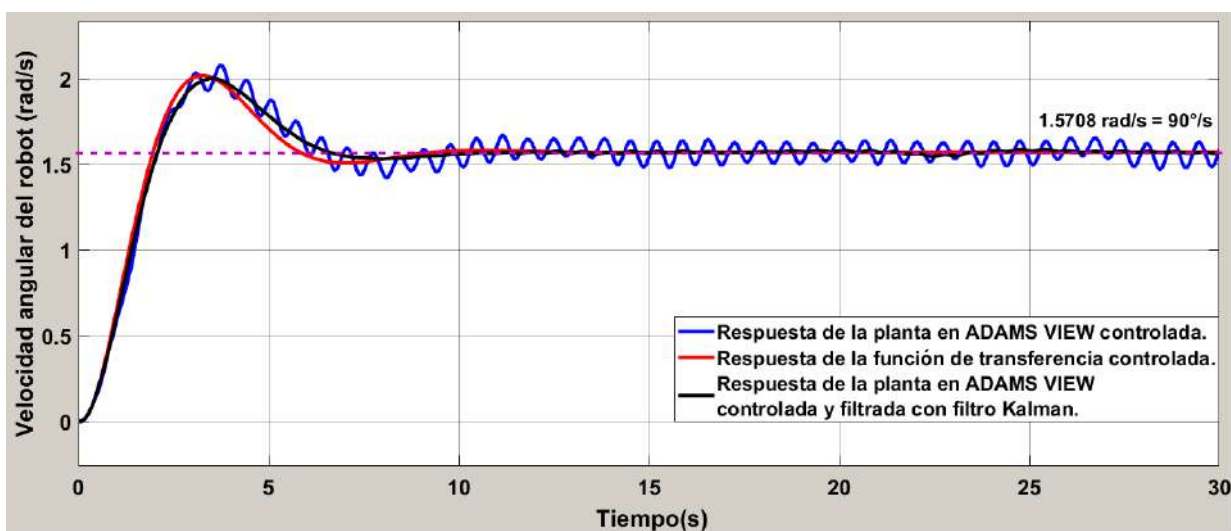


Figura 3.23: Comparación de las respuestas del sistema.

Como se aprecia en la Figura 3.23, se tiene que el filtro Kalman es todo un éxito, ya que filtra a la perfección la señal adquirida del sensor y se asemeja a la respuesta de la función de transferencia, por lo que será de gran utilidad para filtrar la señal adquirida del sensor

giroscopio del robot rueda.

El control se diseñó al inspeccionar el LGR de la función de transferencia de la planta en L.A y L.C, de modo que, así se puede conocer su comportamiento ante la variación de ganancia y/o re-ubicación de los polos y ceros.

Al haber aplicado la metodología del LGR al sistema, se obtuvieron resultados satisfactorios ya que cumple con las características deseadas, así como también, se logró analizar qué polos y ceros hay que eliminar, para que la planta deje de ser inestable en L.C.

El diseño del controlador se validó mediante resultados numéricos de la simulación de las ecuaciones del sistema en lazo cerrado y también con la co-simulación ADAMS-Matlab. La adición de un filtro de Kalman permitió mejorar la respuesta del sistema en lazo cerrado.

Capítulo 4

Manufactura e implementación del robot rueda

El diseño del robot contempla un conjunto de materiales para cada una de las partes del mismo, de modo que, para manufacturarlo se tienen que hacer pruebas y evaluaciones para cumplir con el peso y las medidas del mismo. Sin embargo, no todo el robot va ser apegado al prototipo de la simulación, ya que se le tienen que añadir elementos físicos para que posibilite su funcionamiento, puesto que en la simulación es posible asignarles a los motores un par torsional, sin la necesidad de colocar baterías y demás elementos electrónicos.

4.1. Sistema electrónico

En esta sección se abordará una parte importante del robot, el análisis de los componentes electrónicos necesarios que posibilitan el movimiento del motor acorde al control establecido por una PC mediante comunicación inalámbrica. Así como también, se abordará la parte de la implementación de la electrónica.

A continuación, se muestra de manera ordenada la implementación:

1. Tareas necesarias a realizar y elección de dispositivos.
2. Conexión de los dispositivos mediante un software de simulación
3. Programación del microcontrolador.
4. Simulación del sistema.
5. Pruebas.
6. Construcción física.

4.1.1. Tareas necesarias a realizar y elección de dispositivos

La PC será la encargada de hacer el control del robot con base en las mediciones recibidas del robot, que son la corriente de consumo del motor y la posición angular del robot. Para

lograr esto es necesario hacer las siguientes tareas:

1. Enviar datos de manera inalámbrica entre un microcontrolador y la PC.
2. Controlar la velocidad del motor a partir de los datos recibidos.
3. Sensar la posición del robot y enviarla.
4. Sensar la corriente de consumo del motor, para así estimar el par torsional que éste proporciona.
5. Alimentación de los componentes de manera aislada.

Las tareas anteriormente descritas se pueden visualizar por el esquemático dado por la Figura 4.1. Por otra parte se hace una descripción y un análisis de las tareas a realizar.

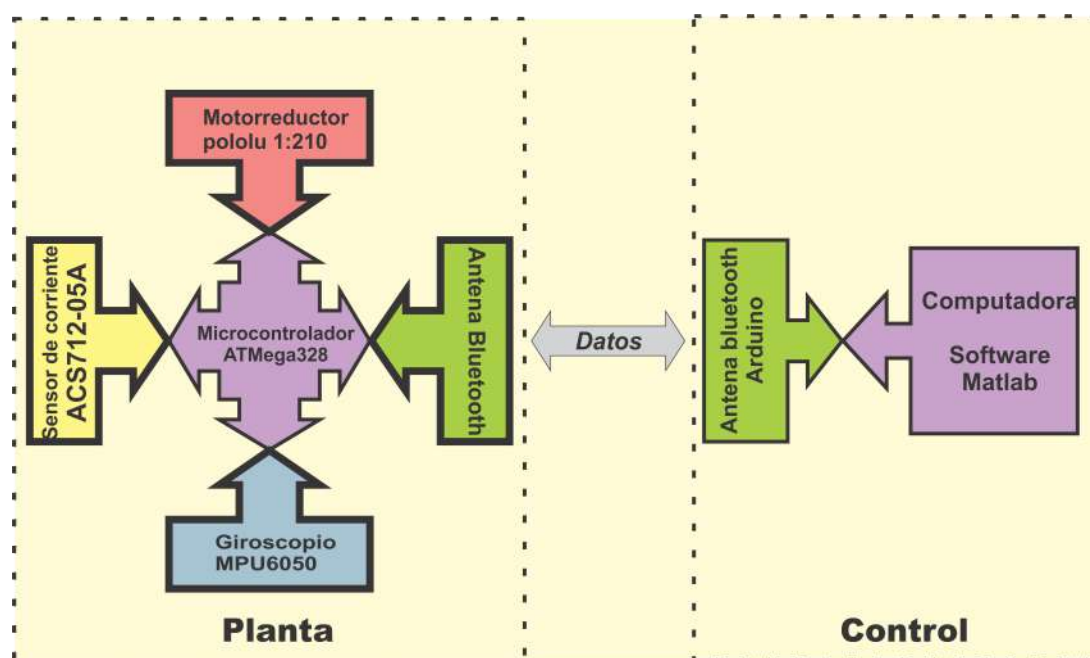


Figura 4.1: Diagrama a bloques del funcionamiento eléctrico.

Para la **primera tarea** es necesario utilizar una antena Bluetooth en el microcontrolador ATmega328p, así como también, otra antena Bluetooth en un Arduino Mega para la transmisión inalámbrica de datos, en donde se utilizará la comunicación USART.

En la **segunda tarea** se tiene que elegir un motor que cumpla con los requerimientos de velocidad y par torsional, de modo que, se elige un Motoreductor Pololu con relación 1:210, el cual tiene una par torsional de $3\text{kg}\cdot\text{cm}$ (El parámetro para la elección del motor, se tomó a partir del par torsional necesario para mover el robot obtenido de la simulación en el software SolidWorks). Una vez elegido el motor, es necesario tener un *driver* y a su vez una etapa de potencia que posibilite el control eficiente del motor mediante PWM, por lo que, el *driver* que cumple con las características deseadas es el integrado L293D (ver hojas de datos [58]).

Para la **tercera tarea** es necesario tener un sensor que posibilite medir la posición del robot en cualquier instante de tiempo, en este caso se hará uso de un **Giroscopio MPU6050** (ver hojas de datos [59] y [60]), el cual usa como protocolo de comunicación I2C o TWI para poder acceder a los registros del sensor. En la **cuarta tarea** es necesario utilizar un sensor compacto y que tenga una sensibilidad adecuada para poder estimar la cantidad de corriente que consume un dispositivo, es por ello que se elige el **sensor ACS712-05A** (ver hoja de datos [61]), el cual tiene la peculiaridad de expresar la cantidad de corriente mediante voltaje en sus terminales de salida. Dicho voltaje debe ser interpolado para saber la corriente circulante, por lo que, el encargado de esta tarea es el microcontrolador del robot. La **quinta tarea** consiste en aislar la alimentación del robot, la parte lógica de la parte del actuador (motor), esto es debido a que la antena bluetooth es sensible en el aspecto de variaciones de voltaje, y puesto que el motor elegido demandará corriente elevada, lo que posibilitará una caída de voltaje. De modo que, se tienen 2 alimentaciones, la primera alimentación es una batería de una celda de litio, la cual es la encargada de alimentar a la antena bluetooth, al giroscopio, al sensor de temperatura y al microcontrolador. Y por otra parte, se tiene la alimentación exclusiva para el motor y el *driver* correspondiente, dicha alimentación consiste en 8 baterías de una celda de litio conectadas en paralelo para dividir propiamente la corriente, por lo que la conexión de las baterías ofrecerá un voltaje de 3.7v. Sin embargo, puesto que se hizo esa mejora en cuanto al suministro de corriente, el voltaje proporcionado por las baterías no es de utilidad ya que el motor no alcanzará su velocidad máxima, para esto necesita tener un voltaje de 9v, por lo que es necesario hacer uso de un convertidor DC-DC tipo boost elevador, de modo que se elige el convertidor DC-DC XL6009 que ofrece una corriente máxima de salida de 3A.

4.1.2. Conexión de dispositivos

Una vez identificados los elementos a utilizar se procede a hacer uso de un software de simulación en este caso es **Proteus 8**, dando como resultado el esquemático mostrado en la Figura 4.2.

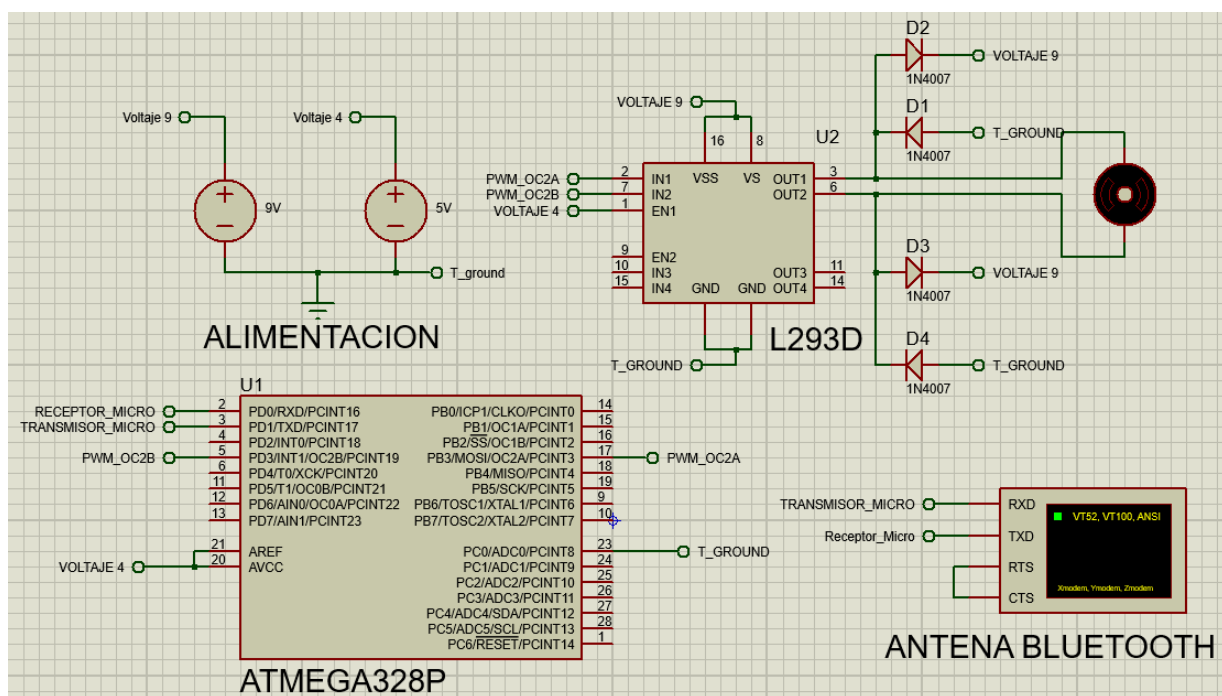


Figura 4.2: Interconexión de elementos electricos internos del robot.

4.1.3. Programación del microcontrolador ATmega328p

Para la programación del microcontrolador se considerarán los elementos soportados por el software de simulación que en este caso son el microcontrolador ATmega328p, el *driver* L293D, una terminal de comunicación serial y un motor, se excluye el giroscopio debido a que no es soportado en el software de simulación para las posteriores pruebas, de ser así, se optará por modificar el código para la adición del giroscopio en la etapa implementación.

El software a utilizar para programación del microcontrolador es el programa Atmel Studio 7.0, por lo que, el código se muestra en el Apéndice B.

4.1.4. Pruebas de programación y simulación

Al simular dicho circuito implementado en el software Proteus 8 (ver Figura 4.2) con el código realizado, se logran obtener resultados factibles para comenzar a realizar la implementación de la circuitería de manera física.

4.1.5. Construcción física del circuito

Es necesario complementar el código de programación del microcontrolador para el uso del giroscopio MPU6050, el *driver* L293D, y la antena bluetooth, la cual tendrá la función de mostrar las lecturas de posición angular del robot, dicho código se puede visualizar en el Apéndice C.

Una vez abordado el código es necesario cargarlo en el microcontrolador de manera física, de

modo que, se hace uso del software Progisp 1.72 y del programador ISP.

Con el microcontrolador programado, es posible implementar el circuito en donde el microcontrolador lee los datos en donde se añadirán 2 elementos más, que son el convertidor CD-CD XL6009 y la antena bluetooth.

Una vez implementado, es necesario visualizar los datos de posición en la computadora, de modo que, una forma mas cómoda de visualizar los datos del giroscopio es mediante un entorno gráfico cuya opción más viable es Matlab Simulink, siendo esto así, se tiene que realizar la etapa del apareamiento de las antenas bluetooth, que conllevan a unir una antena en el Arduino mega y posteriormente a la computadora.

4.2. Comunicación circuito del robot y Matlab Simulink

Teniendo ya apareadas las antenas, es posible establecer comunicación entre el circuito (ver la Sección 4.1) y la computadora, de modo que, se hará uso del Arduino Mega, cuya razón es la necesidad de trabajar en tiempo real y la necesidad de 2 puertos seriales de comunicación, de los cuales, uno es para la antena bluetooth (puerto 1) y otro para comunicar el arduino con la computadora (puerto 0), tal y como se observó en el Figura 4.1.

La necesidad de trabajar en tiempo real es debido a que se aplicará un algoritmo de control para el robot, por lo que, se tiene que usar un programa en el que se pueda implementar y ejecutar en tiempo real como es Matlab Simulink, es por ello que se eligió como primera alternativa para mandar los datos del giroscopio.

Todo lo anterior se abordará en el siguiente orden:

1. Instalación de paquetes necesarios en Matlab Simulink (ver Apéndice D).
2. Construcción y programación de diagrama a bloques para recepción de datos de Arduino.
3. Pruebas de comunicación.

4.2.1. Construcción y programación de diagrama a bloques de Arduino

La parte de la construcción a bloques consistirá de las siguientes tareas:

- * Lectura de datos recibidos por el Arduino Mega en el puerto de la Antena bluetooth (puerto 1).
- * Los datos recibidos por la antena Bluetooth no son más que la información de la posición actual de robot, dichos datos, tienen la característica de tener un caracter que indica el inicio y fin del dato, por lo que hay que decodificar los datos adquiridos mediante un bloque construido en el entorno Script de Matlab.
- * Una vez que se convierten los datos es necesario visualizarlos en un entorno gráfico.
- * Configuración de los parámetros de simulación para la ejecución en tiempo real.

Lectura de datos

Para la lectura de datos se hará uso del bloque Serial Receiver en donde se selecciona el puerto 1 para la lectura de datos, ya que el puerto 0 se usa para la comunicación entre el Arduino Mega y la computadora ejecutando Matlab Simulink.

Además del bloque añadido es necesario agregar un bloque más, el cual tiene la función de convertir los datos tipo **int 8**, que son propiamente el tipo de dato que manda el microcontrolador a la computadora.

Decodificación de datos

Los datos recibidos por parte del microcontrolador vienen decodificados, en donde se indican dos caracteres, uno de inicio de dato y uno de fin de dato. Por lo que, es necesario detectar cuando inicia y cuando termina el dato recibido, así como también identificar el signo que lleva dicho número. Por lo que al usar el bloque **Matlab Function** se programa la decodificación de los datos.

El bloque de función pudiera tener más de una entrada y salida dependiendo de los requerimientos, tal y como se puede visualizar en el código de decodificación en el Apéndice E en donde se tiene una entrada y dos salidas, la única entrada no es más que cada carácter de llegada del microcontrolador que hace referencia al dato de la posición del robot, y la primera señal de salida es el dato de posición en concreto y la segunda señal de salida indica si el robot gira en un sentido u otro.

Visualización de datos.

Para la visualización de datos es necesario utilizar la herramienta Scope en donde se usarán las características por definición.

4.2.2. Pruebas

Teniendo el diagrama de bloques, es posible realizar las pruebas de la recepción de datos, en donde se enciende el circuito construido en la Sección 4.1 y, realizan las conexiones correspondientes con el arduino Mega y su correspondiente antena Bluetooth como se muestra en la Figura 4.3.

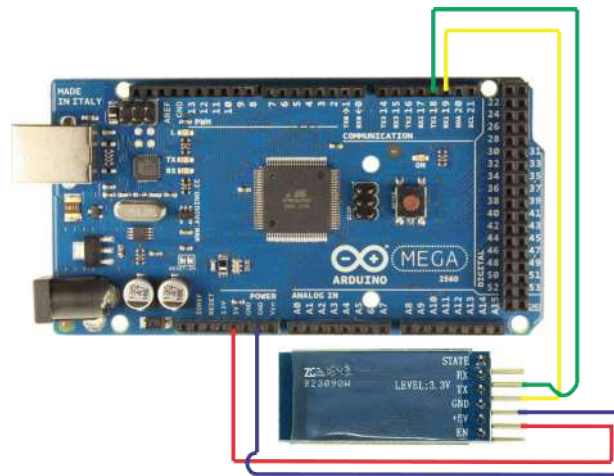
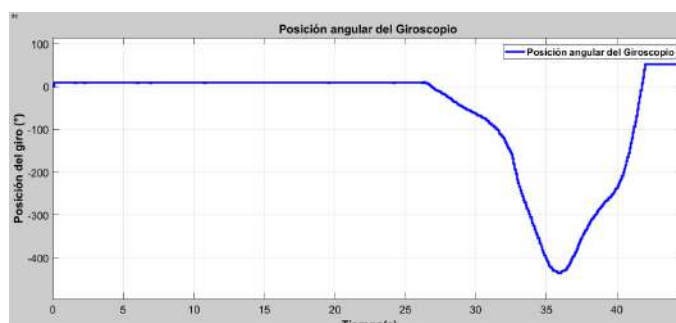
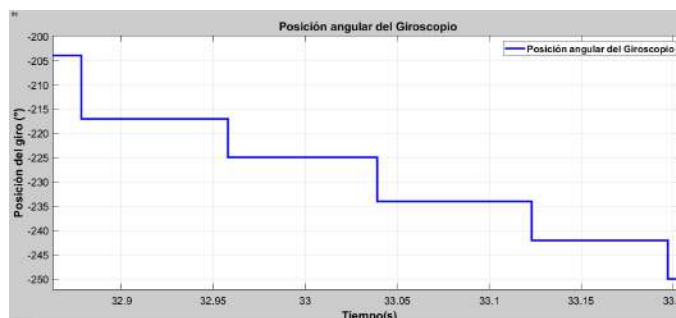


Figura 4.3: Modificación de parámetros.

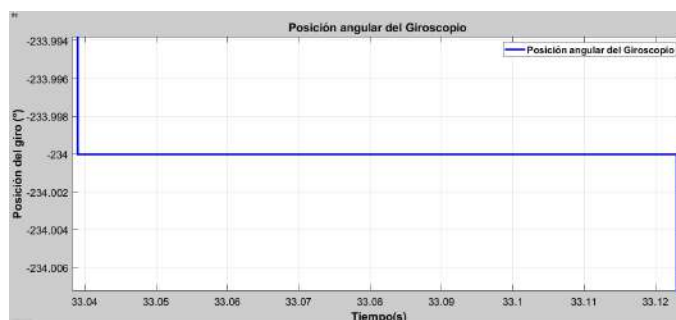
Una vez que se conecta la antena Bluetooth al Arduino Mega, se procede a simular en el Software Matlab Simulink, el cual genera un reporte de la generación del código. Al simular por un tiempo relativamente considerable se obtienen las siguientes gráficas que corresponden al estado estático y de movimiento del circuito de manera circular, dichas gráficas se visualizan en la Figura 4.4.



(a) Posición angular del Giroscopio



(b) Zoom para visualizar la frecuencia



(c) Valor de periodo = 84ms, frecuencia = 11.904Hz

Figura 4.4: Posición angular del sensor Giroscopio.

Como se observa en la Figura 4.4, el circuito tiene una buena respuesta para la lectura de datos, así como también el cambio del signo da a conocer si el robot va de un sentido a otro. Ahora bien, teniendo validado lo anterior, es necesario hacer el envío de datos del PWM para el control del motor, es por ello que se agregarán otros bloques al diagrama a bloques de la Figura 4.4, dichos bloques consisten en:

- * Un conmutador temporal para controlar el encendido o apagado por PWM.
- * Un reloj digital que lleve el conteo temporal del proceso.
- * Un codificador encargado de convertir el dato del valor de PWM en un conjunto de caracteres de 8 bits, que además se encarga de agregar un offset al PWM de 8 bits para

identificar el sentido de giro del motor. El código de codificación de datos es mostrado en el Apéndice F.

* Un bloque de transmisión de datos por el puerto 1.

Una vez que se han agregado los bloques para el PWM se procede a probar mediante una simulación, en la cual se consta el encendido del motor al inicio de la simulación, así como también el apagado del mismo al paso de 8 segundos.

Una vez realizadas las pruebas correspondientes del circuito junto con el entorno de Matlab-Simulink, es posible continuar la manufactura de los elementos del robot, los cuales han sido descritos en el capítulo del modelo matemático del robot.

4.3. Manufactura de elementos de PLA y acrílico

Durante el desarrollo del capítulo anterior se abordó el diseño del robot rueda en SolidWorks y en Adams View en donde se contemplaron los elementos necesarios para generar de manera exitosa el principio de funcionamiento, esto es debido a que los programas de simulación proporcionan los parámetros necesarios en actuadores para que se cumpla lo requerido. Sin embargo, en la vida real es necesario agregar los elementos que proporcionan dichas características en los actuadores, los cuales fueron descritos en la Sección 4.1, por lo que, en este capítulo se abordará la impresión de la carcasa, el péndulo, los engranes y demás elementos descritos en el capítulo anterior agregando además los elementos de acoplamiento, como la carcasa del circuito eléctrico principal y actuador para el robot rueda. Es de importancia señalar, que algunos elementos fueron modificados debido al espacio que ocupan los nuevos elementos, así como también, se modificó el paso diametral debido al desgaste que puede llegar a existir entre los engranes.

Los dibujos técnicos del robot que contemplan los cambios debido a la implementación se proporcionan en el Apéndice H.

Teniendo los diseños para la implementación es posible proceder a la manufactura de las placas de acrílico de la carcasa, de tal forma que se usa el software VISI Cad, en el cual se importa la pieza diseñada en SolidWorks en donde se predefinen las características de corte. Es de importancia señalar que las características de corte se definen acorde a la fresadora a utilizar y el diámetro de la herramienta de corte, en este caso se usó una fresadora router CNC.

Una vez realizadas las configuraciones es posible obtener el código necesario para poder realizar los cortes en la placa de acrílico.

Por otra parte, al tener los diseños para la implementación es posible proceder a la impresión de los elementos en una impresora MBot 3D cuyo software para controlarla es Mprint proporcionado por el mismo fabricante, por lo que se procede a imprimir cada uno de los elementos descritos por el Apéndice H.

En las Figuras 4.5 y 4.6 se muestran algunos de los componentes mecánicos del robot rueda.



(a) Discos base

(b) Pernos y tornillos

(c) Porta carbones

Figura 4.5: Elementos impresos.

(a) Ensamble péndulo, base pendulo, acrílico y PLA.
porta carbones y piñón.

Figura 4.6: Elementos impresos.

Una vez manufacturados los elementos es necesario ensamblarlos para comprobar las tolerancias, por lo que al ensamblar los elementos se obtiene el ensamble final como se muestra en la Figura 4.7.



Figura 4.7: Construcción de prototipo.

Al tener ya los elementos manufacturados es necesario pesarlos para saber sus masas ya que es de importancia para el modelo matemático del sistema real, así como también se pesaron los componentes complementarios de sistema como son los elementos electrónicos de control y los tornillos, por lo que al pesarlos se obtuvieron las masas mostradas en la Tabla 4.1.

| Masas de los elementos | |
|-----------------------------|----------|
| Elemento | Peso(Kg) |
| Carcasa Acrilico ensamblado | 1.255 |
| Soporte de pilas ensamblado | 0.029 |
| Carcasa con circuito | 0.066 |
| Disco base | 0.058 |
| Perno con tornillo | 0.010 |
| Esfera | 0.155 |
| Vástago de péndulo | 0.0238 |
| Base péndulo con motor | 0.02418 |
| Corona | 0.004 |
| Pila | 0.01618 |

Tabla 4.1: Masas de los elementos del Robot Rueda.

Capítulo 5

Modelo matemático de la planta real e implementación del algoritmo de control.

En este capítulo se abordará el cálculo de la función de transferencia del robot de forma experimental, así como también, se diseñará e implementará el algoritmo de control en el robot. Sin embargo, es necesario hacer algunos cambios en cuanto a la comunicación del microcontrolador con la PC. Uno de ellos es cambiar el Arduino Mega por una Raspberry Pi 3B, esto es debido a que la velocidad de procesamiento del Arduino es baja, su oscilador es de 16MHz mientras que en la Raspberry Pi 3B trabaja con un procesador de 1.4GHz quad-core, es decir su velocidad de procesamiento es aproximadamente 1000 veces mayor, es por ello que es viable el cambiar la tarjeta para la implementación. Es de importancia señalar que la instalación y configuración de las librerías de la tarjeta Raspberry pi se consiguen de manera similar a las librerías de la tarjeta Arduino Mega (tal y como se muestra en el apéndice D), con la diferencia que en el buscador de paquetes se coloca como filtro de búsqueda el nombre **Raspberry**.

Por todo lo anterior, se procede a la estimación de la función de transferencia y el control del sistema.

5.1. Modelo matemático

Una vez realizado el robot rueda es posible hallar su modelo matemático, para ello se hará uso de los parámetros del robot real para sustituirlos en el algoritmo creado para estimar las matrices del sistema linealizado (ver código 3.1) basado en las ecuaciones del robot virtual (diseñado en Solidworks y ADAMS view), y así, al agregar la fricción respecto a *theta* como variable y al hacer uso de la ecuación (3.13), el código modificado queda de la siguiente forma:

```
1 clear all
2 clc
3 syms S e mp lt rc b x1 x2 x3 x4 Ip g tau x2_p x4_p fr
4 g1=9.81
5 % masas
6 M_carcaza=1.255
```

```

7 M_perno=10e-3
8 M_pendolo=0.163066 %pendulo compuesto
9 M_corona=(4e-3)*2
10 M_base=(58e-3)*2
11 %Masas de elementos anadidos
12 M_pila=21.678896e-3
13 M_soporte=29e-3
14 M_circ=66e-3
15 %relaciones
16 R_carcaza= 78.9670294
17 R_perno= 53428.5714
18 R_corona= 5.903571161
19 R_base= 454.6951373
20 %Inercias
21 I_carcaza=M_carcaza/R_carcaza
22 I_perno=M_perno/R_perno
23 I_pendolo=417.6775369e-6 %pendulo compuesto
24 I_corona=M_corona/R_corona
25 I_base=M_base/R_base
26 %Inercias de los elementos anadidos
27 I_pila=M_pila/R_perno
28 I_soporte=23.14811e-6
29 I_circ=23.17425e-6
30 %logitudes
31 rc1=0.4/2
32 r_pinon=0.75e-2
33 r_corona=3.975e-2
34 lt1=0.144754
35 lq=0.0164
36 lp=0.032
37 b1=r_pinon/r_corona
38
39 %Valores a sustituir
40
41 e1=(M_carcaza + M_base + M_corona + M_pendolo + M_soporte + M_circ)*(rc1^2) +
      I_soporte + I_circ + I_carcaza + I_base + I_corona + 8*I_perno + (M_perno)
      *(8)*(lq^2 + rc1^2) + 8*I_pila + (M_pila)*(8)*(lp^2 + rc1^2)
42 mp1=M_pendolo
43 lt1=0.144754
44 rc1=0.4/2
45 b1=r_pinon/r_corona
46 Ip1=I_pendolo
47 g1=9.81
48 %valor de la friccion de contacto con el suelo
49 %fr=0.0124
50 %Se cambia fr de constante a variable
51 %coeficientes de ecuaciones para linealizacion
52 p= e - 2*mp*lt*rc*cos(b*x3 - x1) + Ip + mp*(lt^2)
53 q= mp*lt*rc*cos(b*x3 - x1)- Ip - mp*(lt^2)
54 r= mp*lt*rc*sin(b*x3 - x1)*((b*x4 - x2)^2)
55 t= q
56 u= Ip + mp*(lt^2)
57 s= mp*g*lt*sin(b*x3 - x1)

```



```

58 w= tau/b
59 f_r=fr*x2
60
61 % calculo del determinante para la linealizacion
62
63 ecua_deter=[p, q*b; q,u*b]
64 d_ec= det(ecua_deter)
65 d_ec_num=subs(d_ec,{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,0,0,0,e1,mp1,lt1,
    rc1,b1,Ip1,g1})
66
67 % declaracion de theta1 dos puntos y alfa dos puntos
68
69 x2_p=(1/d_ec_num)*[(r+s-f_r)*u*b - (q)*b*(w-s)]
70 x4_p=(1/d_ec_num)*[-1*t*(r+s-f_r) + (p)*(w-s)]
71
72 % proceso de linealizacion donde X1=theta1 y X2=theta1 punto, X3=alfa y
73 % X4=alfa punto. donde la diferencia de bx3-x1=0 bx4-x2=0
74
75 a11=subs(diff(x2,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
76 a12=subs(diff(x2,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
77 a13=subs(diff(x2,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
78 a14=subs(diff(x2,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
79 a21=subs(diff(x2_p,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
80 a22=subs(diff(x2_p,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
81 a23=subs(diff(x2_p,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
82 a24=subs(diff(x2_p,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
83 a31=subs(diff(x4,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
84 a32=subs(diff(x4,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
85 a33=subs(diff(x4,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
86 a34=subs(diff(x4,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,mp1
    ,lt1,rc1,b1,Ip1,g1})
87 a41=subs(diff(x4_p,x1),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
88 a42=subs(diff(x4_p,x2),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
89 a43=subs(diff(x4_p,x3),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
90 a44=subs(diff(x4_p,x4),{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,x2,0,x2/b1,e1,
    mp1,lt1,rc1,b1,Ip1,g1})
91 A=[a11, a12, a13, a14; a21, a22, a23, a24; a31, a32, a33, a34; a41, a42, a43,
    a44]
92 % A=subs(A,{x1,x2,x3,x4,e, mp, lt, rc, b, Ip, g},{0,0,0,0,e1,mp1,lt1,rc1,b1,Ip1

```

```

    ,g1}}
93 sistem_linealizado_en_B=[diff(x2,tau);diff(x2_p,tau);diff(x4,tau);diff(x4_p,
    tau)]
94 B=[subs(sistem_linealizado_en_B,{x1,x3,e,mp,lt,rc,b,Ip,g}},{0,0,e1,mp1,lt1
    ,rc1,b1,Ip1,g1})]
95
96 %busqueda del U_eq se iguala theta dos puntos y alfa dos punto a cero.
97
98 x2_psub=subs(x2_p,{x1,x2,x3,x4,mp,lt,rc,b,Ip,g}},{0,0,0,0,mp1,lt1,rc1,b1,
    Ip1,g1})
99 x2_psub1=subs(x2_p,{x1,x2,x3,x4},{0,0,0,0})
100 x4_psub=subs(x4_p,{x1,x2,x3,x4,e,mp,lt,rc,b,Ip,g}},{0,0,0,0,e1,mp1,lt1,
    rc1,b1,Ip1,g1})
101 x4_psub1=subs(x4_p,{x1,x2,x3,x4},{0,0,0,0})
102
103 % del desarrollo se tiene que U_eq=0 por lo tanto:
104
105 C=[diff(x1,x1)diff(x1,x2)diff(x1,x3)diff(x1,x4);diff(x2,x1)diff(x2,x2)diff
    (x2,x3)diff(x2,x4);diff(x3,x1)diff(x3,x2)diff(x3,x3)diff(x3,x4);diff(x4,
    x1)diff(x4,x2)diff(x4,x3)diff(x4,x4)]
106 D=[diff(0,tau);diff(0,tau);diff(0,tau);diff(0,tau)]
107
108 % [num1,den1]=ss2tf(A,B,C,D)
109 I=diag(diag(ones(size(A))))
110 G=C*inv(S*I-(A))*B + D
111
112 %Cada var-iesima representa los coeficientes de G(2) de la matriz G
113
114 var1=225582790042068964210478991041951110265402637325638507656888737461004129
    e58
115 var2=707179481507158549128113753131193380712354183422117595107721785
116 var3=707179481507158549128113753131193380712354183422117595107721785
117 var4=8076435299562453330589672786507105467087314172568968376363253760*fr
118 var5=45540035857331521598987586135597750547730092980755413265717133312
119 var6=487721676564820207312356110065285075606976920652343304243006930944*fr
120 var7=9894686710352577342889767531125203424375617748940420968943190016
121 var8=707179481507158549128113753131193380712354183422117595107721785
122 var9=43237554909407879044470388759040811517366607175784748300351076291
123 var10=707179481507158549128113753131193380712354183422117595107721785
124 var11=707179481507158549128113753131193380712354183422117595107721785
125 var12=8076435299562453330589672786507105467087314172568968376363253760*fr
126 var13=45540035857331521598987586135597750547730092980755413265717133312
127 var14=487721676564820207312356110065285075606976920652343304243006930944*fr
128
129 %Reconstruccion del elemento G(2) de G
130
131 G_eq2= var1/(var2*(var3*S^3 + var4*S^2 + var5*S + var6)) - (var7*(var8*S^2 +
    var9))/(var10*(var11*S^3 + var12*S^2 + var13*S + var14))
132
133 vq1=((var1)/(var2*var3))/(S^3 + (var4/var3)*S^2 + (var5/var3)*S + (var6/var3))
    - (((var7*var8)/(var10*var11))*(S^2 + (var9/var8)))/(S^3 + (var12/var11)*S^2
    + (var13/var11)*S + (var14/var11))
134

```

```

135 vq2=(4.5107e+03)/(S^3 + 11.4206*fr*S^2 + 64.3967*S + 689.6717*fr) - ((13.9918)
      *(S^2 + 61.1409))/(S^3 + 11.4206*fr*S^2 + 64.3967*S + 689.6717*fr)
136
137 vq3=(-13.9918*S^2 + 3.6552e+03)/(S^3 + 11.4206*fr*S^2 + 64.3967*S + 689.6717*fr
      )
138
139 num=[-13.9918 0 3.6552e+03]
140 den0=[1 11.4206*fr 64.3967 689.6717*fr ]

```

Por lo que, del código anterior se obtiene que la función de transferencia está descrita por la ecuación:

$$G(s, fr) = \frac{-13,9918s^2 + 3,6552e + 03}{s^3 + 11,4206 * fr * s^2 + 64,3967s + 689,6717 * fr} \quad (5.1)$$

Ahora bien, dada la función de transferencia en términos de la fricción, es posible hallar dicha fricción(fr), mediante una comparación con la respuesta del sistema real, de modo que, se cambia el valor de la fricción hasta hallar la más óptima para que las diferencias de respuestas sean mínimas. Para realizar la aproximación se siguen los pasos mostrados a continuación:

1. Se somete el robot a una entrada máxima constante de PWM, en donde es de interés saber la velocidad que alcanza. Dicho PWM equivale a dar toda la potencia del motor, es decir, el par torsional máximo.
2. Estimar el par torsional máximo que el sistema es capaz de proporcionar.
3. Se simula la ecuación (5.4) haciendo $fr = 0$, dando como entrada el par torsional $0,091968N.m$ que es el máximo que el sistema puede proporcionar.
4. Al haber hallado el valor de la velocidad del sistema sin fricción se propone un valor de fricción que no sea mayor a la unidad y se compara con la respuesta del sistema, de allí se toma la decisión de si es necesario corregir el coeficiente de fricción
5. Iterar en la variación del coeficiente de fricción hasta que la función de transferencia (ecuación (5.4) llegue al mismo valor de estabilización que la planta física a PWM máximo.

5.1.1. Respuesta del robot con PWM máximo de entrada

Para encontrar la respuesta del sistema ante un PWM máximo, se procede a crear el diagrama a bloques en Matlab dado por la Figura 5.1 en la cual se pueden observar que el robot se estabiliza a una velocidad de $150^\circ/s$.

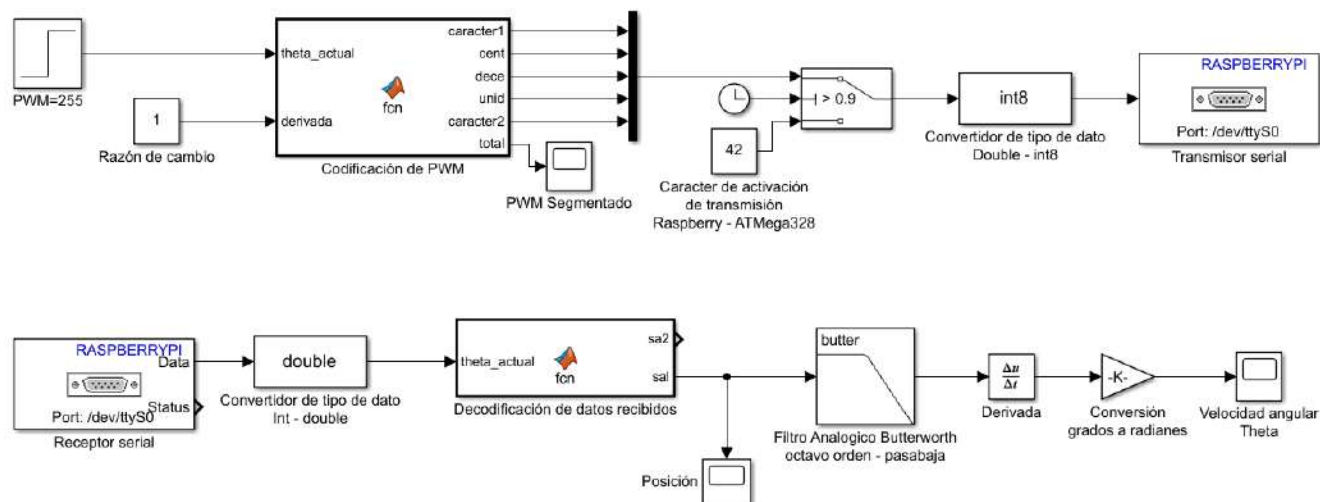


Figura 5.1: Diagrama a bloques de la comunicación entre Raspberry Pi y microcontrolador ATmega328.

La descripción de cada bloque de la Figura 5.1 se muestra a continuación:

1. **PWM=255**: Realiza la tarea de enviar una entrada escalón con el máximo PWM, la entrada escalón se activa en pasando el primer segundo.
2. **Razón de cambio**: Tiene la función de dar énfasis si la señal de control incrementa o decrementa (pendiente)
3. **Codificación de PWM**: Realiza la función de descomponer el número de PWM en un conjunto de caracteres ascii con la finalidad de ser enviados por medio del puerto USART en tramas de 8 bits, cada trama de 8 bits consta de un caracter de inicio de trama así como también uno de fin de trama. (ver código en Apéndice F).
4. **PWM Segmentado**: Visualización del PWM de salida codificado.
5. **Carácter de activación**: Tiene la función de enviar el carácter de activación para la recepción y envío de datos del microcontrolador.
6. **Convertidor de tipo de dato**: Realiza la tarea de convertir un tipo de dato a otro con la final de poder ser interpretado ya sea por la Raspberry o por el microcontrolador.
7. **Transmisor Serial**: Tiene la labor de enviar cada dato de 8 bits por el puerto serie.
8. **Receptor Serial**: Tiene la labor de recibir cada dato de 8 bits por el puerto serie.
9. **Decodificación de PWM**: Interpreta cada valor ascii recibido, que forma parte de cada dígito del número a ser interpretado, así como también, se identifican los caracteres de inicio y fin de trama de datos (ver código en Apéndice E).

10. **Filtro analógico Butterworth:** Filtra la señal digital ante cambios bruscos por ruido, así como también, tiene la función transformar en señal analógica la señal de entrada, para así poder ser derivada y estimar la velocidad del robot.
11. **Convertidor de grados a radianes:** Convertir los grados a radianes puesto que el microcontrolador manda información de la posición angular en grados y el control trabaja con radianes.

Al ejecutar el diagrama a bloques se obtienen las posiciones dadas por la Figura 5.2, en donde se puede apreciar que el robot responde de manera similar ante las tres pruebas experimentales, sin embargo, hay un aspecto importante a señalar, es la presencia de perturbaciones periódicas en cada una de las pruebas experimentales. Además, como se observa, puede suceder que no siempre aparezcan las perturbaciones en todos los experimentos.

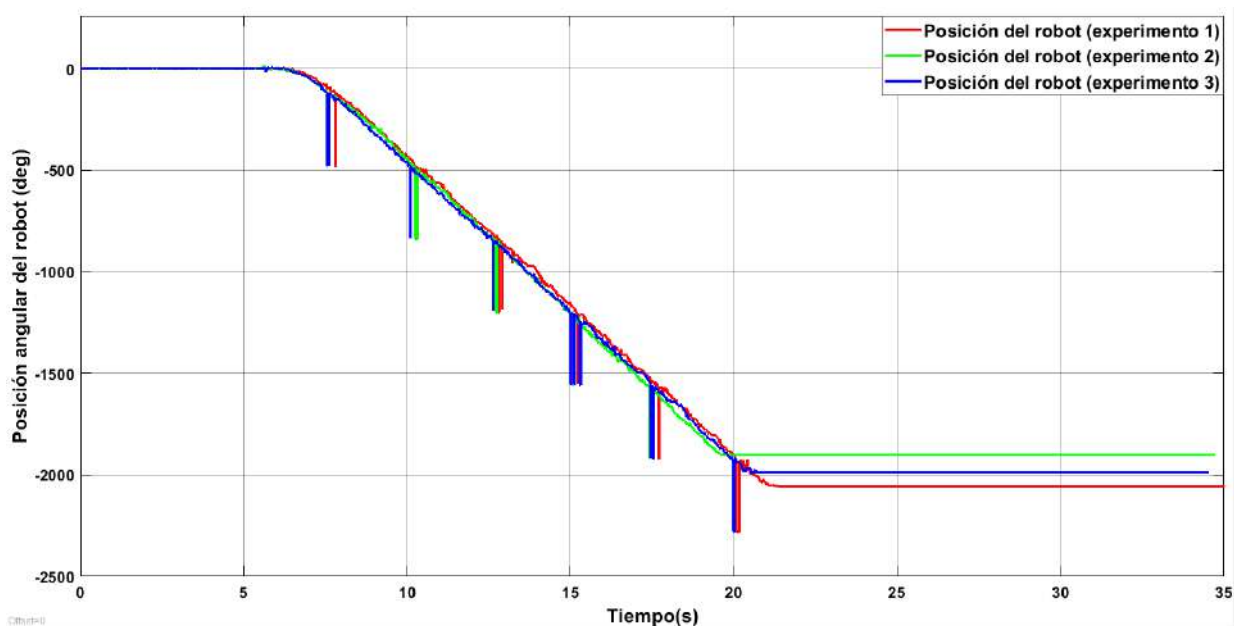


Figura 5.2: Posición del robot rueda ante una entrada de 255 de PWM.

En cuanto a la velocidad que adquiere el robot, se tiene la Figura 5.3, en la cual se puede apreciar que el sistema llega a adquirir una velocidad de $150^\circ/s$ por lo que, ese valor representa la velocidad máxima que puede alcanzar el robot rueda. Como observación a las gráficas de velocidad, se tiene que las perturbaciones de ruido en la posición sin filtrar producen las oscilaciones alrededor de la referencia.

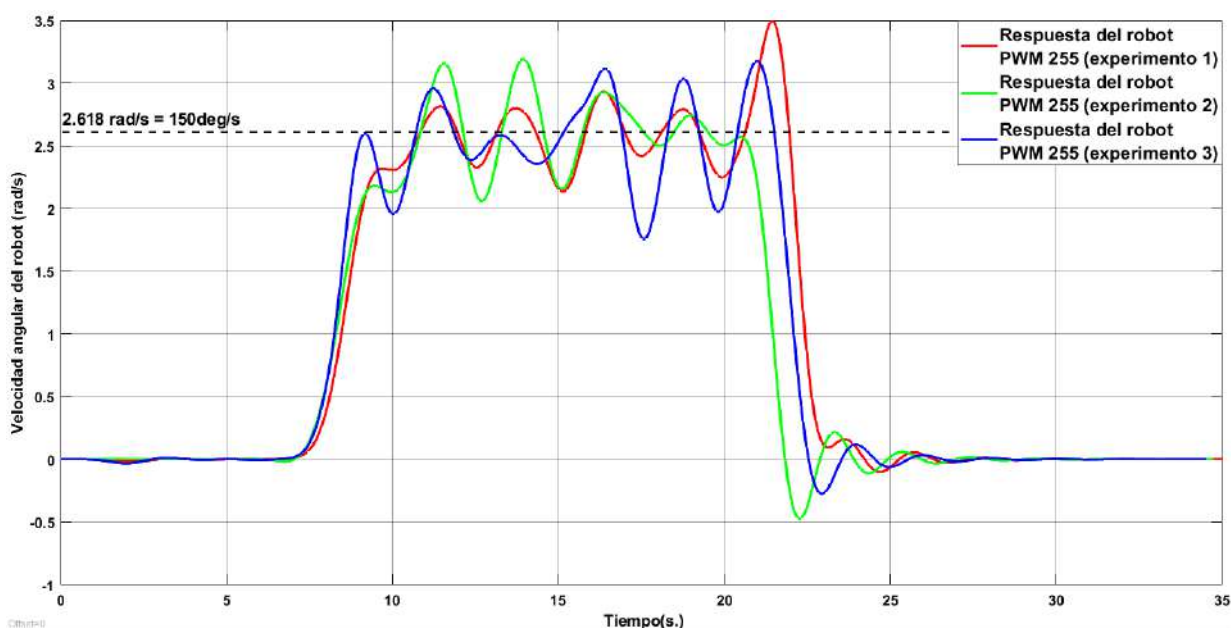


Figura 5.3: Velocidad del robot rueda ante una entrada de 255 de PWM.

5.1.2. Estimación del par torsional máximo del sistema

La estimación del par torsional máximo se puede realizar mediante la excitación del motor con determinado PWM, en donde el motor permanece con la flecha detenida y se sensa la cantidad de corriente que pasa por él.

Conociendo el valor de corriente que pasa a través del motor se procede a consultar con los datos del fabricante la relación de par torsional máximo con la corriente de consumo, por lo que se obtuvo que el motor proporciona $T_{max} = 3kg * cm$ con una corriente de consumo de 1,6A. Por lo tanto la constante de par torsional está definida por:

$$K_t = \frac{(3kg * cm) * (9,81m/s^2)}{(100cm)(1,6A)} = 0,1839375N * m/A \quad (5.2)$$

Por lo tanto, el par torsional en función de la corriente queda definida por:

$$T_{salida} = 0,1839375 * i(Nm) \quad (5.3)$$

donde i es la corriente que circula por el motor.

Al realizar pruebas con variación de PWM se obtienen los resultados dados por la tabla: Haciendo referencia a la Tabla 5.1, el par torsional máximo que puede proporcionar el sistema es $0,0938N * m$

5.1.3. Simulación de la ecuación de fricción indefinida (fr)

Para graficar la ecuación se propone el siguiente diagrama a bloques con la función de transferencia:

| Prueba | PWM | Corriente | Par torsional |
|--------|-----|-----------|---------------|
| 1 | 0 | 0 | 0 |
| 2 | 50 | 0.1 | 0.01833 |
| 3 | 100 | 0.22 | 0.04046 |
| 4 | 150 | 0.31 | 0.05702 |
| 5 | 200 | 0.45 | 0.08277 |
| 6 | 250 | 0.5 | 0.09196 |
| 7 | 255 | 0.51 | 0.09380 |

Tabla 5.1: Relaciones de Par-PWM

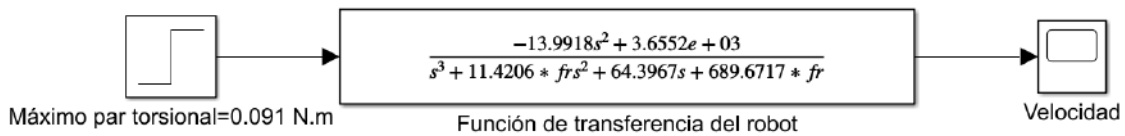


Figura 5.4: Diagrama a bloques para el cálculo de fr del sistema.

Al realizar la primera simulación $fr=0$, se obtiene la gráfica de velocidad mostrada por la Figura 5.5 , en la cual es evidente que si no hay fuerza de fricción en el robot, la velocidad crecerá de manera indefinida en el tiempo.

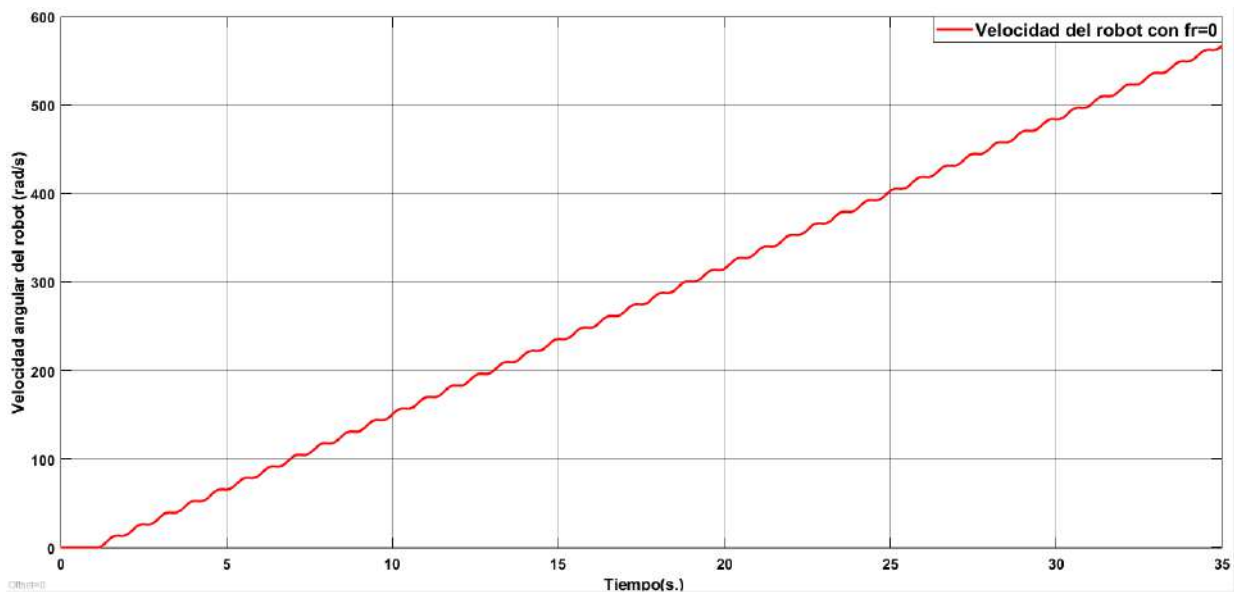


Figura 5.5: respuesta de velocidad del robot $fr=0$.

Haciendo iteraciones para incrementar el coeficiente de fricción de tal forma que, se aproxime la respuesta de velocidad a la del sistema real. Siendo esto así, se obtiene que $FR = 0,185$, cuya respuesta de velocidad es mostrada en la Figura 5.6.

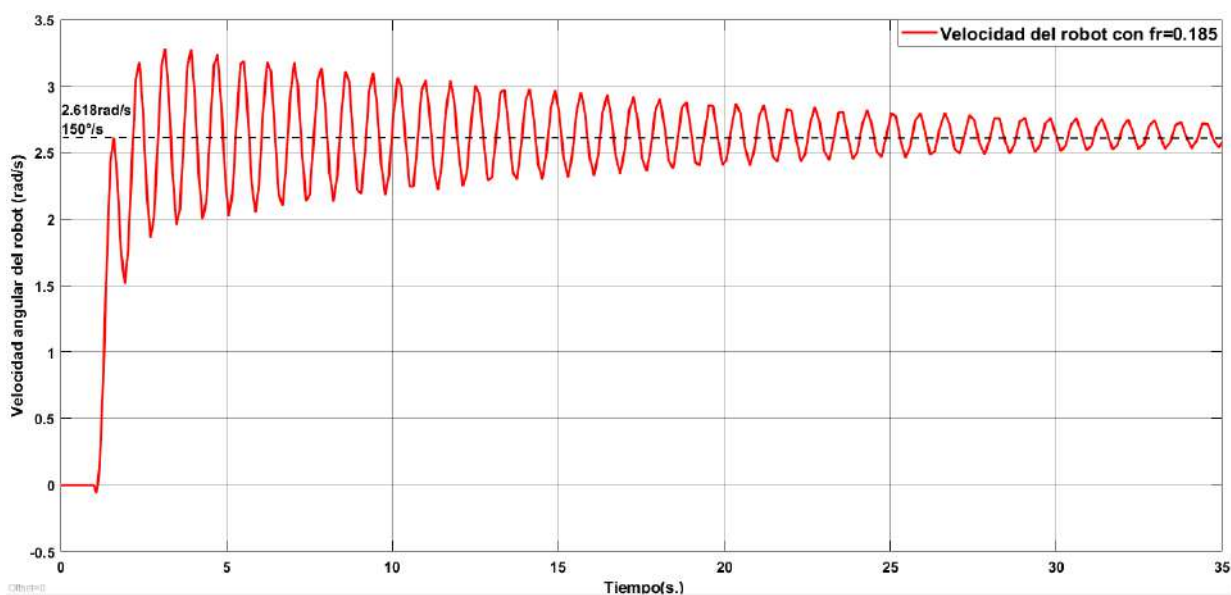


Figura 5.6: respuesta de velocidad del robot $fr=0.185$.

Para hacer una comparación más acertada se tiene que hacer un desplazamiento en el tiempo de la respuesta de la función de transferencia, esto es debido a que hay un offset causado por el filtro. Dicha comparativa es mostrada por la Figura 5.7, de la cual se puede apreciar que las respuestas son casi idénticas, sin embargo, se puede observar el retraso de la señal de velocidad del robot, es decir tarda en responder, esto puede ser causado por problemas mecánicos en el robot, por la saturación de datos en el buffer del microcontrolador o dinámicas no modeladas.

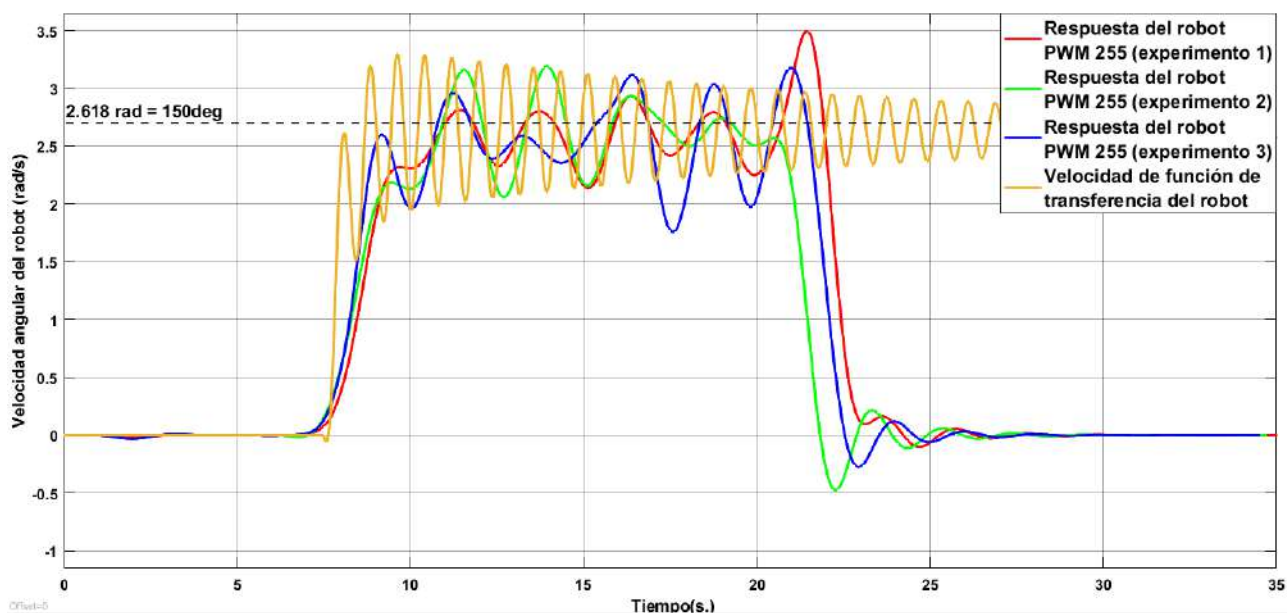


Figura 5.7: Comparación de la respuestas de la función de transferencia y el sistema físico.

Una vez expuesto lo anterior, se puede decir que la constante de fricción hallada posibilita una aproximación certera al físico, quedando la función de transferencia denotada por la ecuación siguiente:

$$G(s) = \frac{-13,9918s^2 + 3,6552e + 03}{s^3 + 2,112811 * s^2 + 64,3967s + 127,589264} \quad (5.4)$$

5.2. Diseño de control del sistema físico

En esta sección se abordará la implementación del control del sistema mediante el uso de la función de transferencia del sistema físico. El control a implementar será diseñado mediante el Lugar Geométrico de las Raíces, es por ello que se hará un análisis similar al del Capítulo 4. Nota: A partir de ahora se denotará a la función de transferencia de la planta real(robot implentado) como simplemente "**sistema real**" y la función de transferencia del sistema diseñado en SolidWorks se denominará "**sistema virtual**".

Para saber el estado de control del sistema es de importancia ver el mapa de polos y ceros en lazo cerrado, tal y como se muestra en la Figura 5.8, en donde se logra apreciar que el modelo real presenta una simetría idéntica a la del sistema virtual (prototipo de robot en simulación). Ahora bien, dados los argumentos anteriores se puede proceder con pasos similares para el diseño del algoritmo de control.

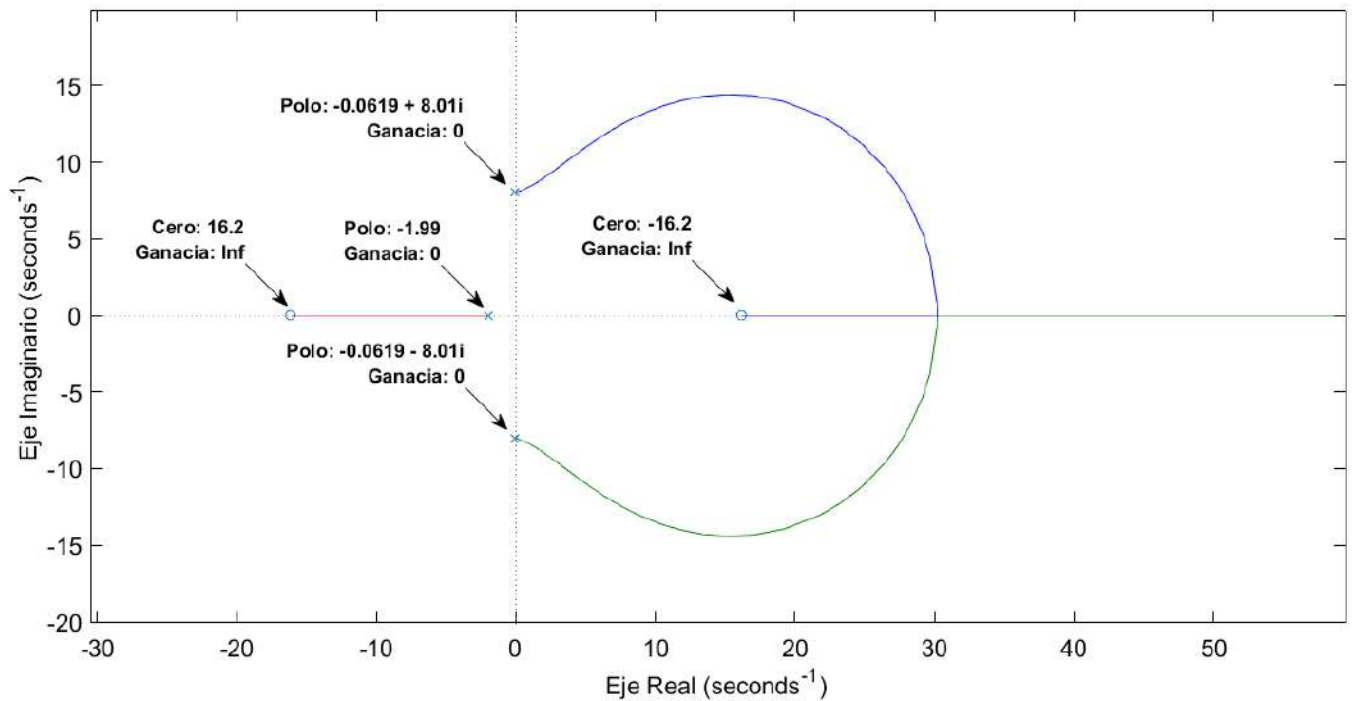


Figura 5.8: Lugar geométrico de las raíces del sistema real.

Las características deseadas tendrán el objetivo de mejorar la repuesta del sistema real,

basándose en las características actuales del mismo. Por el polinomio característico dado por la ecuación (5.5) se obtiene que la frecuencia de oscilación es $8,009rad/s$ y el amortiguamiento es $0,00773$, por lo que, se proponen las mismas características para el sistema virtual, el coeficiente de amortiguamiento $\zeta = 0,5$ y la frecuencia de oscilación $\omega_n = 1rad/s$, por lo que, los polos dominantes están definidos por $s_{1,2} = \zeta\omega_n \pm \omega_n\sqrt{1 - \zeta^2}i$, de modo que los polos deseados son $s_{1,2} = -0,5 \pm 0,866i$.

$$Pol_{caract} = s^3 + 2,112811s^2 + 64,3967s + 127,589264 = (s + 1,988912)(s^2 + 0,1238988s + 64,15027) \quad (5.5)$$

Por lo analizado en el Capítulo 4, se tiene que es de importancia eliminar el par de polos conjugados y el cero de la planta en L.A, es decir agregar un compensador directo con la planta ($G(s)_{supr}$), por lo que la función de transferencia queda de la siguiente manera:

$$\begin{aligned} G(s) * G(s)_{supr} &= \frac{-13,9918s^2 + 3655}{s^3 + 2,112811s^2 + 64,3967s + 127,589264} * \left(\frac{s^2 + 0,1238988s + 64,15027}{(s + 16,1628811)(s)} \right) \\ &= \frac{-32,3079(s + 19,088454)(s - 19,088454)}{(s + 0,288)(s^2 + 0,01503027s + 95,49204)} * \left(\frac{s^2 + 0,01503027s + 95,49204}{(s + 19,088454)(s)} \right) \\ &= \frac{-13,99(s - 16,1628811)}{(s + 1,9889)(s)} \end{aligned} \quad (5.6)$$

Ahora bien, es importante notar que la planta queda simplificada, sin embargo, no simplemente se le pueden hacer cambios a la función de transferencia, sino que se tiene que compensar ese cambio con la adición un compensador de adelanto que cumpla con los requerimientos del funcionamiento de la planta, de modo que, el compensador tiene que compensar el ángulo de deficiencia dada por los polos conjugados deseados en L.C de la planta en L.A.

La ecuación (5.7) cuantifica el ángulo compensar.

$$\begin{aligned} Angulo_{comp} &= 180^\circ + AngCeros_{s=-0,5+0,866i} - AngPolos_{s=-0,5+0,866i} \\ &= -180^\circ + 2,9751748^\circ + 30,1846^\circ + 120^\circ = -26,8402252^\circ \end{aligned} \quad (5.7)$$

Sabiendo el ángulo a compensar, se puede proceder a ubicar el polo y el cero que cancelan la deficiencia de ángulo, para ello se estable el valor del polo en 1, como se muestra en la Figura 5.9 en donde la ubicación del polo genera una deficiencia de ángulo de 60° , de modo solo faltaría agregar el angulo del cero el cual es $-60 + 26,84020 = 33,1597$, de modo que, la posición del cero es calculada por:

$$P_{cero} = 0,5 + \frac{0,866025}{\tan(33,1597)} = 1,8254547 \quad (5.8)$$

Como se muestra en la Figura 5.9 en donde el polo está dado en $-1,65520103$ y el cero en $-0,604129$, siendo esto así, la función de transferencia del sistema compensado en L.A queda dado por la ecuación (5.9).

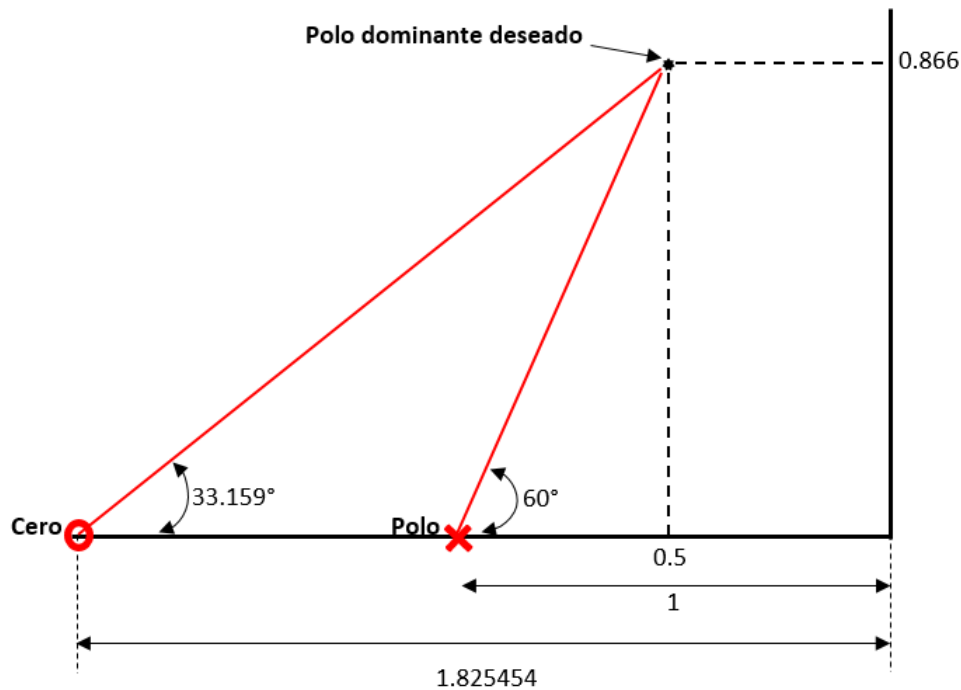


Figura 5.9: Esquema de localización del polo y cero del compensador de adelanto.

$$G_T = G(s) * G(s)_{comp} = \frac{-13,99(s - 16,1628811)}{(s + 1,9889)(s)} * \frac{K(s + 1,8254547)}{s + 1} \quad (5.9)$$

Ahora, solo falta calcular la ganancia \mathbf{K} del sistema para que tenga los polos dominantes en el lugar deseado, por lo que se debe de cumplir la condición de magnitud y ángulo, en donde la magnitud debe ser 1 y ángulo de 180° en el polo deseado. La compensación de ángulo está dada por el polo y el cero del compensador con respecto a los polo dominantes, por otro lado, la condición de magnitud se puede encontrar al sustituir el polo dominante deseado $s = -0,5 + 0,866i$ en la función de transferencia (5.9) e igualar a -1.

$$\begin{aligned} \frac{-13,99(-0,5 + 0,866i - 16,1628811)}{(-0,5 + 0,866i + 1,9889)(-0,5 + 0,866i)} * \frac{K(-0,5 + 0,866i + 1,8254547)}{-0,5 + 0,866i + 1} &= -1 \\ (-214,57063 - 2,6 * 10^{-6}i)K &= -1 \\ 214,57K_{\angle -179,97^\circ} &= -1 \\ K &= |4,66046e - 03| \end{aligned} \quad (5.10)$$

Al tener la ganancia \mathbf{K} , dada por la ecuación (5.10), es posible establecer la función de transferencia en lazo abierto como se muestra en la ecuación (5.11).

$$G_T = \frac{-13,99(s - 16,1628811)}{(s + 1,9889)(s)} * \frac{4,66046e - 03(s + 1,8254547)}{s + 1} \quad (5.11)$$

Para probar si el diseño cumple con las especificaciones, se procede a cerrar el lazo y se grafica su lugar geométrico de las raíces, como se muestra en las Figuras 5.10 y 5.11.

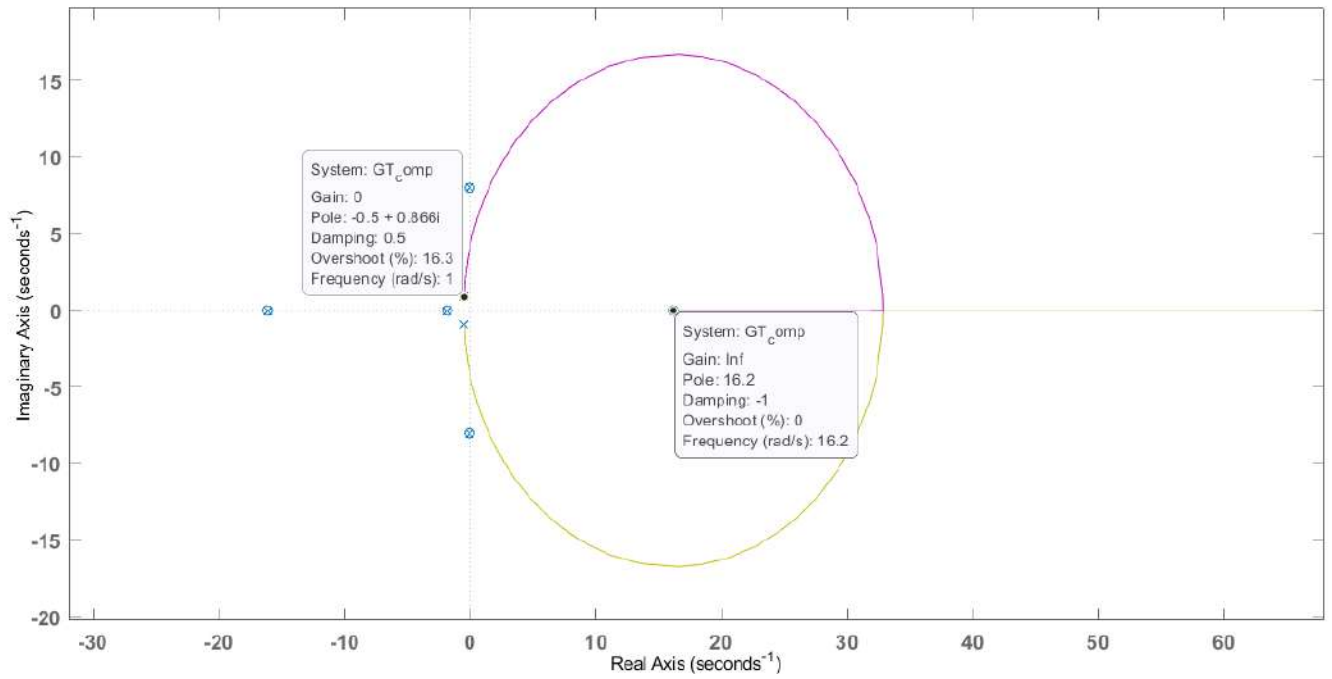


Figura 5.10: Lugar geométrico de las raíces del sistema compensado.

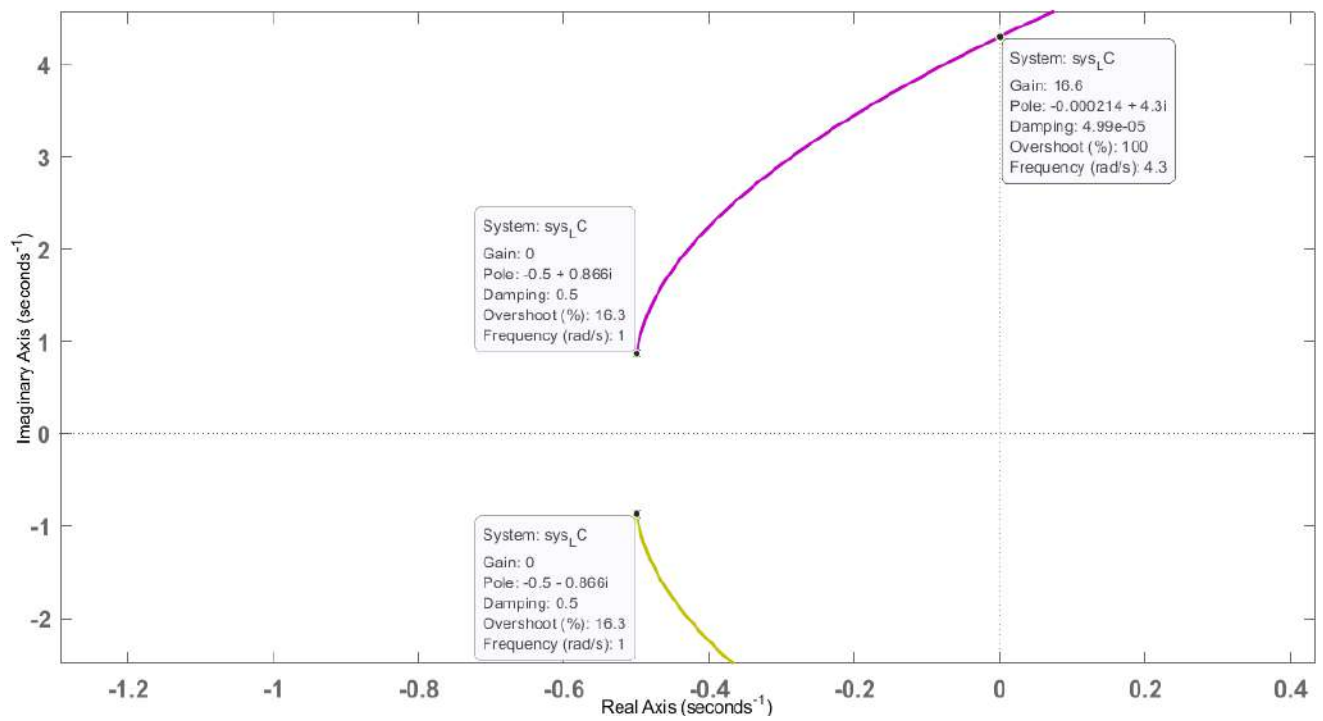


Figura 5.11: Lugar geométrico de las raíces del sistema compensado ampliado.

Debido a que el algoritmo de control aplicar es a un sistema físico en donde existe la posibilidad de no tener la dinámica completamente modelada, es necesario analizar cual es el margen de ganancia y fase que tiene el sistema para poder sintonizar, de tal forma que se hace uso de los diagramas de bode para visualizar dichos margenes.

La figura 5.12 muestra los margenes de fase y ganancia en función de la frecuencia en donde se observa que el margen de ganancia es de $24,4\text{dB}$ y el de de frecuencia es de $86,1^\circ$, para comprobar si los margenes son correctos es necesario convertir el margen de ganancia en decibelios del diagrama de bode a ganancia en el LGR en LC, esto es expresado por la ecuación:

$$\begin{aligned} K(\text{dB}) &= 20 \log(K) \\ 24,4\text{dB} &= 20 \log(K) \\ \frac{24,4}{20} &= \log(K) \\ 10^{1,22} &= K \\ K &= 16,5958 \end{aligned} \quad (5.12)$$

Como se puede apreciar en la ecuación (5.12) el margen de ganancia convertido del diagrama de bode a LGR son la misma, esto es al ver la ganancia critica del LGR de la figura 5.11 en donde conuerda la ganancia y la frecuencia.

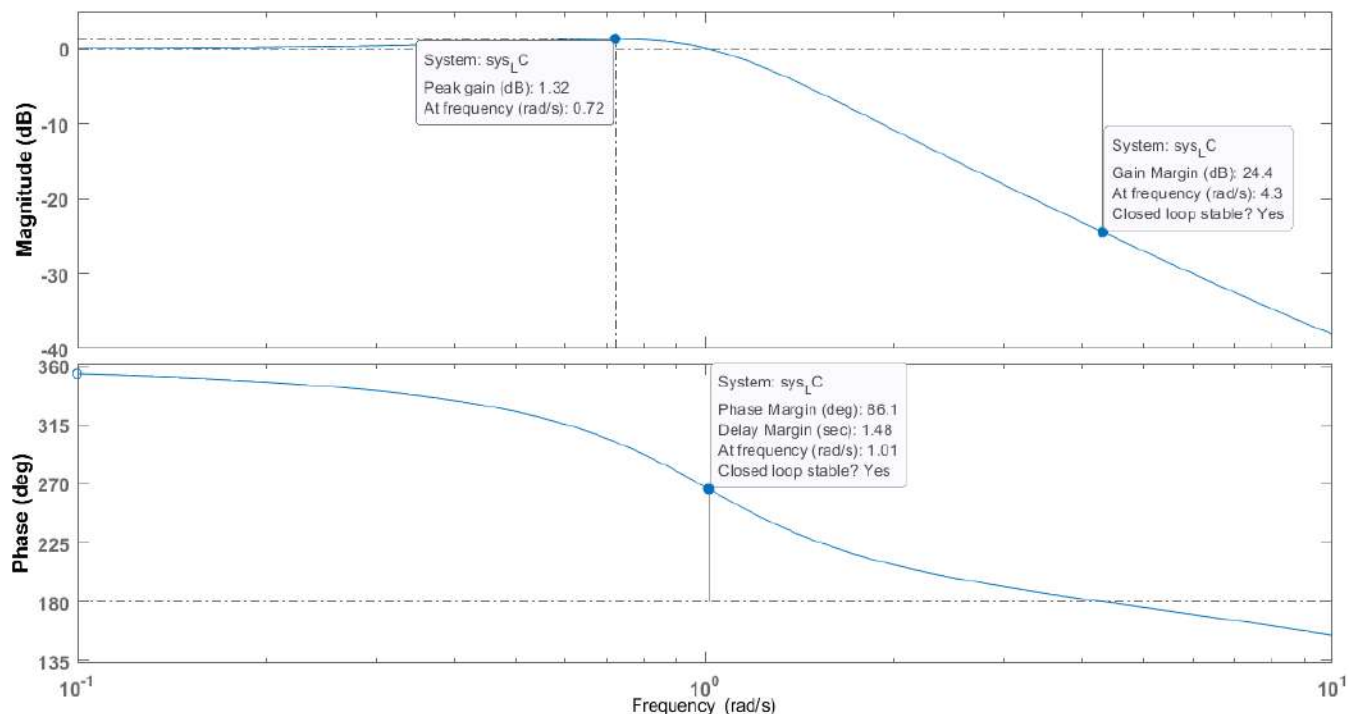


Figura 5.12: Diagrama de bode del sistema compensado en LC.

Al cumplir los requerimientos del sistema mediante el compensador en el LGR en LC y al tener identificado el margen de ganancia, se continuará con la simulación en Matlab simulink (ver Figura 5.13) para analizar la respuesta del sistema.

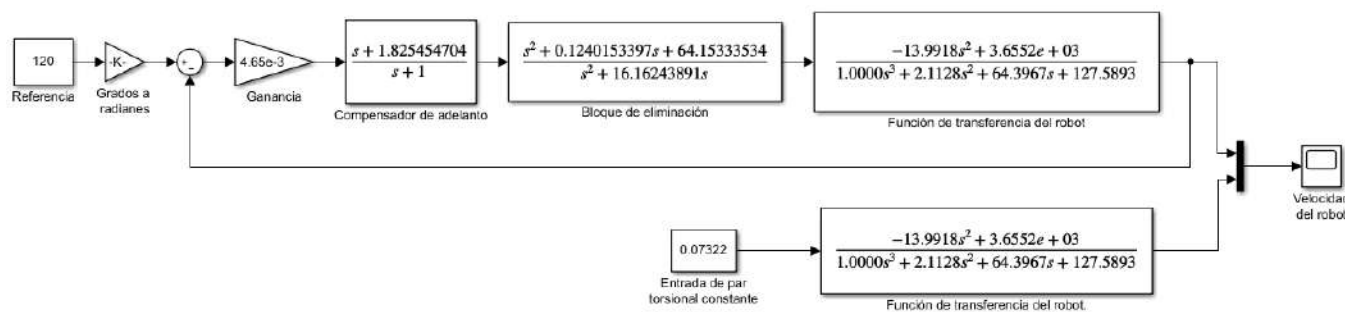


Figura 5.13: Diagrama a bloques del sistema real controlado.

El resultado del control se muestra en la Figura 5.14 en donde se comparan las respuestas del sistema tanto en lazo abierto con par torsional constante y lazo cerrado con control, de modo que se puede observar que el control tiene una gran eficiencia, basada en los parámetros de sobre elongación (1.25) y tiempo de asentamiento (5% de tolerancia del valor final de asentamiento). En cuanto al tiempo de asentamiento, que es prácticamente 12 veces más rápido que el sistema con par torsional constante.

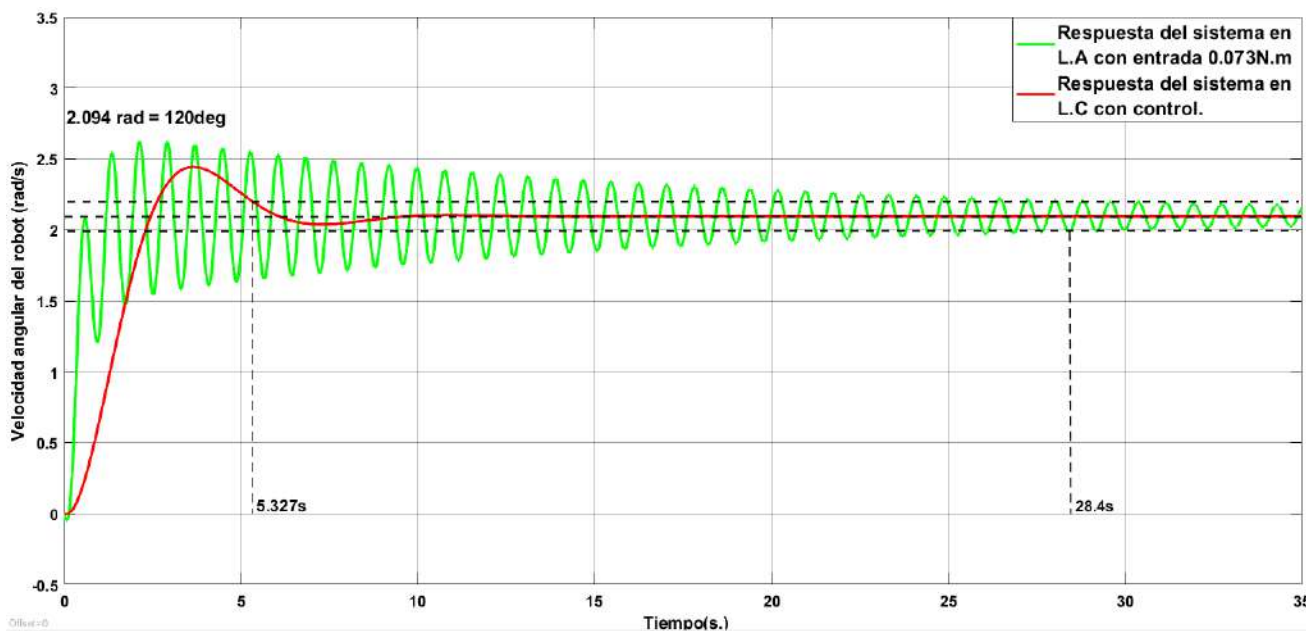


Figura 5.14: Comparación de las respuestas del sistema con control y en L.A.

Por todo lo anterior, se puede decir que el algoritmo de control diseñado cumple con lo requerido para que el sistema funcione adecuadamente.

5.3. Pruebas del control en el sistema físico

Para la implementación del control externo en Matlab, se harán cambios en el diagrama a bloques mostrado en la Figura 5.1, dando como resultado el diagrama a bloques mostrado en la Figura 5.15.

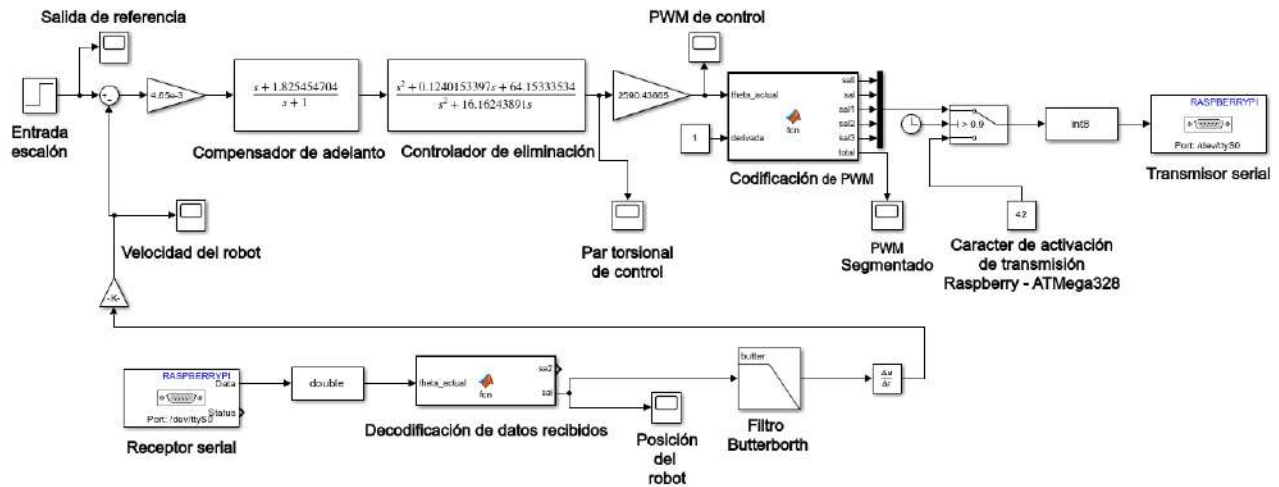


Figura 5.15: Diagrama a bloques del algoritmo de control del sistema real.

Es de importancia mencionar los aspectos que fueron cambiados, y dar a conocer su funcionamiento:

1. **Entrada escalón:** Realiza la tarea de enviar una entrada escalón con el valor de la velocidad de referencia en radianes, la entrada escalón se activa pasando el primer segundo.
2. **Salida de referencia:** Permite observar la velocidad de referencia.
3. **Compensador de adelanto:** Compensa todos los cambios generador por el **controlador de eliminación** y posiciona de manera adecuada los polos conjugados deseados.
4. **Compensador de eliminación:** Cancela los polos conjugados y el cero conjugado negativo de la planta, esto es con el fin de mejorar la estabilidad de la planta.
5. **Par torsional de control:** Muestra el par torsional requerido para el control de la planta.
6. **PWM de control:** Muestra el PWM requerido para el accionamiento del motor en función del par torsional requerido, el factor de relación de par torsional y PWM surge de la relación lineal de la 5.1.
7. **Codificación de PWM:** Realiza la función de descomponer el número de PWM en un conjunto de caracteres ascii con la finalidad de ser enviados por medio del puerto USART, en tramas de 8 bits, cada trama de 8 bits consta de un carácter de inicio de

trama, así como también, uno de fin de trama, a la vez, este bloque tiene la función de truncar el valor de PWM excedente a 255, dando un valor de salida de 255, y en caso de ser inferior a -255 se trunca a un valor de -255 . (ver código en Apéndice F).

8. **PWM Segmentado:** Visualización del PWM de salida codificado.
9. **Caracter de activación:** Tiene la función de enviar el carácter de activación para la recepción y envío de datos del microcontrolador.
10. **Convertidor de tipo de dato:** Realiza la tarea de convertir un tipo de dato a otro con la finalidad de poder ser interpretado ya sea por la Raspberry o el microcontrolador.
11. **Transmisor Serial:** Tiene la labor de enviar cada dato de 8 bits por el puerto serie.
12. **Receptor Serial:** Tiene la labor de recibir cada dato de 8 bits por el puerto serie.
13. **Decodificación de PWM:** Interpreta cada valor ascii recibido, que forma parte de cada dígito del número a ser interpretado, así como también, se identifican los caracteres de inicio y fin de trama de datos (ver código en Apéndice E).
14. **Posición del robot:** Muestra la posición en la que se encuentra el robot.
15. **Filtro analógico Butterworth:** Filtra la señal digital ante cambios bruscos por ruido, así como también, tiene la función transformar en señal digital de entrada en señal analógica, para así poder ser derivada y estimar la velocidad del robot.
16. **Velocidad del robot:** Muestra la velocidad que adquiere el robot.

Una vez expuesto el funcionamiento de cada bloque del sistema de control, se procede a evaluar el control del sistema con una velocidad de referencia de $100^\circ/s = 1,745rad/s$, dando como resultado las siguientes gráficas:

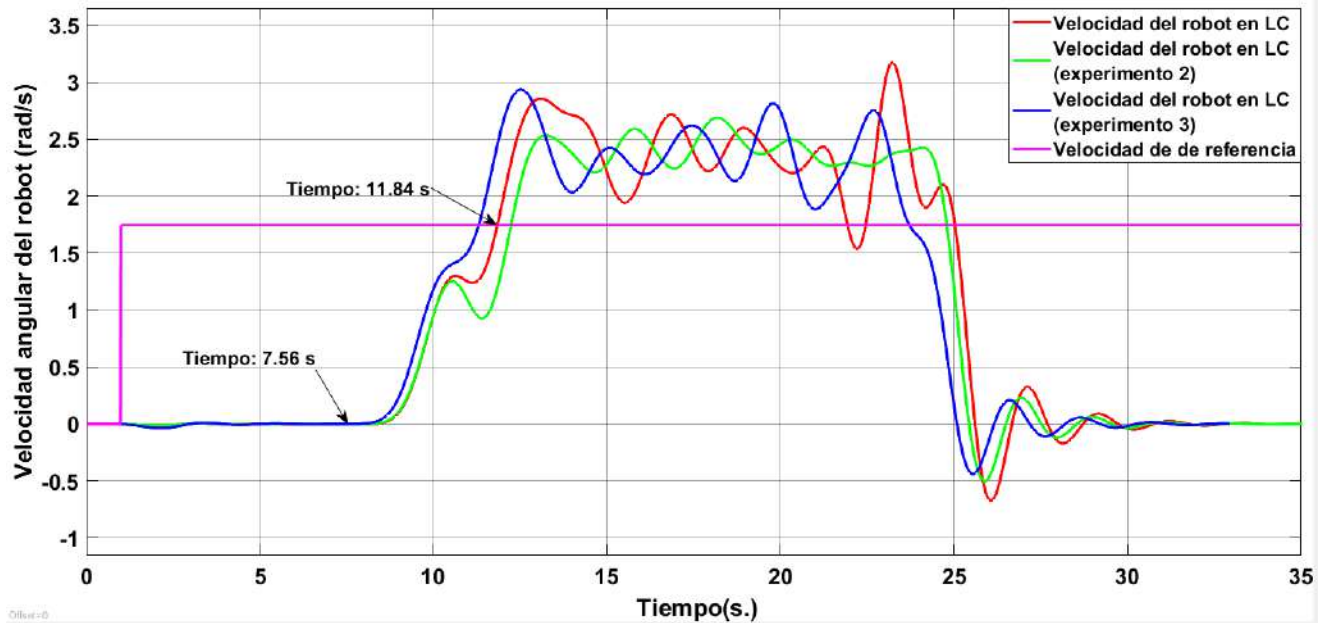


Figura 5.16: Respuesta en velocidad del sistema al algoritmo de control.

Al correr la simulación se obtienen las siguientes respuestas:

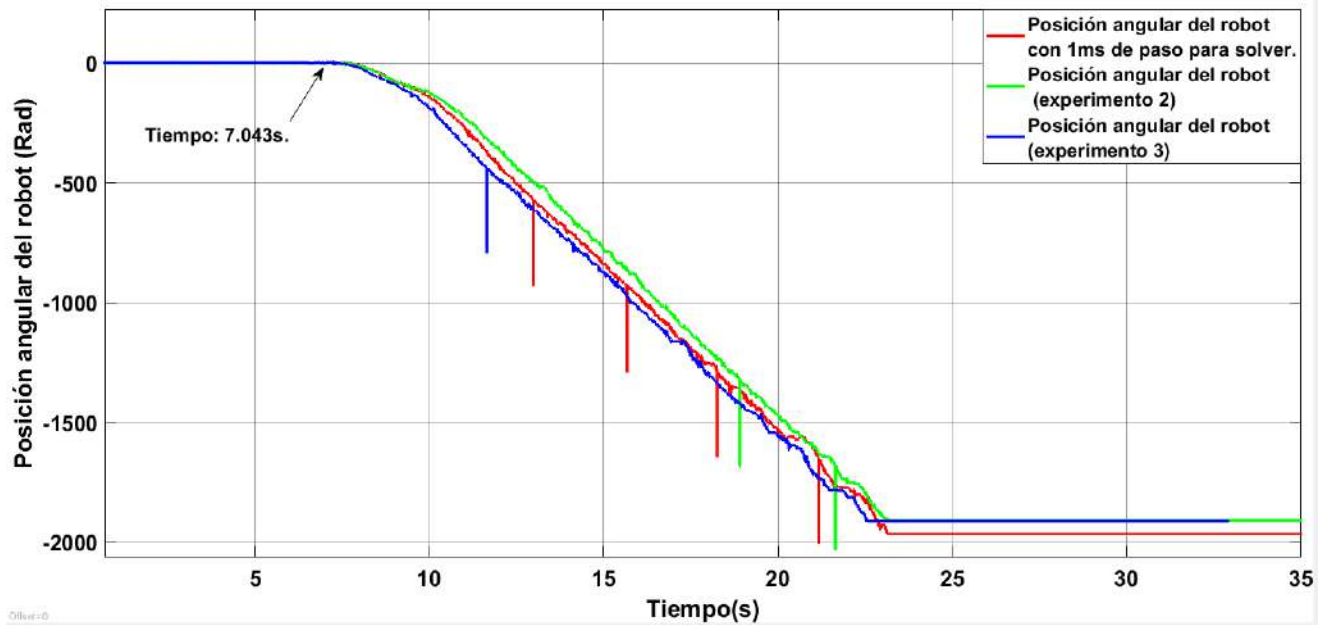


Figura 5.17: Respuesta en posición del sistema al algoritmo de control.

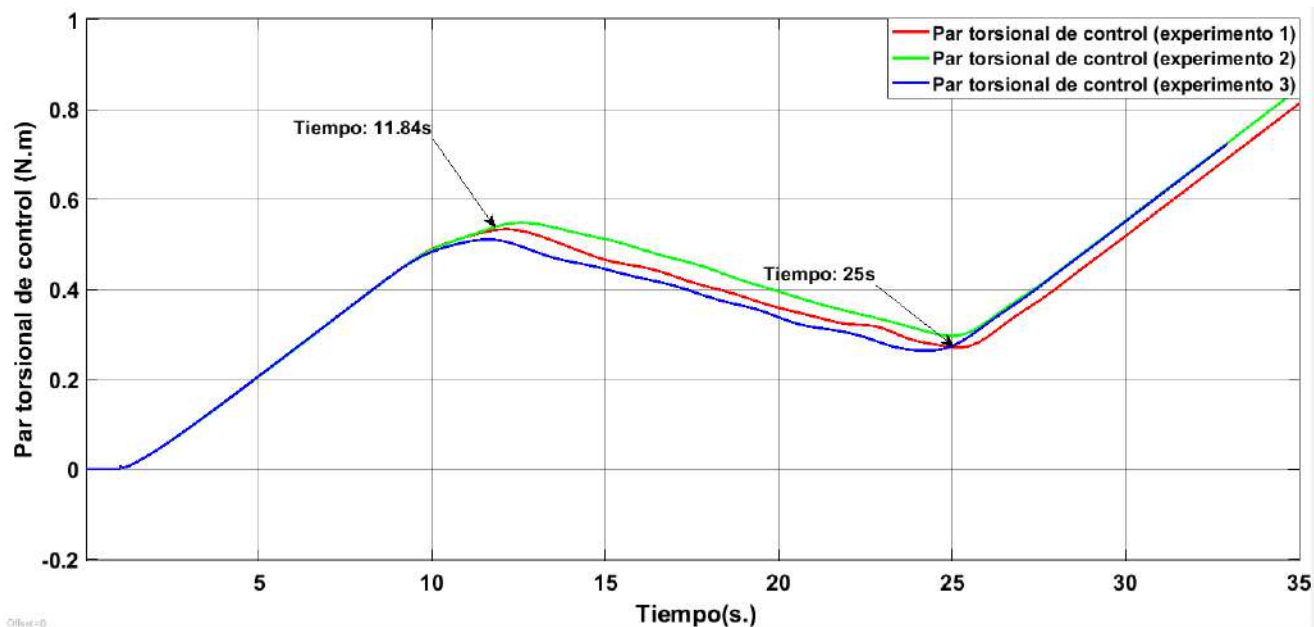


Figura 5.18: Par torsional de control.

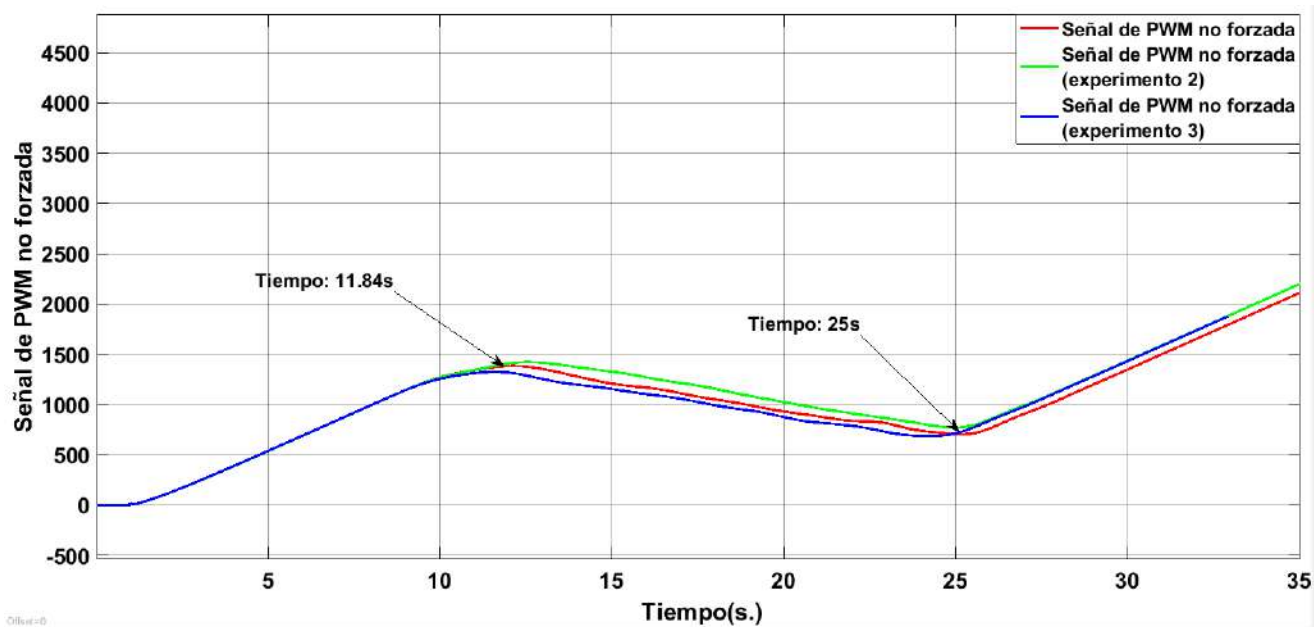


Figura 5.19: PWM de control.

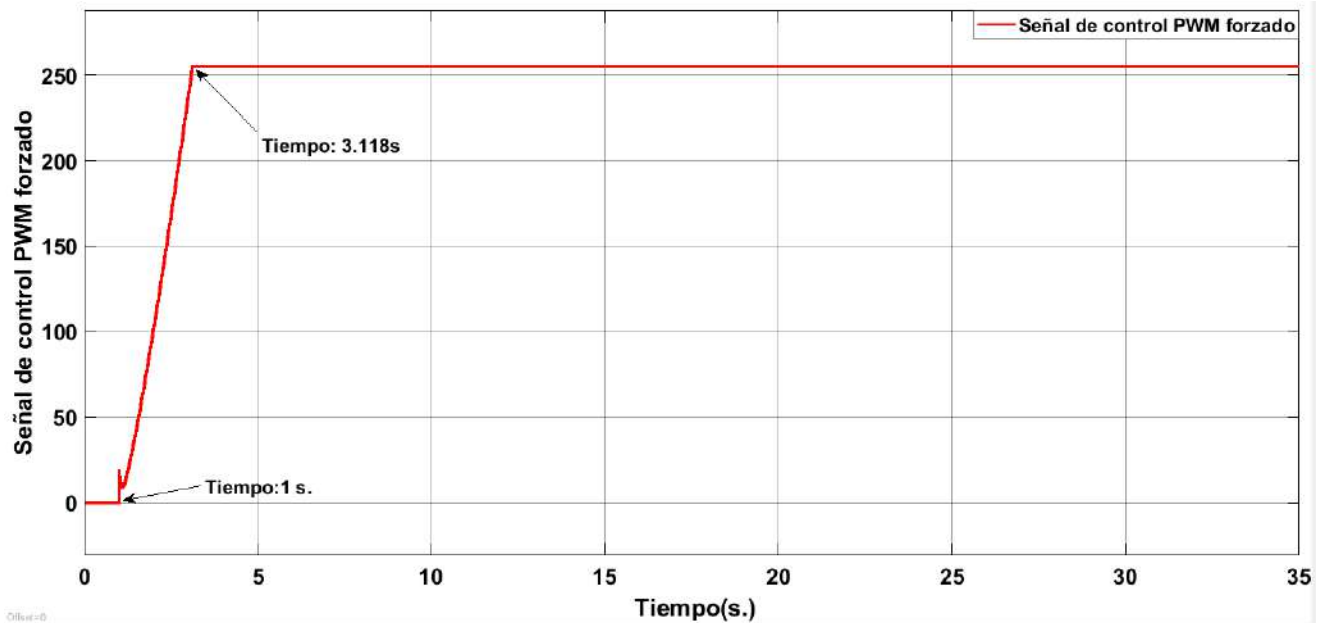


Figura 5.20: PWM segmentado.

Como es posible notar en la Figura 5.16 el control no se desempeñó igual en el sistema físico que en la simulación debido a que hay grandes retrasos en el procesamiento de los datos del algoritmo de control por parte del microcontrolador, esto es fácil de notar puesto que se envía la señal máxima de PWM en el segundo 3,11s (ver Figura 5.20) y al observar la posición y velocidad del robot en ese instante aún son 0 (ver Figuras 5.17 y 5.16), por otra parte, el sistema comienza a responder en el segundo 7,043s, tal y como se puede ver en la Figura de la posición, y en el caso de la velocidad del robot tarda en incrementar debido al filtro Butterborth, el cambio notorio comienza el segundo 7,56s. Por lo tanto la diferencia de tiempo entre el estado actual del sistema y el estado posterior al filtrado es en promedio 0,5s. Sin embargo, este retraso no es el causante del deficiente desempeño del controlador, sino la saturación del buffer en el microcontrolador, esta conjetura se puede sugerir debido a que ya se descartó el retraso causado por filtro. Además, tiene sentido pensar en la saturación del buffer puesto que, se le están mandando un conjunto de números de PWM repetidos por cada periodo de solver mediante el transmisor serial, de modo que, el microcontrolador coloca cada valor de PWM en un determinado tiempo para el accionamiento del motor y después el siguiente valor, pero si el valor de PWM es el mismo, no hay cambio en la señal de PWM que se le manda al motor, por lo tanto, es allí donde ocurre el retraso de accionamiento de la planta.

Ahora bien, cuando el sistema responde y la velocidad sobrepasa la referencia, el PWM debería comenzar a decrementar, pero no es así, debido a que el par torsional demandado fue incrementando al no haber respuesta del sistema, el valor que adquirió es sumamente grande. Siendo esto así, se pensaría que el valor del par torsional que hizo accionar al sistema es muy grande, cuando en realidad se accionó a causa del paso del tiempo, tal y como se muestra en las Figuras 5.18 y 5.19, en donde el máximo local del par torsional de control es a los 11,84s con un valor de $0,5Nm$ que es propiamente el momento en que el sistema llega a la velocidad

del referencia. Sin embargo, el valor de envío de PWM no llega a bajar del valor de 255 (ver Figura 5.20), puesto que el algoritmo de control demanda un valor de PWM de 1400 en el máximo local (ver Figura 5.19) el cual fue truncado a 255. Por lo tanto, para que baje la señal de PWM que se le envía al sistema, el algoritmo de control tendría que proporcionar un valor de PWM inferior a 255, pero para esta práctica, el algoritmo de control no llega a ese valor antes de que se acabe el camino de avance del robot rueda, que es en el segundo 25.

Una vez analizado todo lo anterior se propone como alternativa bajar el valor de la velocidad de referencia, para así llegar antes a la velocidad deseada, y poder analizar el efecto de la reducción de velocidad. Para ello se propone una velocidad de referencia de $50/s = 0,8727rad/s$, dando como respuestas las siguientes gráficas:

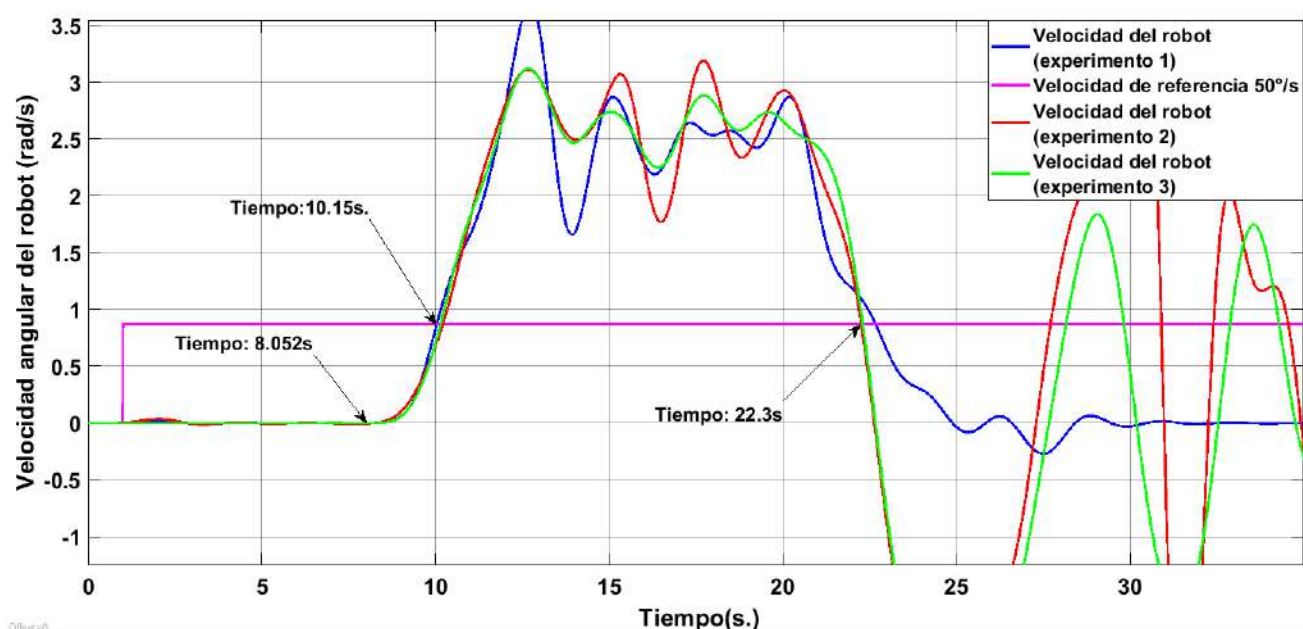


Figura 5.21: Respuesta en velocidad del sistema al algoritmo de control.

Al correr la simulación se obtienen las siguientes respuestas:

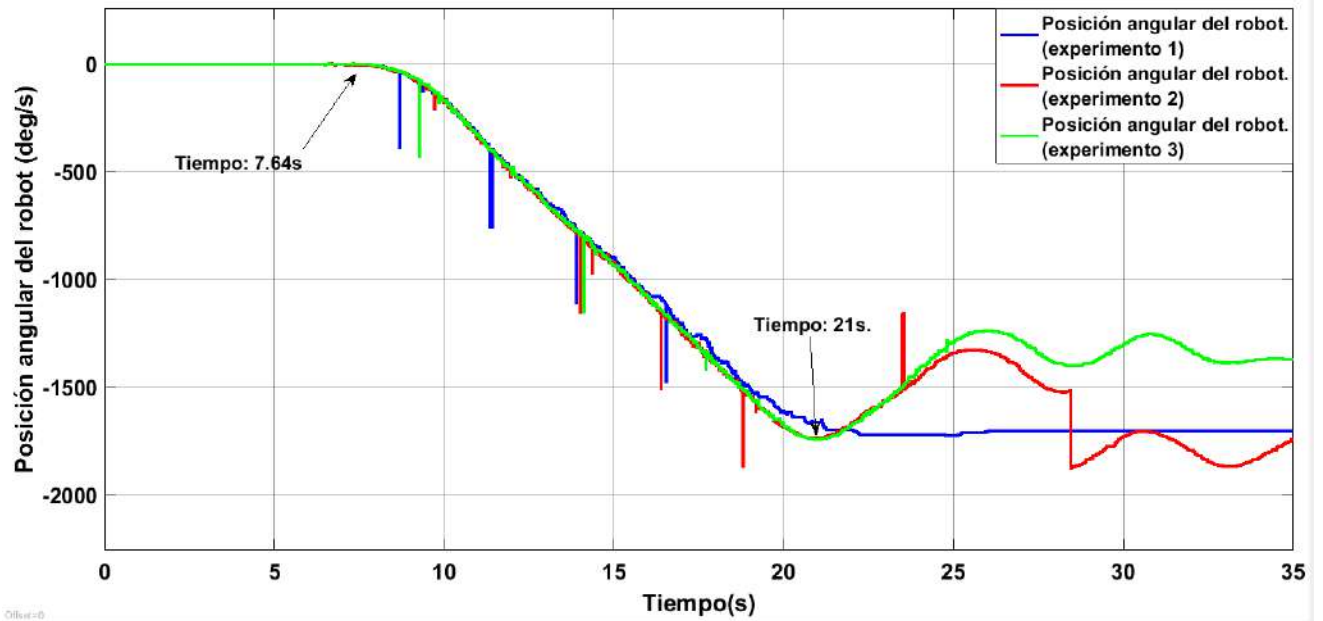


Figura 5.22: Respuesta en posición del sistema al algoritmo de control.

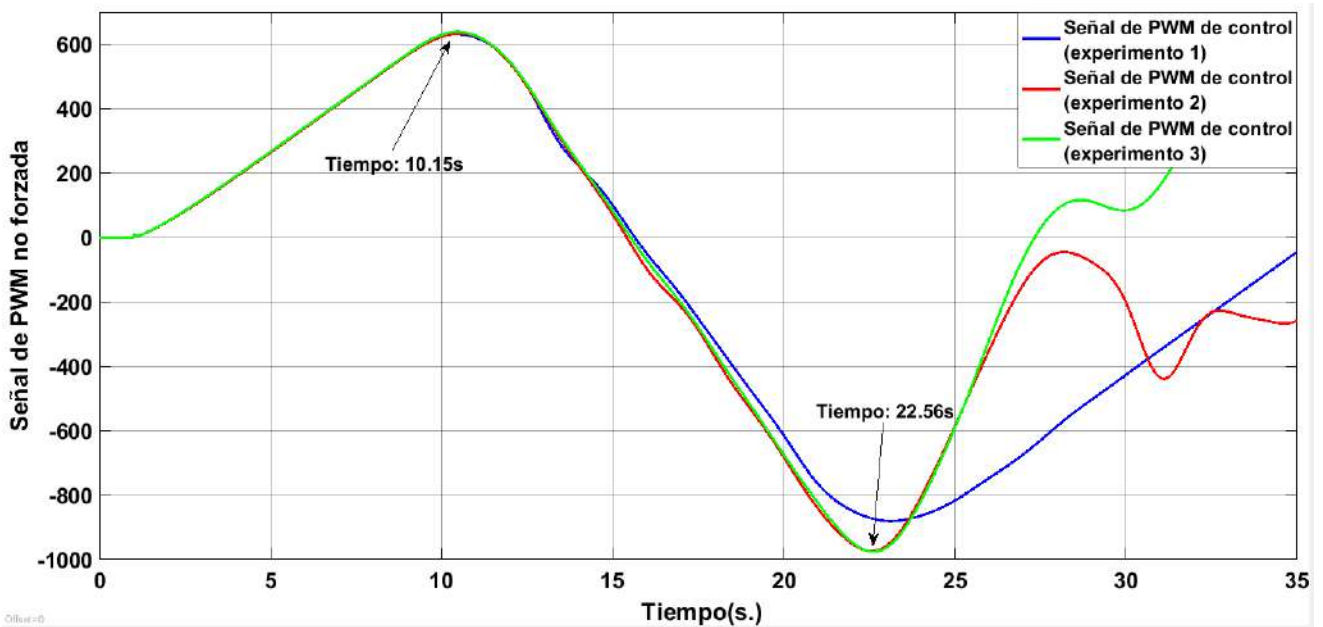


Figura 5.23: PWM de control.

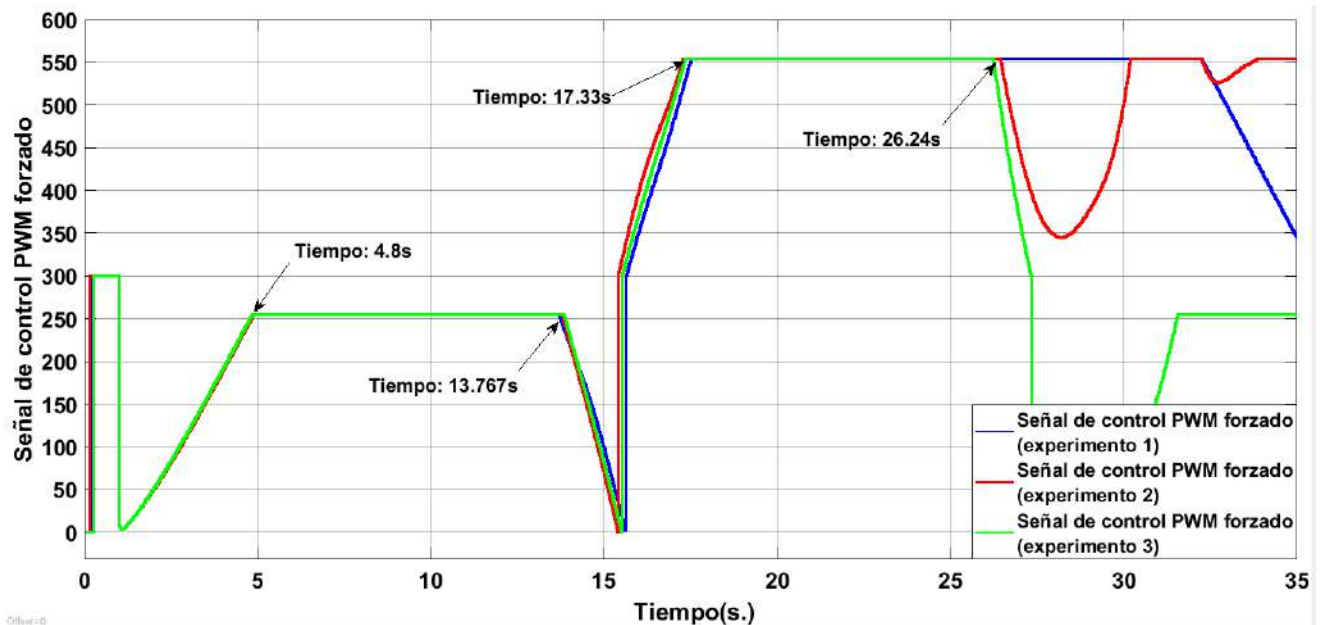


Figura 5.24: PWM segmentado.

Analizando los resultados, se observa lo siguiente:

1. El sistema comienza a moverse a los 7,54s (ver Figura 5.22), para ese entonces el PWM ya está en su valor máximo de 255 (ver Figura 5.24) previamente truncado. por lo que se puede decir que el robot sigue presentando el atraso.
2. Al ir acelerando el robot, pasa la referencia en el segundo 10,15, para ese entonces, el PWM sin truncar tiene un valor superior a 600 (ver Figura 5.23) y empieza a descender hasta el segundo 22,56 con un valor de PWM de -980 . Es de importancia mencionar que el PWM que sale por el transmisor serial tomará valores inferiores a los 255 a partir del segundo 13,767 (ver Figura 5.24) esto es debido que el algoritmo de control debe bajar el valor de PWM de 600 a 255, y para ello le habrá tomado un tiempo de $T = 13,767s - 10,15s = 3,617s$.
3. A pesar de que el PWM comienza a bajar en el segundo 13,76 hasta tomar el valor más bajo de -255 en el segundo 17,33s el sistema no responde en lo absoluto sino hasta el segundo 21s cuando el robot empieza a ir de reversa (ver Figuras 5.22 y 5.21).
4. Puesto que el sistema tardó en responder, para ese entonces el PWM ya es negativo, por lo que este se irá de reversa. Por lo tanto, se es posible aplicar la técnica de control al robot.

Una vez realizado lo anterior se puede decir que el microcontrolador del robot no cumple con los requerimientos necesarios para llevar a cabo el algoritmo de control, puesto que tarda en procesar los datos de llegada en el buffer causando retardos en la respuesta del sistema en general. Dichos retardos provocan ineficiencia al algoritmo de control.

Capítulo 6

Conclusiones y trabajos a futuro

6.1. Conclusiones

Durante el desarrollo del modelo matemático, el análisis del modelo matemático, el diseño del control del robot y el filtrado de la respuesta de velocidad en co-simulación, se observaron resultados satisfactorios debido a que todo concuerda desde el origen, sentando bases desde el diseño mecánico del robot rueda en donde se consideraron los materiales, la simetría de los componentes y los requerimientos de los actuadores, de modo que, al momento de crear la función de transferencia se hicieron tantas consideraciones como fueran posibles para apegarse más al sistema diseñado, y al comparar la respuesta de la función de transferencia con el mecanismo creado, se obtuvieron resultados idénticos, por lo que, se puede garantizar que las consideraciones del mecanismo para la modelación matemática fueron suficientes.

El diseño de control fue enfocado tanto para el sistema de manera virtual como para la función de transferencia, puesto que representan al mismo sistema al comparar las respuestas de ambos. Dicho control se basa en el lugar geométrico de las raíces (LGR) en donde se analizan las propiedades del sistema, las cuales no son óptimas, ya que el sistema es lento, por lo que se propusieron nuevas características como el factor de amortiguamiento y la frecuencia de oscilación, las cuales son interpretadas en un par de polos conjugados dominantes. Para aplicar el lugar geométrico de las raíces es necesario analizar la distribución de polos y ceros del sistema, en el cual se observó que la causa de la inestabilidad al momento de cerrar el lazo. Dicha inestabilidad es causada por el par de polos dominantes del sistema, así como también, por uno de los polos conjugados de la misma. Una solución posible es eliminar tanto los polos dominantes y el cero conjugado negativo de la planta, dando paso a agregar un compensador de eliminación cuya tarea es la eliminación de ellos.

Para agregar un compensador de eliminación se debe tener en cuenta que los polos o ceros a eliminar deben estar en el semiplano negativo, ya que de esa manera no se generará inestabilidad alguna por inexactitud en la eliminación de dichos polos y ceros.

Al comprobar el control con la función de transferencia, se cumplen los requerimientos del sistema, sin embargo, para aproximarse más a la realidad se prueba el algoritmo de control con el sistema virtual. Al hacer la co-simulación se obtienen los mismos resultados que con la

función de transferencia.

Se estimaba que el control funcionaría en el sistema real. Sin embargo, al implementar el robot rueda no se obtuvieron resultados satisfactorios debido a que el microcontrolador no dió la eficiencia suficiente como para procesar la cantidad de datos recibidos por el puerto USART, mientras que, los datos que enviaba eran pocos y no repetitivos, de manera que la Raspberry Pi si es capaz de recibirlos y procesarlos sin ningún problema, todo esto fue notorio por las pruebas realizadas y reportadas en la Sección 5.3. A pesar de que el microcontrolador no dió la eficiencia en las pruebas experimentales, se logró notar el accionamiento del control cada vez que la velocidad del robot cruzaba el valor de referencia, dicho accionamiento incrementaba o decrementaba el valor del PWM para tratar de estabilizar al sistema, así también el accionamiento era más rápido que la repuesta del sistema ante tal cambio, por lo que continuaría el retraso en respuesta del sistema. De modo que, se podría decir que el control funcionaría si se tuviera el microcontrolador que fuera capaz de procesar una trama grande de datos por USART.

Por todo lo anterior se puede decir que se obtuvieron resultados congruentes con base en el diseño, modelación, linealización, control, implementación del sistema y pruebas en lazo abierto, puesto que la función de transferencia estimada del sistema físico da resultados idénticos al sistema mismo, por lo tanto, se estima que el control funcionará al hacer cambios en el hardware del robot rueda.

6.2. Trabajos a futuro

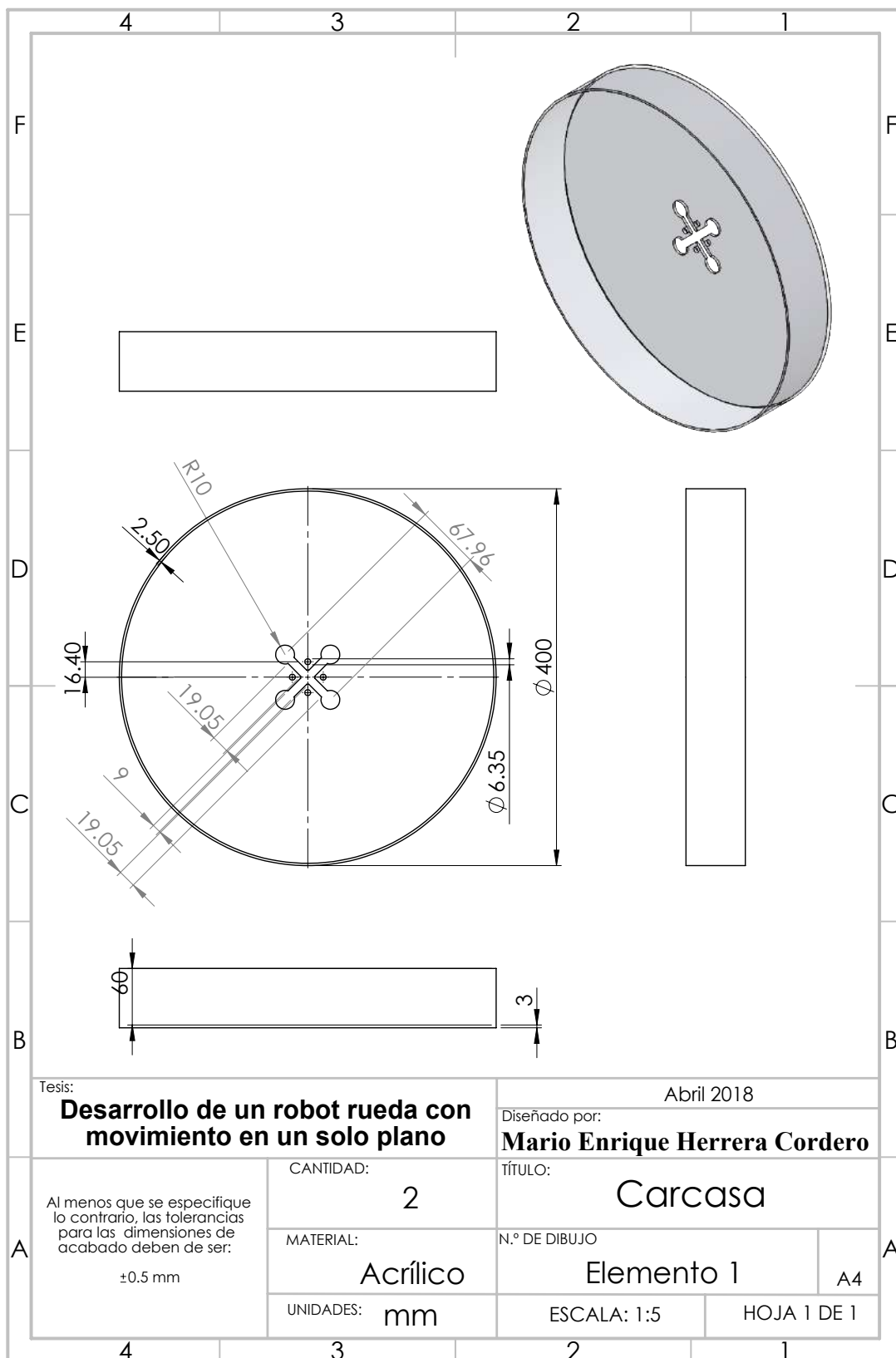
El robot rueda presentado cuenta con aspectos a mejorar y que dan pauta a trabajos a futuro para el mismo, dichos trabajos a futuro son mostrados a continuación:

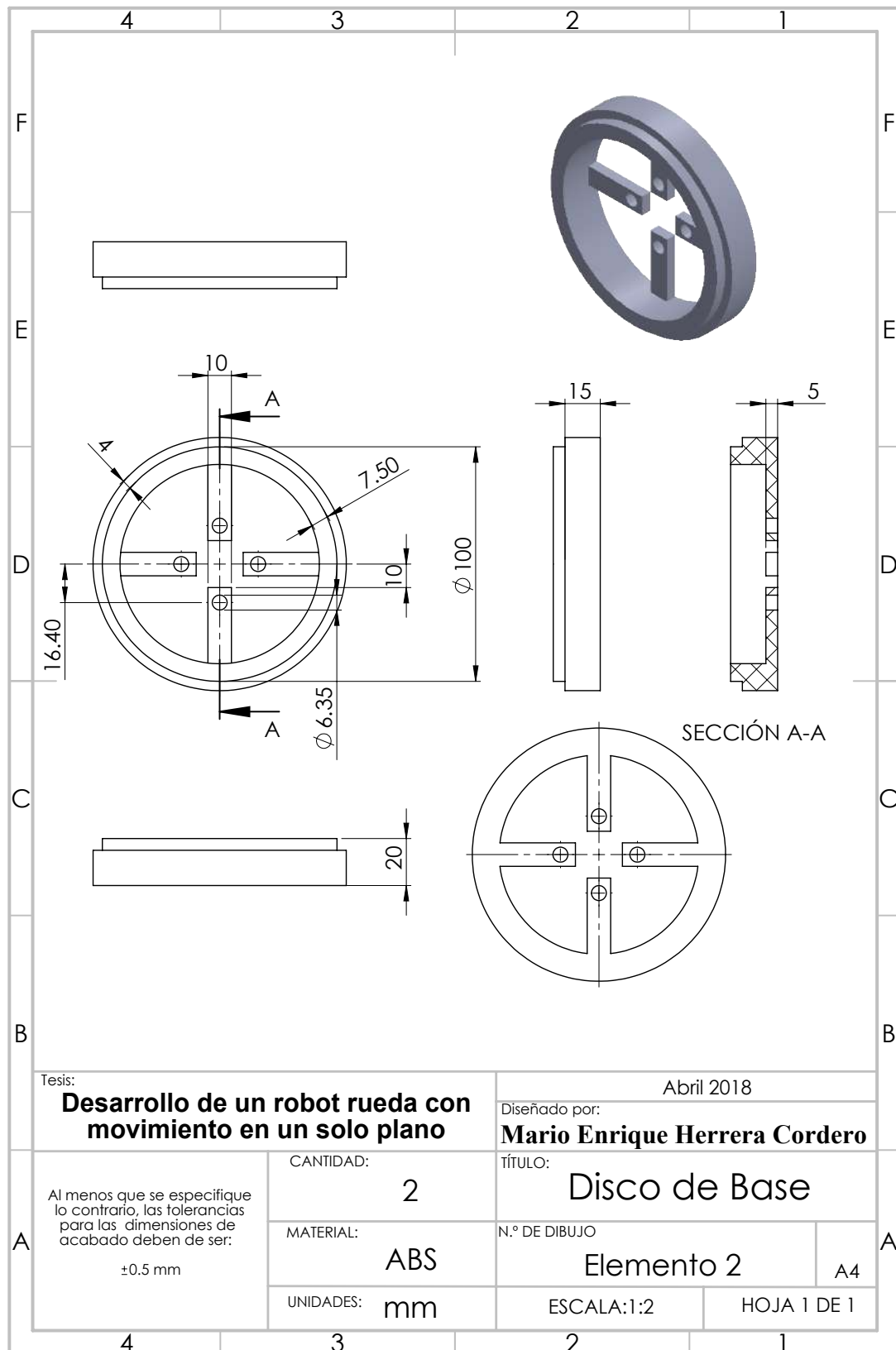
1. Cambiar el microcontrolador interno del robot, por uno de mayor procesamiento.
2. Cambiar el motor pololu por un motor con mayor relación velocidad - par torsional, y que además tenga la flecha extendida.
3. Modificar el modo de alimentación del motor, no usar carbones, sino rodamientos.
4. Colocar un péndulo que sea más pesado que la suma de los pesos de los demás elementos.
5. Cambiar el giroscopio MPU6050 por otro sensor, o aplicar un algoritmo de filtrado eficiente para evitar las perturbaciones ante impulsos del robot rueda.
6. Cambiar la comunicación por Bluetooth por comunicación de radio frecuencia.
7. Implementar algoritmos de control más robustos, para mejorar la respuesta del sistema en L.C.
8. Tratar de implementar el algoritmo de control directamente en el microcontrolador del robot rueda.

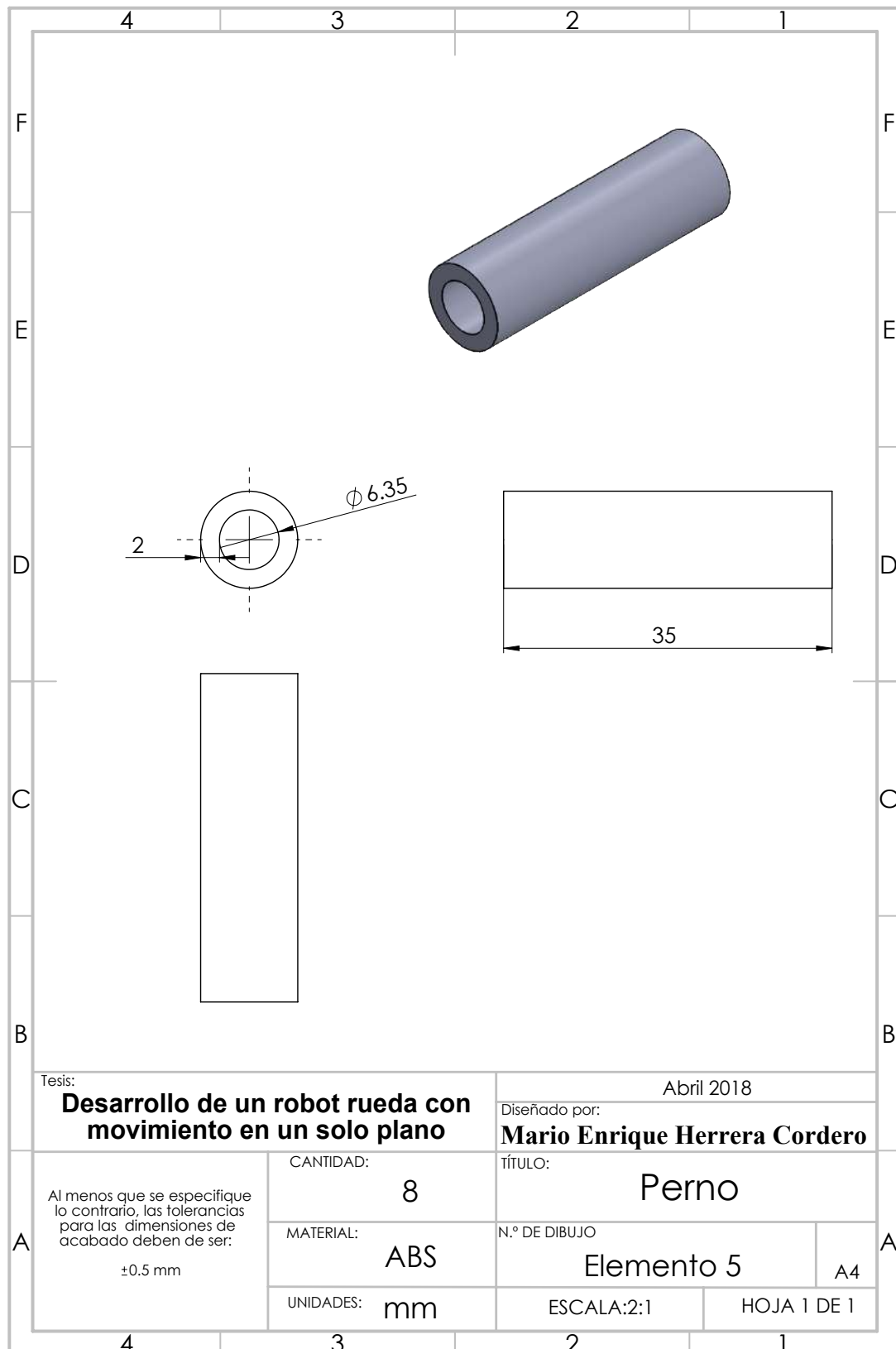
9. Implementar el giro del robot rueda para que sea capaz de desplazarse en un plano.

Apéndice A

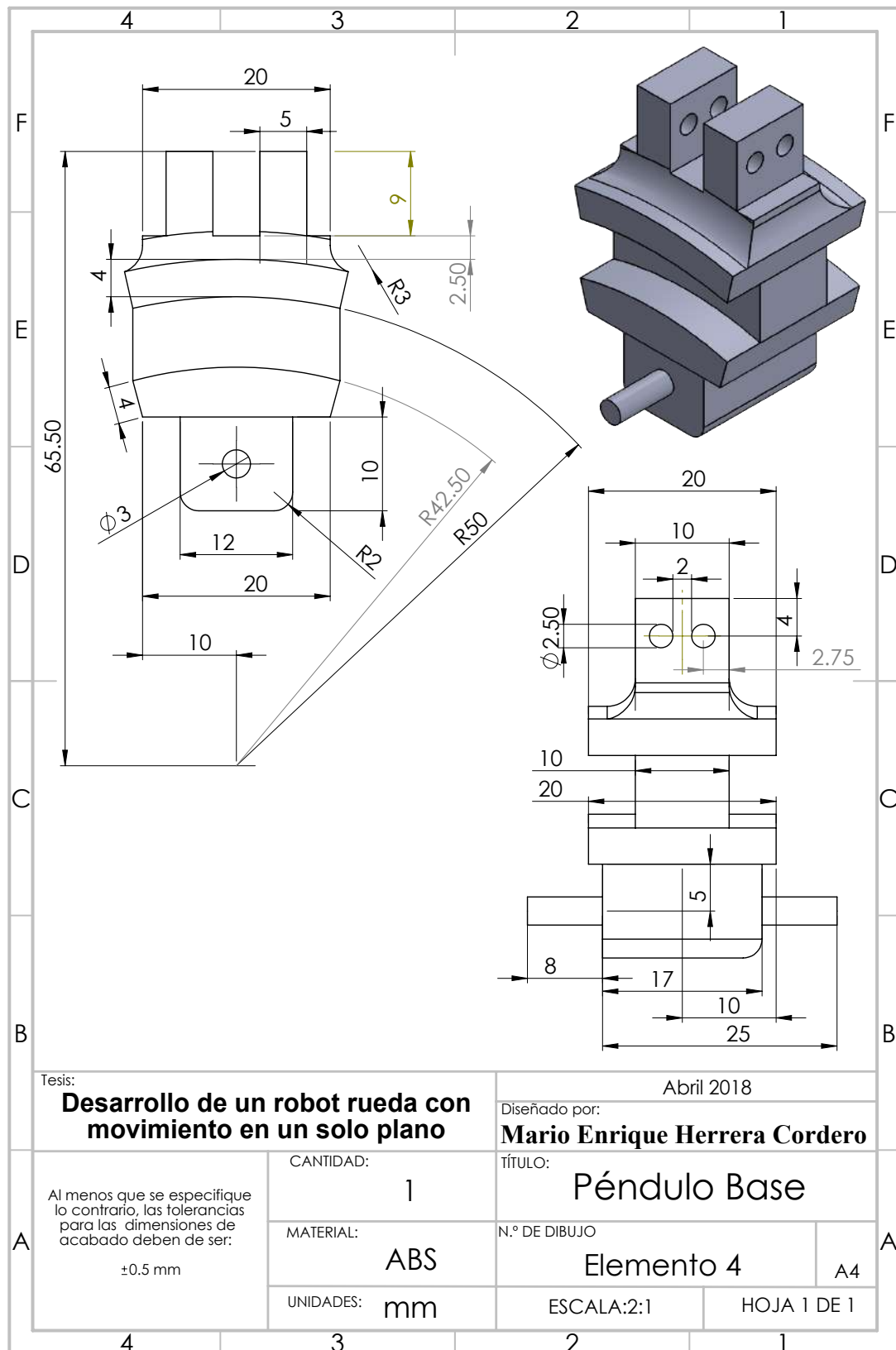
Dibujos técnicos del robot rueda

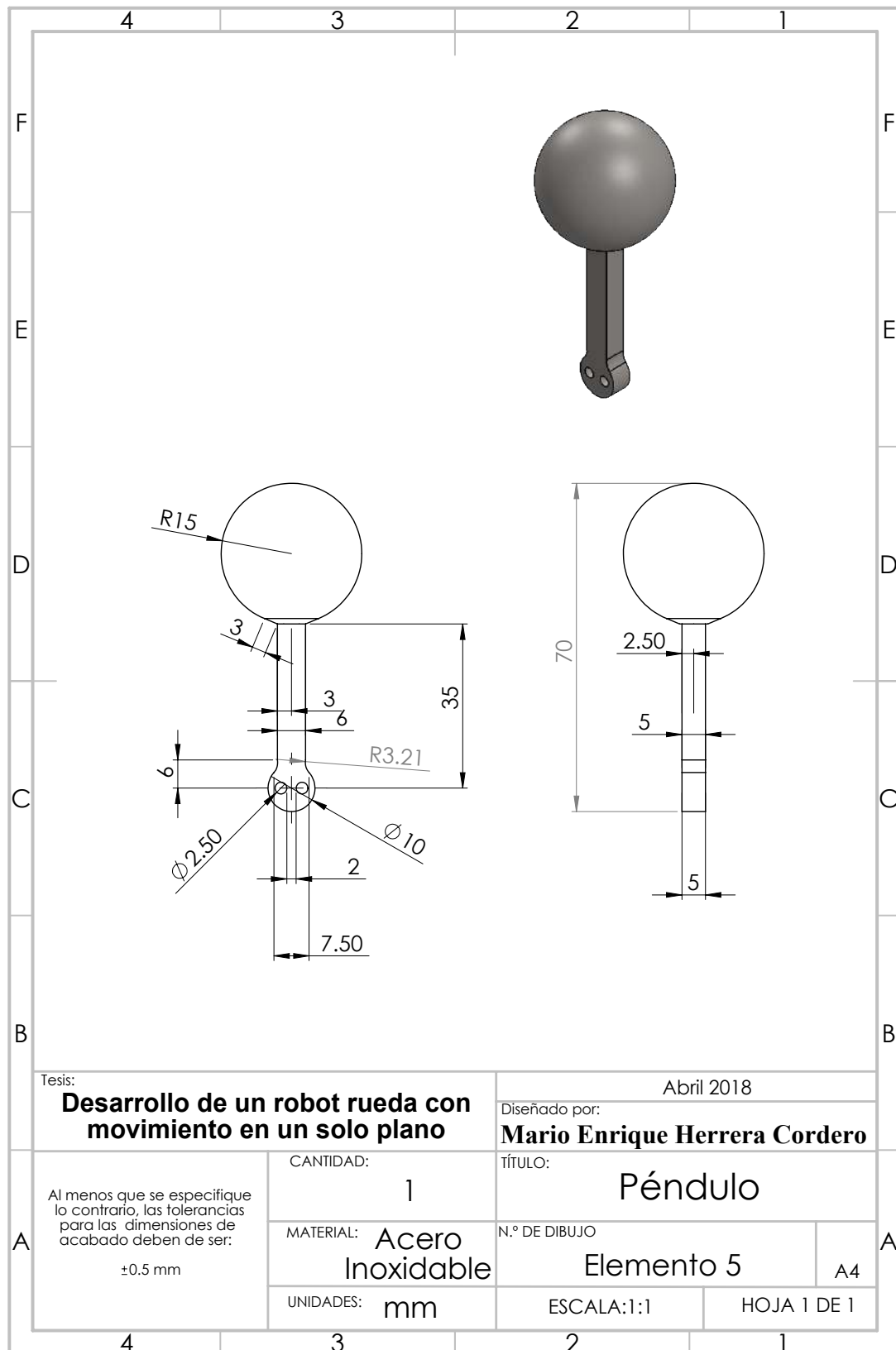


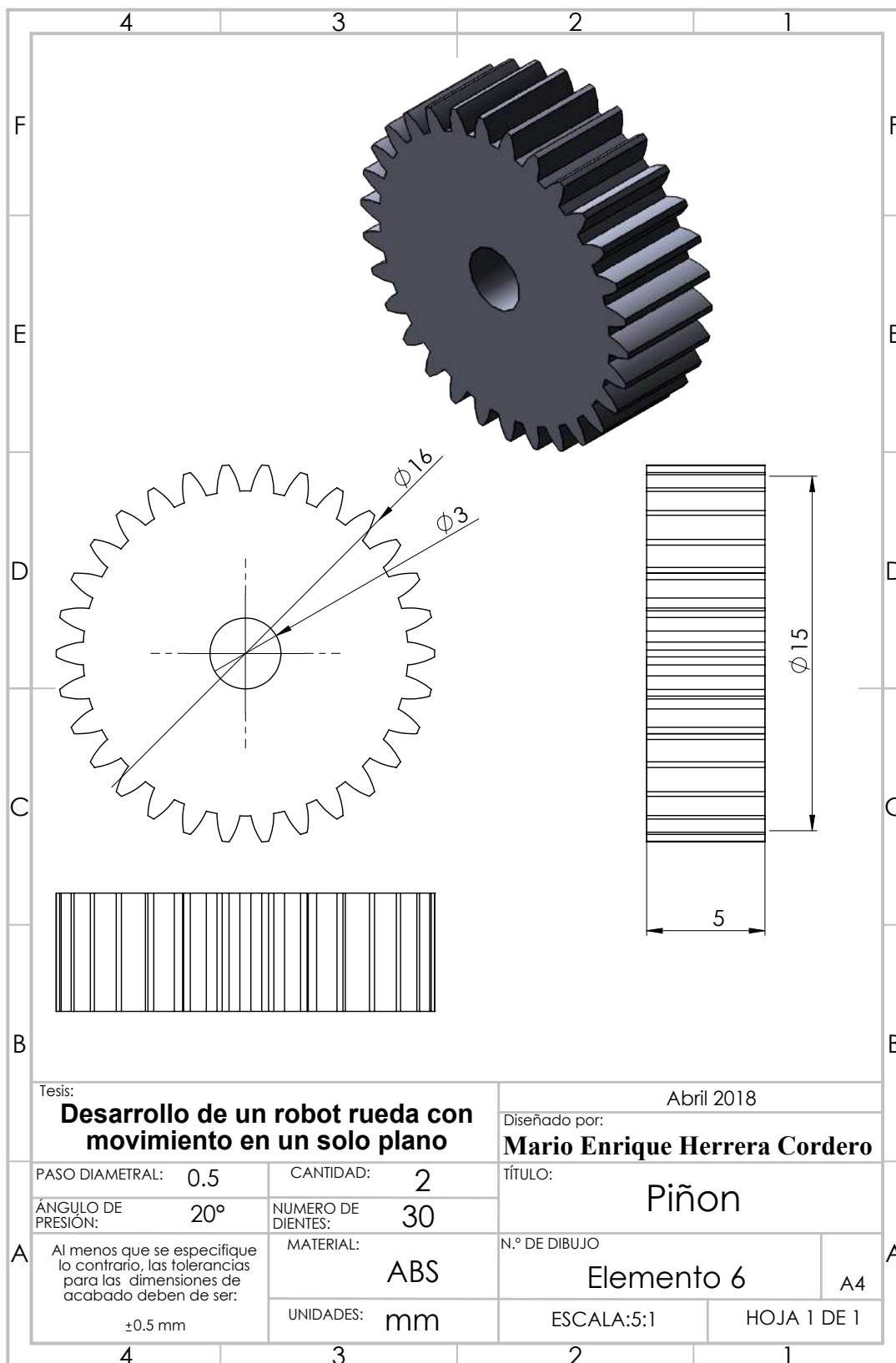


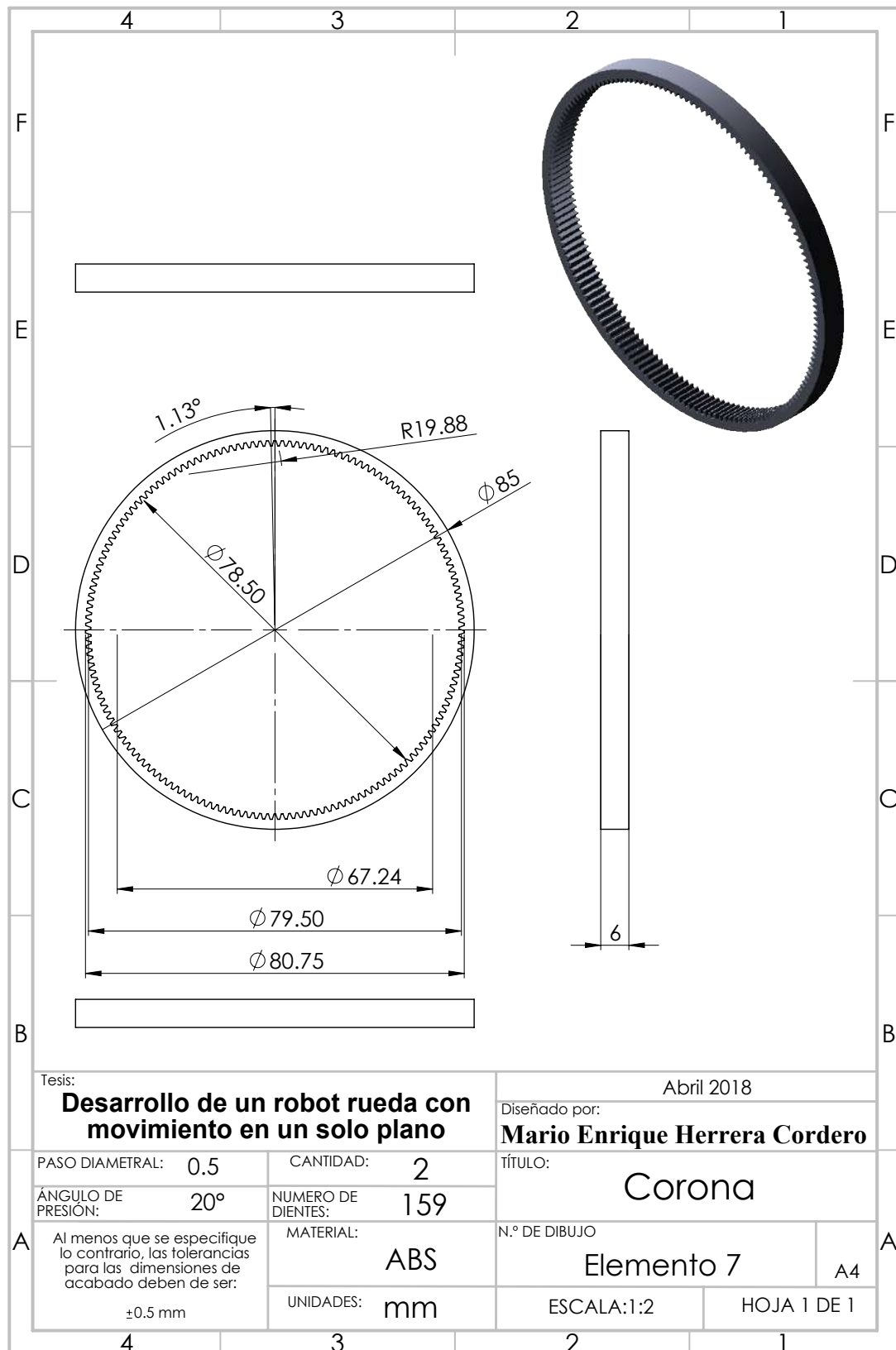


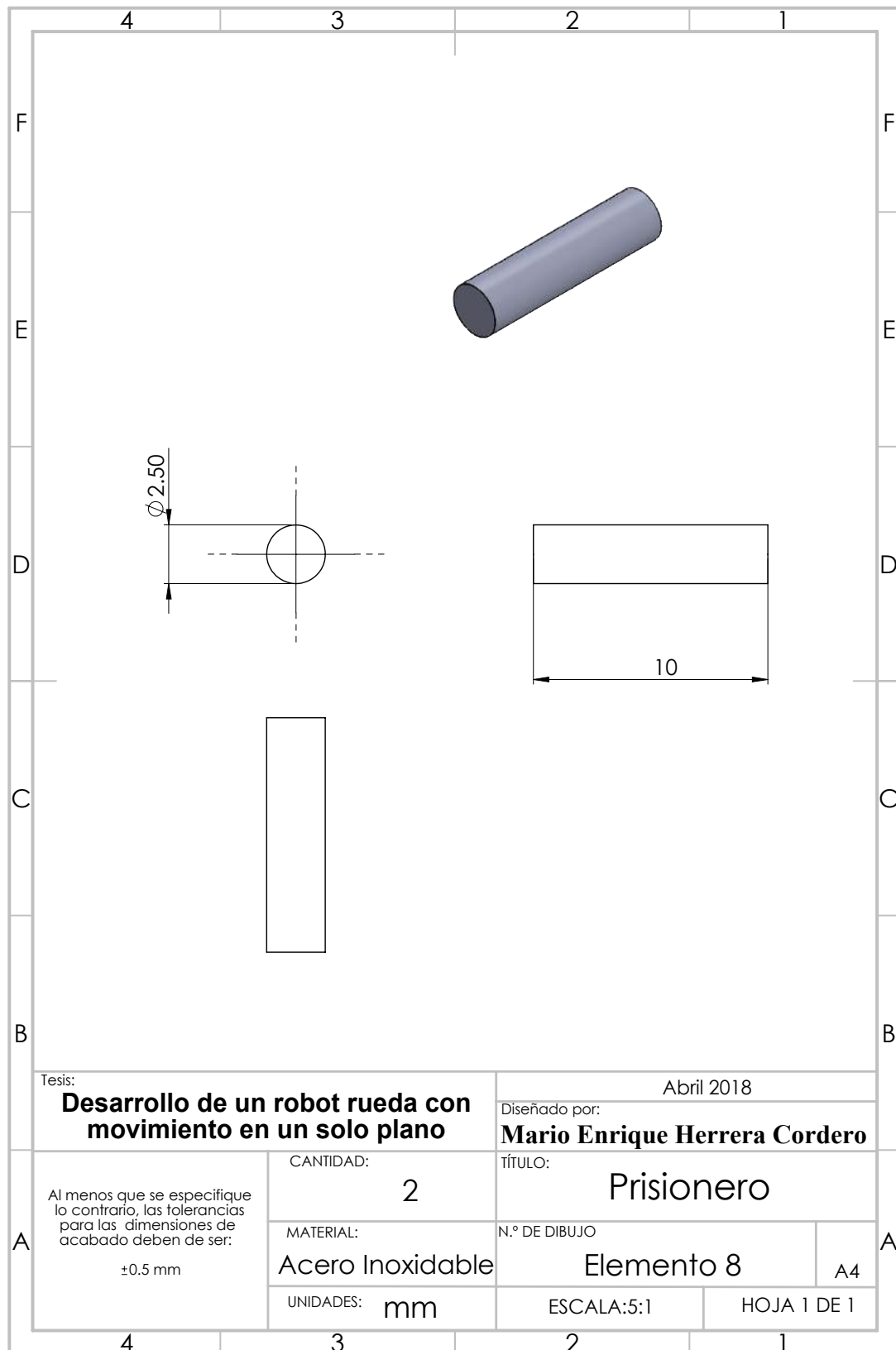
| | | | |
|---|---|---|------------------|
| Tesis: | | Abril 2018 | |
| Desarrollo de un robot rueda con movimiento en un solo plano | | Diseñado por: Mario Enrique Herrera Cordero | |
| A | Al menos que se especifique lo contrario, las tolerancias para las dimensiones de acabado deben de ser: ± 0.5 mm | CANTIDAD: 8 | TÍTULO: Perno |
| | | MATERIAL: ABS | |
| | | UNIDADES: mm | ESCALA:2:1 |











| N.º DE PIEZA | CANTIDAD |
|---------------|----------|
| Carcasa | 2 |
| Disco de Base | 2 |
| Péndulo | 1 |
| Péndulo Base | 1 |
| Perno | 8 |
| prisionero | 2 |
| Corona | 2 |
| Piñon | 2 |

| | | | |
|--|--|---|-------------|
| Tesis: | | Abril 2018 | |
| Desarrollo de un robot rueda con movimiento en un solo plano | | Diseñado por: Mario Enrique Herrera Cordero | |
| A Al menos que se especifique lo contrario, las tolerancias para las dimensiones de acabado deben de ser: $\pm 0.5 \text{ mm}$ | CANTIDAD: 1 | TÍTULO: Explosionado del Robot Rueda | |
| | MATERIAL: ABS, Acrílico y Acero inox | N.º DE DIBUJO Elemento 9 | A4 |
| | UNIDADES: mm | ESCALA:1:5 | HOJA 1 DE 1 |

Apéndice B

Código de programación de microcontrolador de Proteus 8

```
1 #define SCL_CLOCK 1000000L
2 #define F_CPU 1000000UL
3 #include <stdlib.h>
4 #include <string.h>
5 #include <avr/io.h>
6 #include <avr/pgmspace.h>
7 #include <avr/interrupt.h>
8 #include <util/delay.h>
9 #include <inttypes.h>
10 #include <compat/twi.h>
11 #include <math.h>
12 char buf[]="hola_mario_enrique#";
13 char buf2[]="hola_mario_enrique#";
14 volatile uint8_t registroPWM =0;
15 volatile uint16_t SUMAPWM =0;
16 volatile int ContadorF =0;
17 volatile int Corriente=0;
18
19
20 // Funcion de Configuracion de PWM
21
22 void conf_PWM() {
23     TCCR2A=0xA3; //PWM no invertido y PWM rapido, funciona en OCR2A y en OCR2B.
24     TCCR2B=0x01; // sin precalador.
25 }
26
27
28 // Funcion de Configuracion de USART
29
30 void conf_USART() { // Configuramos para una velocidad de 9600.
31     UBRR0H=0x00; // Modo asincrono a doble velocidad.
32     UBRR0L=12; //Habilitamos transmisor y reseptor.
33     UCSR0A=0X02; //los 7-4 bits (interrupcion por dato disponible en
        recepcion) 3-0 bits(doble velocidad)
34     UCSR0B=0X98; //los 7-4 bits (Habilita las interrupciones por recibir
```

```

    dato (RXCIE) y habilita receptor(RXEN),
35      // 3-0 bits Habilita transmisor (TXEN)
36  UCSR0C=0X06; // Operacion asincrona UMSEL[0:1] (0,0), sin paridad, bit de
    paro, tamano de caracter 8 bits.
37      //TXD flanco de subida, RXD flanco de bajada
38  }
39
40  -----
41  //      Funcion de configuracion de lectura analogica
42  -----
43  void confAnalog(void){
44      ADMUX=0x00;
45      ADCSRA=0xCB; // solo se realiza una conversion, por lo que se requiere
46  } //hacer una compuerta or con ADCSRA y 0x40, para mas
    conversiones.
47      // se crea una interrupcion al terminar la conversion.
48      // se sugiere que se permuten las lecturas.
49
50  -----
51  //      Funcion para enviar datos mediante USART
52  -----
53  int enviar(void){ // Enviamos el array a la maquina
54      //UCSR0B=0X08; // se deshabilita el receptor
55      int w=0;
56      int contador=255;
57      while(buf[w]!='#'){ // Enviamos hasta fin de array
58          while(!(UCSR0A & 1 << UDRE0) && contador--); //Esperamos a buffer vacio
59
60          if(contador==0){return 0;}
61          UDR0=buf[w]; // Enviamos el dato.
62          contador=255;
63          w++;
64          _delay_ms(10);
65      } //UCSR0B=0X98; // Se deshabilita el receptor
66      return 1;
67  }
68
69  -----
70  //      Funcion de interrupcion para recepcion de datos
71  -----
72  ISR(USART_RX_vect){
73      int contador=255;
74      int dato=0;
75      dato= UDR0;
76      while(!(UCSR0A & 1 << UDRE0) && contador--); // Esperamos a buffer vacio.
77      if(dato==48){
78          SUMAPWM=0;
79      }
80      else{
81          if(dato==57){
82              SUMAPWM=254;
83          }

```

```

84         else {
85             SUMAPWM=554;
86         }
87     }
88 }
89
90
91 //          Funcion de conversion de datos para enviar
92
93 void conver2(int16_t dato_A){
94     int val=0, val_1=0,aux=0,aux1=0,aux2=0,aux3=0,signo=0;
95     buf[0]=0;
96     buf[1]=0;
97     buf[2]=0;
98     buf[3]=0;
99     buf[4]=0;
100    buf[5]=0;
101    buf[6]=0;
102    if(dato_A<0){
103        signo=1;
104        dato_A=-1*dato_A;
105    }
106    aux=dato_A/1000;           // Numero de millares
107    val=dato_A-aux*1000;     // Residuo de millares
108    aux1=val/100;           // Numero de centenas
109    val_1=val-aux1*100;     // Residuo de centenas
110    aux2=val_1/10;         // Numero de decenas
111    aux3=val_1-aux2*10;     // Unidades
112    if(signo==0){
113        buf[0]='o';
114        buf[1]='+';
115        buf[2]=aux+0x30;
116        buf[3]=aux1+0x30;
117        buf[4]=aux2+0x30;
118        buf[5]=aux3+0x30;
119        buf[6]='p';
120        buf[7]='#';
121    }
122    else {
123        buf[0]='o';
124        buf[1]='-';
125        buf[2]=aux+0x30;
126        buf[3]=aux1+0x30;
127        buf[4]=aux2+0x30;
128        buf[5]=aux3+0x30;
129        buf[6]='p';
130        buf[7]='#';
131    }
132 }
133
134
135
136

```

```

137 // Funcion para la interrupcion por lectura analogica
138
139 ISR(ADC_vect) {
140     ContadorF++; // Contador para el filtrado
141     Corriente=Corriente + ADCW; // Suma de las senales adquiridas
142     if(ContadorF==50){
143         Corriente=Corriente/ContadorF;
144         Corriente=(0.000048267*pow((Corriente),2) + (-0.0827964)*Corriente
145             + 32.42535)*100;
146         conver2(Corriente);
147         if(enviar()==0){
148             conf_USART();
149         }
150         ContadorF=0;
151         Corriente=0;
152     }
153     _delay_ms(5);
154     ADCSRA=ADCSRA | 0x40;
155 }
156
157 // Funcion principal
158
159 int main(void) {
160     DDRB=0x0F;
161     DDRD=0xFA; // Habilitamos el pin 2 como salida, ya que en ese se enviaran
162                // los datos atravez de la USART
163     PORTD=0x05;
164     conf_USART();
165     conf_PWM();
166     confAnalog();
167     sei();
168     for(;;) {
169         if(SUMAPWM<256){
170             OCR2B=0;
171             OCR2A=SUMAPWM;
172         }
173         else{
174             if(SUMAPWM>256){
175                 OCR2A=0;
176                 OCR2B=SUMAPWM-300;
177             }
178         }
179     }
180 }
181 }

```


Apéndice C

Código programación de microcontrolador con giroscopio

```
182 #define SCL_CLOCK 1000000L
183 #define F_CPU 1000000UL
184 #include <stdlib.h>
185 #include <string.h>
186 #include <avr/io.h>
187 #include <avr/pgmspace.h>
188 #include <avr/interrupt.h>
189 #include <util/delay.h>
190 #include <inttypes.h>
191 #include <compat/twi.h>
192 #include <math.h> //include libm
193
194
195 // Declaracion de Registros para Giroscopio
196
197 #define MPU6050_ADDR (0x68 <<1)
198 #define MPU6050_RA_PWR_MGMT_1 0x6B
199 #define MPU6050_PWR1_SLEEP_BIT 6
200 #define MPU6050_RA_ACCEL_XOUT_H 0x3B
201
202 // Declaracion de Codigos de Respuestas de I2C
203
204 #define SCL_CLOCK 1000000L
205 /* I2C timer max delay */
206 #define I2C_TIMER_DELAY 0xFF
207 #define TW_START 0x08
208 #define TW_REP_START 0x10
209 #define TW_MT_SLA_ACK 0x18
210 #define TW_MR_SLA_ACK 0x40
211 #define TW_MT_SLA_NACK 0x20
212 #define TW_MR_DATA_NACK 0x58
213 #define TW_MT_DATA_ACK 0x28
214
215
216 // Constante para leer del dispositivo I2C
```

```

217
218 #define I2C_READ    1
219
220
221 //                Constante para escribir del dispositivo I2C
222
223 #define I2C_WRITE   0
224
225
226 //                Declaracion de constantes y variables
227
228 #define A_R 16384.0
229 #define G_R 131.0
230 #define RAD_A_DEG 57.295779
231 volatile uint8_t buffer [14];
232 char buf[]="hola_mario_enrique#";
233 char buf2[]="hola_mario_enrique#";
234 volatile uint8_t registroPWM =0;
235 volatile uint16_t SUMAPWM =0;
236 volatile uint8_t comienzo =0;
237
238 //                Inicializar I2C
239
240 void i2c_init(void)
241 {
242
243     TWSR = 0x00;        // Sin prescalador
244     TWBR = 0x00;
245
246 }
247
248
249 //                Iniciar I2C
250 //                recibe la direccion de inicio
251 //                retorna 1 para acceso fallido y 0 para acceso autorizado
252
253 int8_t i2c_start(unsigned char address)
254 {
255     uint32_t i2c_timer = 0;
256     uint8_t twst;
257     TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN); // Condicion de inicio.
258     i2c_timer = I2C_TIMER_DELAY; // Tiempo de espera para la transmision.
259     while (!(TWCR & (1<<TWINT)) && i2c_timer--);
260     if(i2c_timer == 0) // Si se agoto el tiempo retornar 1.
261         return 1;
262
263     twst = TW_STATUS & 0xF8; // Visualizar si hay algun error de envio.
264     if ( (twst != TW_START) && (twst != TW_REP_START)) return 1;
265
266     TWDR = address; // Enviar la direccion de direccion a acceder.
267     TWCR = (1<<TWINT) | (1<<TWEN);
268
269

```

```

270     i2c_timer = I2C_TIMER_DELAY; // Tiempo de espera para la transmision.
271     while (!(TWCR & (1<<TWINT)) && i2c_timer--);
272     if(i2c_timer == 0) // Si se agoto el tiempo retornar 1.
273         return 1;
274
275
276     twst = TW_STATUS & 0xF8; // Visualizar si hay algun error de envio.
277     if ( (twst != TW_MT_SLA_ACK) && (twst != TW_MR_SLA_ACK) ) return 1;
278
279     return 0; // Se accedio exitosamente a la direccion.
280
281 }
282
283


---


284 // Funcion para terminar la transferencia de datos
285
286 int8_t i2c_stop(void)
287 {
288     uint32_t i2c_timer = 0;
289
290     TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO); // Condicion de paro.
291     i2c_timer = I2C_TIMER_DELAY; // Tiempo de espera para la transmision.
292     while((TWCR & (1<<TWSTO)) && i2c_timer--);
293     if(i2c_timer==0){return 1;} // Si se agoto el tiempo retornar 1.
294     return 0; // Se accedio exitosamente a la direccion.
295 }
296
297


---


298 Enviar un dato al dispositivo
299
300 Entrada: Byte de informacion
301 Retorno: 0 de dato enviado.
302          1 de dato no enviado.
303


---


304 int8_t i2c_write( unsigned char data )
305 {
306     uint32_t i2c_timer = 0;
307     uint8_t twst;
308
309     TWDR = data; // Enviar el dato.
310     TWCR = (1<<TWINT) | (1<<TWEN);
311
312     // wait until transmission completed
313     i2c_timer = I2C_TIMER_DELAY; //Tiempo de espera para la transmision.
314     while (!(TWCR & (1<<TWINT)) && i2c_timer--);
315     if(i2c_timer == 0) // Se agoto el tiempo
316         return 1;
317
318
319     twst = TW_STATUS & 0xF8; // Visualizar si hay algun error de envio.
320     if( twst != TW_MT_DATA_ACK) return 1;
321     return 0; // Se envio exitosamente el dato.
322

```

```

323 }
324
325 

---


326 // Funcion de solicitar un Byte.
327
328 // Retornar: Retornar el Byte leído
329 

---


330 unsigned char i2c_readAck(void)
331 {
332     uint32_t i2c_timer = 0;
333
334     TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
335     i2c_timer = I2C_TIMER_DELAY; // Tiempo de espera.
336     while (!(TWCR & (1<<TWINT)) && i2c_timer--);
337     if(i2c_timer == 0) // Error al recibir al dato.
338         return 0;
339
340     return TWDR; // Enviar el dato.
341 }
342
343 

---


344 // Leer un Byte del dispositivo continuado por una condicion de paro
345
346 // Retorno: Byte leído del dispositivo
347 

---


348
349 unsigned char i2c_readNak(void)
350 {
351     uint32_t i2c_timer = 0;
352
353     TWCR = (1<<TWINT) | (1<<TWEN); // Solicitar dato y condicion de paro.
354     i2c_timer = I2C_TIMER_DELAY; // Tiempo de espero.
355     while (!(TWCR & (1<<TWINT)) && i2c_timer--);
356     if(i2c_timer == 0) // Error al leer el dato.
357         return 0;
358
359     return TWDR; // Enviar dato extraido.
360 }
361 }
362
363 

---


364 // Leer datos en los registros del Giroscopio
365 

---


366 int8_t mpu6050_readBytes(uint8_t regAddr, uint8_t length, uint8_t *data) {
367     uint8_t i = 0;
368     int8_t count = 0;
369     if(length > 0) {
370         if(i2c_start(MPU6050_ADDR | I2C_WRITE)==1){return 0;} // No hubo exito.
371         if(i2c_write(regAddr)==1){return 0;} // Error accediendo al registro.
372         _delay_us(10);
373         if(i2c_start(MPU6050_ADDR | I2C_READ)==1){return 0;} // No hubo exito.
374         for(i=0; i<length; i++) { // Extraccion de datos.
375             count++;

```

```

376         if(i==length-1)
377             data[i] = i2c_readNak(); // Extraer el ultimo dato y parar.
378         else
379             data[i] = i2c_readAck(); // Extraer datos.
380     }
381     if(i2c_stop()==1){return 0;} // Detener la comunicacion I2C.
382 }
383 return count; // Mandamos el contador.
384 }
385
386
387 // Escribir un dato en el dispositivo
388
389 int8_t mpu6050_writeByte(uint8_t regAddr, uint8_t data) {
390     return mpu6050_writeBytes(regAddr, 1, &data);
391 }
392
393
394 // Leer un dato en el dispositivo
395
396 int8_t mpu6050_readByte(uint8_t regAddr, uint8_t *data) {
397     return mpu6050_readBytes(regAddr, 1, data);
398 }
399
400
401 // Leer un dato en el dispositivo
402
403 int8_t mpu6050_writeBit(uint8_t regAddr, uint8_t bitNum, uint8_t data) {
404     uint8_t b;
405     if(mpu6050_readByte(regAddr, &b)==0){return 0;}
406     b = (data != 0) ? (b | (1 << bitNum)) : (b & ~(1 << bitNum));
407     if(mpu6050_writeByte(regAddr, b)==0){return 0;}
408     return 1;
409 }
410
411
412 // Funcion para despertar al giroscopio
413
414 int8_t mpu6050_setSleepDisabled() {
415     return mpu6050_writeBit(MPU6050_RA_PWR_MGMT_1, MPU6050_PWR1_SLEEP_BIT, 0);
416 }
417
418
419 // Funcion para inicializar al giroscopio
420
421 void mpu6050_init(void) {
422     do{
423         i2c_init();
424         _delay_us(10);
425     }while(mpu6050_setSleepDisabled()==0);
426     _delay_ms(10);
427 }
428

```

```

429
430 //          Funcion para extraer datos de registro de acelerometro
431
432 int8_t mpu6050_getRawData(int16_t* ax, int16_t* ay, int16_t* az, int16_t* gx,
    int16_t* gy, int16_t* gz, int8_t* ktr, int8_t* ktr1) {
433     int retorno=0;
434     retorno=mpu6050_readBytes(MPU6050_RA_ACCEL_XOUT_H, 14, (uint8_t *)buffer);
435     if(retorno<14){return 0;}
436     *ktr=buffer[0];
437     *ktr1=buffer[1];
438     *ax = (((int16_t)buffer[0]) << 8) | buffer[1];
439     *ay = (((int16_t)buffer[2]) << 8) | buffer[3];
440     *az = (((int16_t)buffer[4]) << 8) | buffer[5];
441     *gx = (((int16_t)buffer[8]) << 8) | buffer[9];
442     *gy = (((int16_t)buffer[10]) << 8) | buffer[11];
443     *gz = (((int16_t)buffer[12]) << 8) | buffer[13];
444     return 1;
445 }
446
447
448
449 //          Funcion de Configuracion de PWM
450
451 void conf_PWM(){
452     TCCR2A=0xA3; // PWM no invertido y PWM rapido, funciona en OCR2A y en OCR2B
453
454     TCCR2B=0x01; // sin precalador.
455 }
456
457 //          Funcion de Configuracion de USART
458
459 void conf_USART(){ // Configuramos para una velocidad de 9600.
460     UBRR0H=0x00; // Modo asincrono a doble velocidad.
461     UBRR0L=12; // Habilitamos transmisor y reseptor.
462     UCSR0A=0X02; // Los 7-4 bits (interrupcion por dato disponible en
        recepcion) 3-0 bits(doble velocidad)
463     UCSR0B=0X98; // Los 7-4 bits (Habilita las interrupciones por recibir
        dato (RXCIE) y habilita receptor(RXEN),
464 // 3-0 bits Habilita transimisor (TXEN)
465     UCSR0C=0X06; // Operacion asincrona UMSEL[0:1] (0,0), sin paridad, bit
        de paro, tamano de caracter 8 bits.
466 // TXD flanco de subida, RXD flanco de bajada
467 }
468
469
470 //          Funcion para enviar datos mediante USART
471
472 int enviar(void){ // Enviamos el array a la maquina
473     int w=0;
474     int contador=255;
475     while(buf[w]!='#'){ // Enviamos hasta fin de array
476         while(!(UCSR0A & 1 << UDRE0) && contador--); // Esperamos a buffer vacio

```

```

477         if(contador==0){return 0;}
478         UDR0=buf[w];           // Enviamos el dato.
479         contador=255;
480         w++;
481         _delay_ms(10);
482     }
483     return 1;           // Se envia el dato con exito
484 }
485
486
487 //           Funcion de Interrupcion para la recepcion de datos.
488
489 //Para la interpretacion de datos recibidos se ordenan de mayor a menor digito
490 //primero las centenas, despues las decenas, y por ultimo las unidades con
491 //caracter de inicio y paro, LF y . correspondientemente.
492
493 ISR(USART_RX_vect){
494     int dato=0;
495     dato= UDR0;
496     while (!(UCSR0A & 1 << UDRE0)); //Esperamos a buffer vacio.
497     if(dato==10){           // Se recibe LF (nueva linea), inicio de numero.
498         buf2[0]=dato;
499         registroPWM=1; // Conteo de digitos
500     }
501     else{
502         if(47<dato && dato<58 && dato!=46 && registroPWM < 4){
503             buf2[registroPWM]=dato;
504             registroPWM=registroPWM+1;
505         }
506         else{
507             if(dato==46){ // Se recibe . (punto), comienza la decodificacion.
508                 if(registroPWM==2){
509                     SUMAPWM=buf2[1]- 48;
510                 }
511                 if(registroPWM==3){
512                     SUMAPWM=(buf2[1]-48)*10 + buf2[2]- 48;
513                 }
514                 if(registroPWM==4){
515                     SUMAPWM=(buf2[1]-48)*100 + (buf2[2]- 48)*10 + buf2[3]-
516                         48;
517                 }
518                 registroPWM=10;
519             }
520             else{
521                 if(dato==42){
522                     comienzo=1;
523                 }
524             }
525         }
526     }
527 }

```

```

528
529
530 //          Conversion de numero en codigo ascci
531 // Se convierte el numero a numeros ascci con caracteres de inicio y paro
532 //          Se proponen la 'o' y la 'p' como caracteres de inicio y paro.
533
534 void conver2(int16_t dato_A){
535     int val=0, val_1=0,aux=0,aux1=0,aux2=0,aux3=0,signo=0;
536     buf[0]=0;
537     buf[1]=0;
538     buf[2]=0;
539     buf[3]=0;
540     buf[4]=0;
541     buf[5]=0;
542     buf[6]=0;
543     if(dato_A<0){
544         signo=1;
545         dato_A=-1*dato_A;
546     }
547     aux=dato_A/1000;           // numero de millares.
548     val=dato_A-aux*1000;     // residuo de millares.
549     aux1=val/100;           // numero de centenas.
550     val_1=val-aux1*100;     // residuo de centenas.
551     aux2=val_1/10;         // numero de decenas.
552     aux3=val_1-aux2*10;     //unidades.
553     if(signo==0){
554         buf[0]='o';
555         buf[1]='+';
556         buf[2]=aux+0x30;
557         buf[3]=aux1+0x30;
558         buf[4]=aux2+0x30;
559         buf[5]=aux3+0x30;
560         buf[6]='p';
561         buf[7]='#';
562     }
563     else{
564         buf[0]='o';
565         buf[1]='-';
566         buf[2]=aux+0x30;
567         buf[3]=aux1+0x30;
568         buf[4]=aux2+0x30;
569         buf[5]=aux3+0x30;
570         buf[6]='p';
571         buf[7]='#';           // caracter de paro para envio.
572     }
573 }
574
575
576 //          Funcion para extraer los datos del acelerometro del giroscopio
577
578 int data_giroscopio(void){
579     int8_t kt=0;
580     int8_t kt1=0;

```

```

581     int16_t ax = 0;
582     int16_t ay = 0;
583     int16_t az = 0;
584     int16_t gx = 0;
585     int16_t gy = 0;
586     int16_t gz = 0;
587     int Acc[2];
588
589 // Agregar los valores de las aceleraciones en cada una de las coordenadas
590     while(mpu6050_getRawData(&ax, &ay, &az, &gx, &gy, &gz,&kt,&kt1)==0);
591
592     // Aplicacion de expresion matematica para angulo de inclinacion.
593     if(ay>0) // Evaluacion de giro para cambio de signo.
594     {Acc[1]=90 + atan((ax/A_R)/sqrt(pow((ay/A_R),2)))*(RAD_A_DEG);}
595     else{
596         Acc[1]=270 - atan((ax/A_R)/sqrt(pow((ay/A_R),2)))*(RAD_A_DEG);
597     }
598     return Acc[1]; // retornar el angulo estimado
599 }
600 }
601
602
603 // Funcion principal
604
605 int main(void) {
606     DDRB=0x0F; // Primeros 4 pines del puerto B como salida.
607     DDRD=0xFA; // Pin 2 y 4 del puerto D como salida de la comunicacion USART.
608     PORTD=0x01; // Pin 1 del puerto D como entrada de la comunicacion USART.
609     int counter=0,theta_actual=0,theta_anterior=0,bandera=0, posicion=0,
        PosInicial=0;
610     conf_USART(); // Configuracion de la comunicacion serial.
611     conf_PWM(); // Configuracion del PWM.
612     sei();
613     mpu6050_init(); // Despertar al Giroscopio.
614     _delay_ms(50);
615     while(counter<5){ // Filtrado de la posicion inicial.
616         PosInicial=PosInicial + data_giroscopio();
617         counter++;
618     }
619     PosInicial=PosInicial/5; //Estimacion del promedio.
620     counter=0;
621     theta_actual=theta_anterior=data_giroscopio();
622     for(;;) {
623         if(comienzo==1){
624             if(theta_actual>100 && theta_actual<200 && bandera==1 && 100<
                theta_anterior && theta_anterior<200){
625                 bandera=0; }
626             if(theta_anterior>(theta_actual+80) && bandera==0){ //detectar si
                se dio una revolucion positiva
627                 counter=counter+1;
628                 bandera=1; }
629             if((theta_anterior+80)<theta_actual && bandera==0){ //detectar si

```

```
        se dio una revolucion negativa
630         counter=counter-1;
631         bandera=1; }
632     theta_anterior=theta_actual;
633     posicion = theta_actual + counter*360 ;
634     theta_actual=data_giroscopio();
635     conver2(posicion-PosInicial);
636     if(enviar()==0){
637         conf_USART(); }
638     //Si el valor del PWM es menor a 255 el motor gira en un sentido.
639     if(SUMAPWM<256){
640         OCR2B=0;
641         OCR2A=SUMAPWM; }
642     //El valor del PWM es mayor a 255 el motor gira en sentido
        contrario.
643     else{
644     if(SUMAPWM>256){
645         OCR2A=0;
646         OCR2B=SUMAPWM-300;
647     }
648     }
649     }
650     }
651 }
```

Apéndice D

Instalación de paquetes complementarios en Matlab Simulink

Los paquetes adicionales de Simulink no son más que complementos para que el software Matlab reconozca la placa Arduino Mega, y que además pueda operar comandos o bloques del propio Matlab en la placa Arduino.

D.1. Instrucciones de instalación de paquetes

los pasos de instalación se muestran a continuación:

1. Antes que nada, se instalan los paquetes complementarios en la sección Add-Ons.

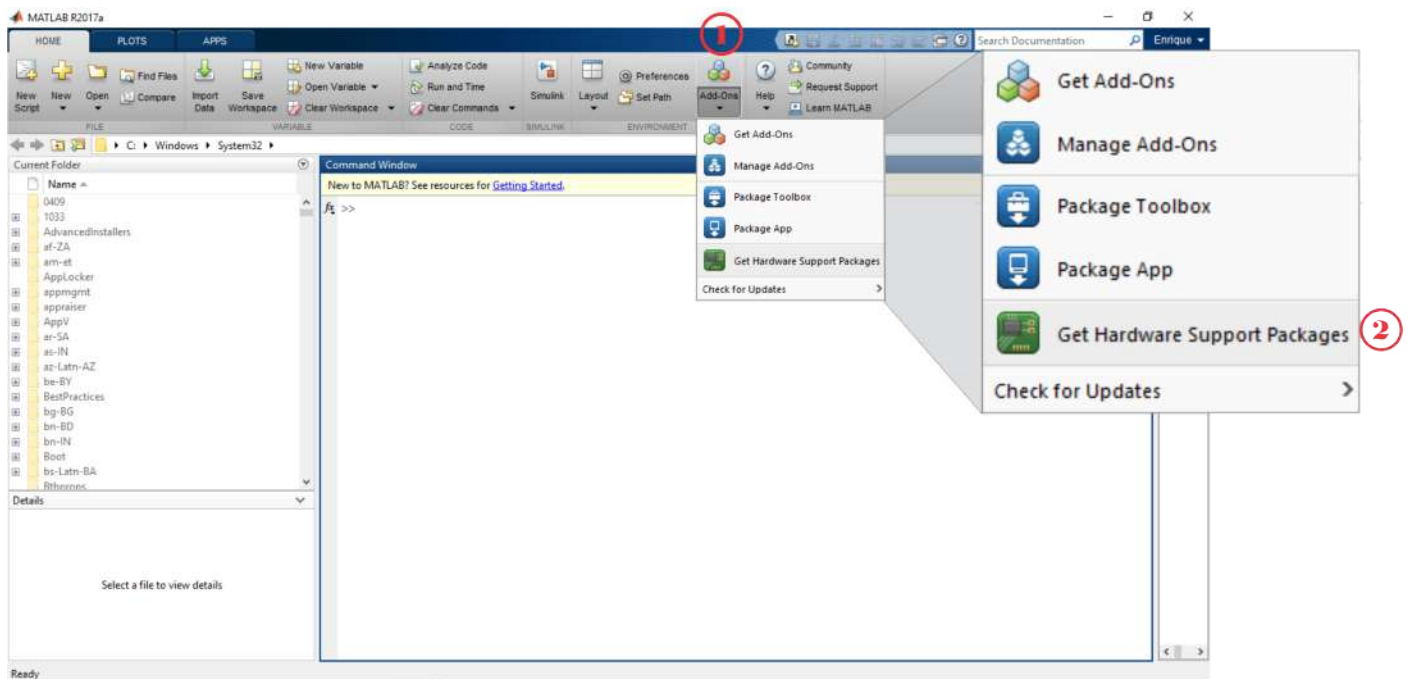


Figura D.1: Actualización de paquetes.

2. Se ven los paquetes instalados.

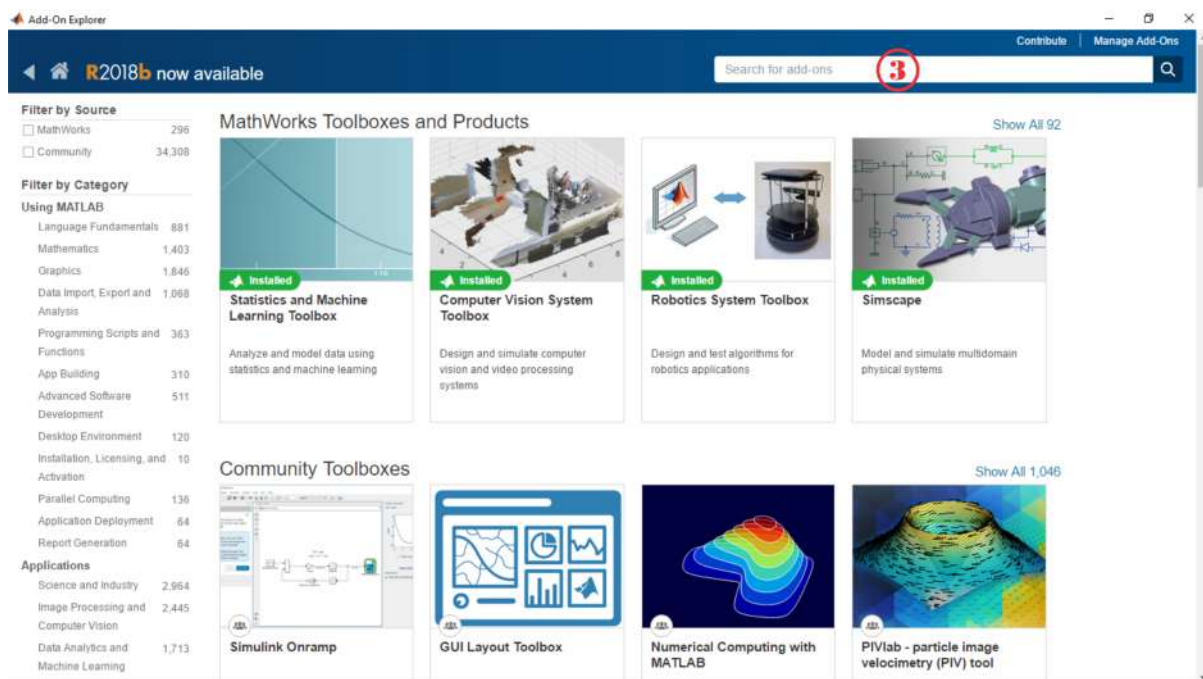


Figura D.2: Complementos instalados.

3. Se buscan los paquetes de la placa Arduino (Punto 4) en los filtros de paquetes, en donde aparecen los paquetes no instalados. Los puntos 5 y 6 son los paquetes necesarios a instalar.

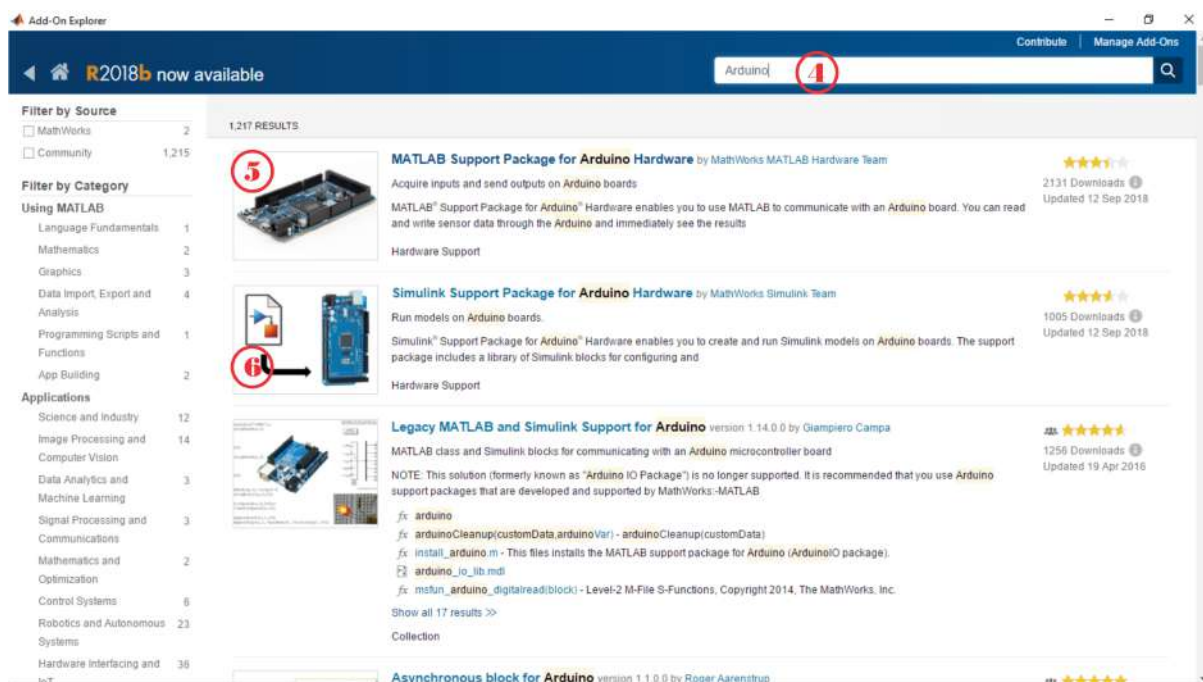


Figura D.3: Buscador de paquetes de Matlab.

4. Primero se instalan los complementos de Matlab Support Package for Arduino. (ver punto 5.A)

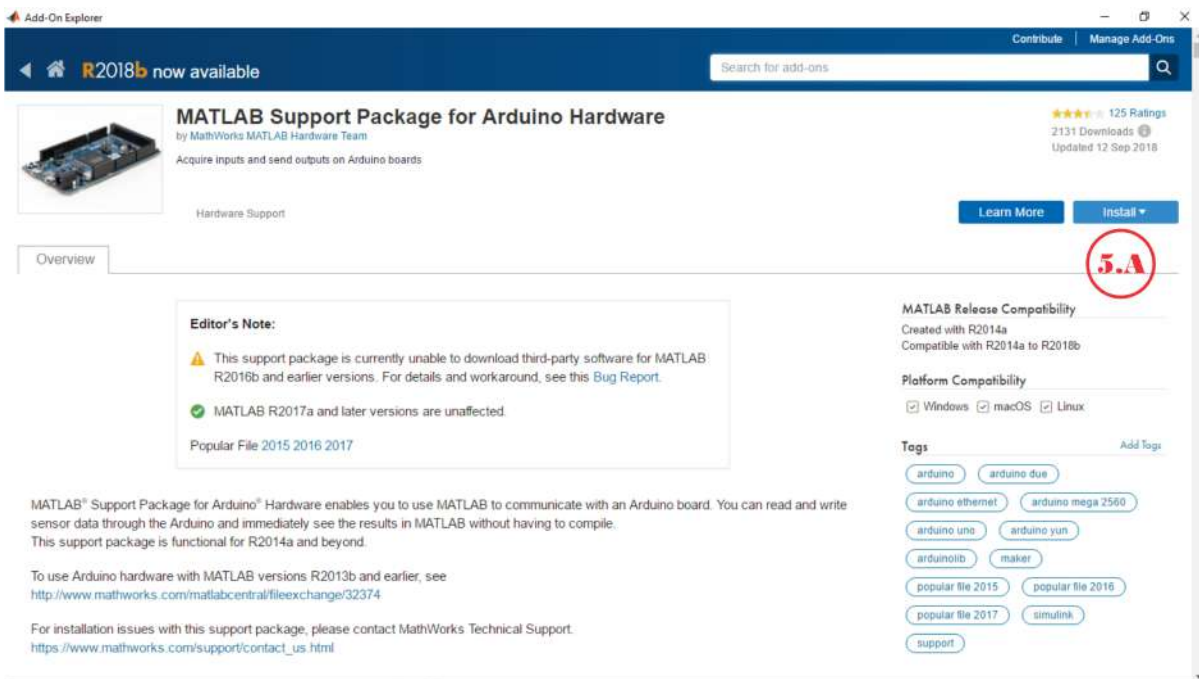


Figura D.4: Paquete Matlab Support Package.

5. En el menú habilitado se selecciona simplemente Install.

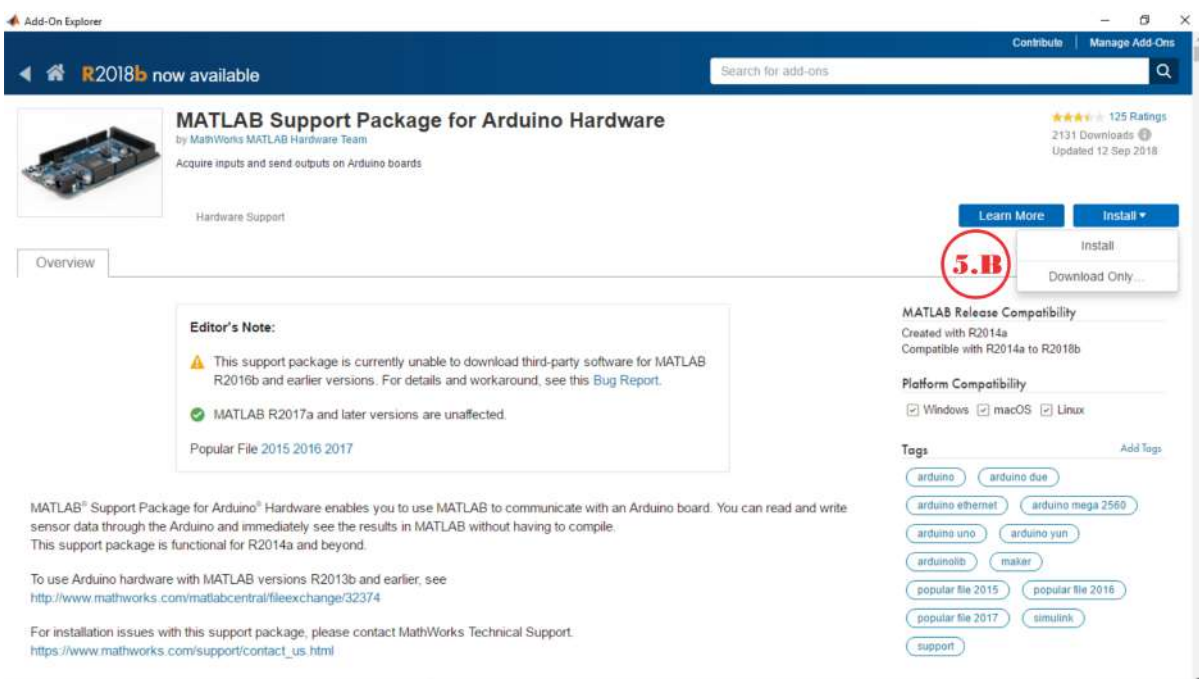


Figura D.5: Instalacion de Matlab Support Package.

6. Se aceptan los términos y condiciones que impone Matlab al paquete.

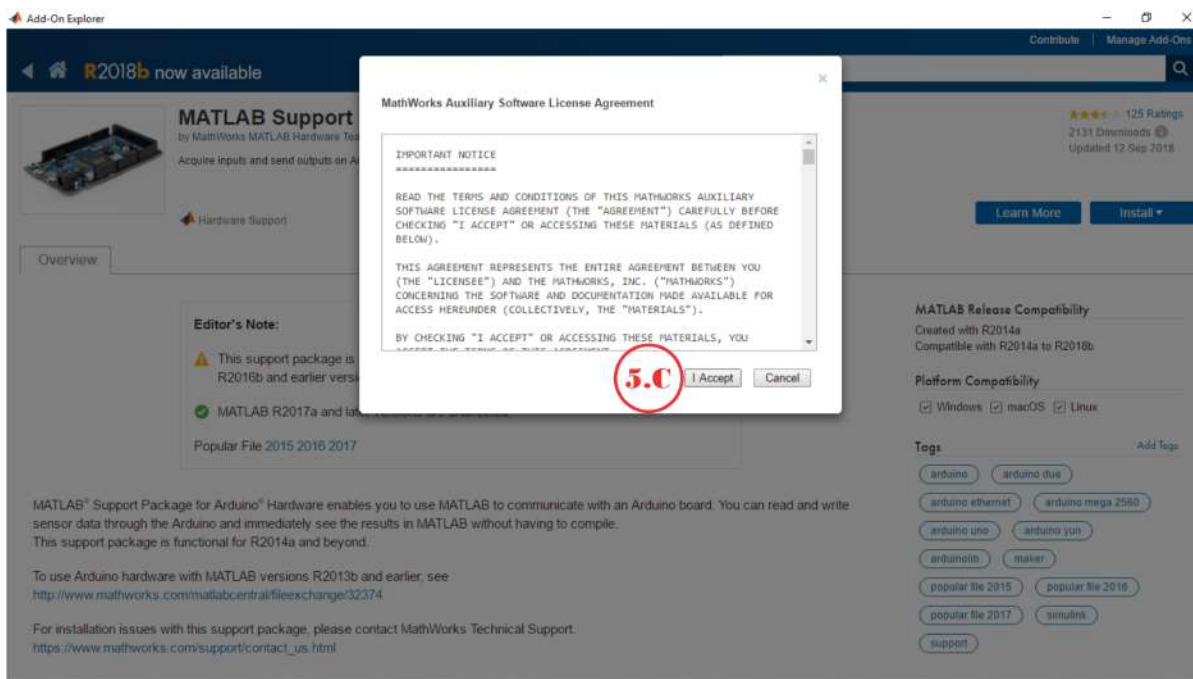


Figura D.6: Instalacion de Matlab Suport Package.

7. Se selecciona continuar a las licencias de los paquetes.

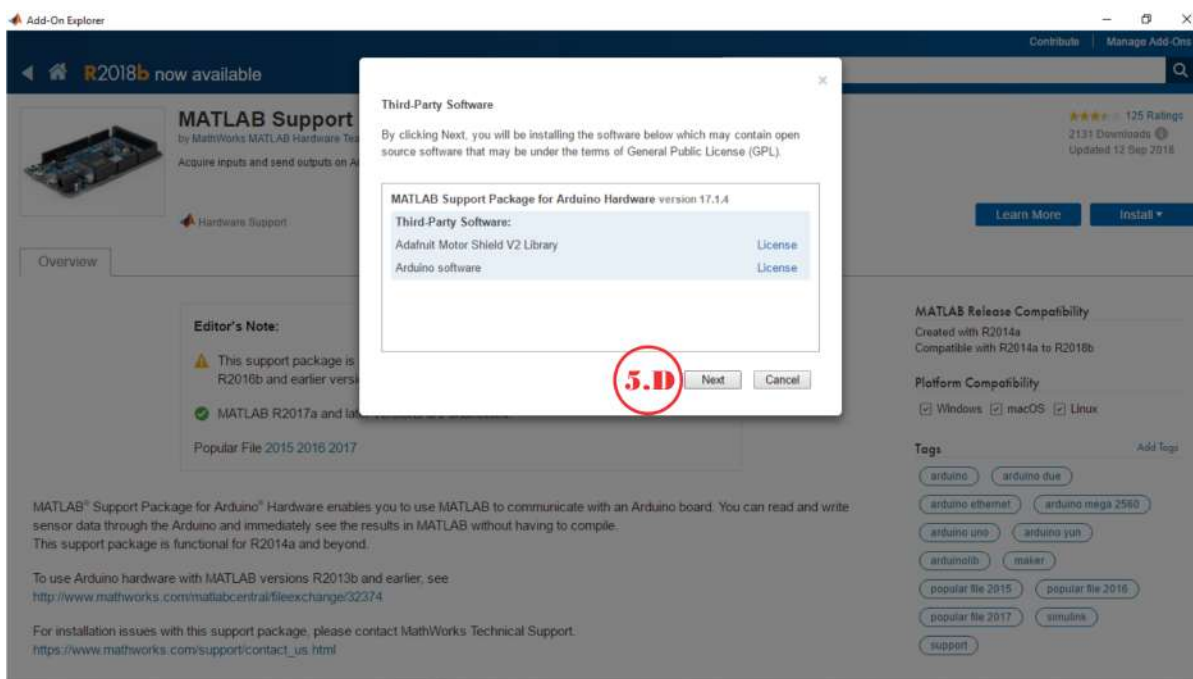


Figura D.7: Instalacion de Matlab Suport Package.

8. Al continuar, se muestra la guía de instalación.

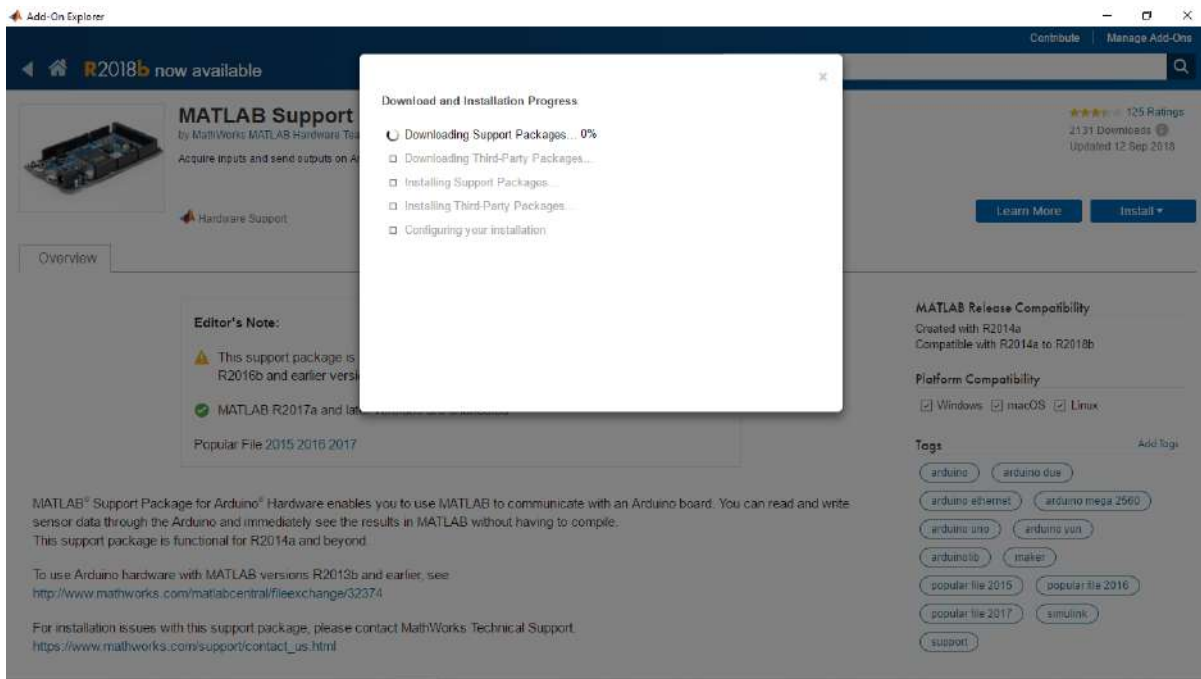


Figura D.8: Instalacion de Matlab Suport Package.

9. Se pregunta, si es de gusto configurar la instalación. En este caso se va a configurar de una vez.

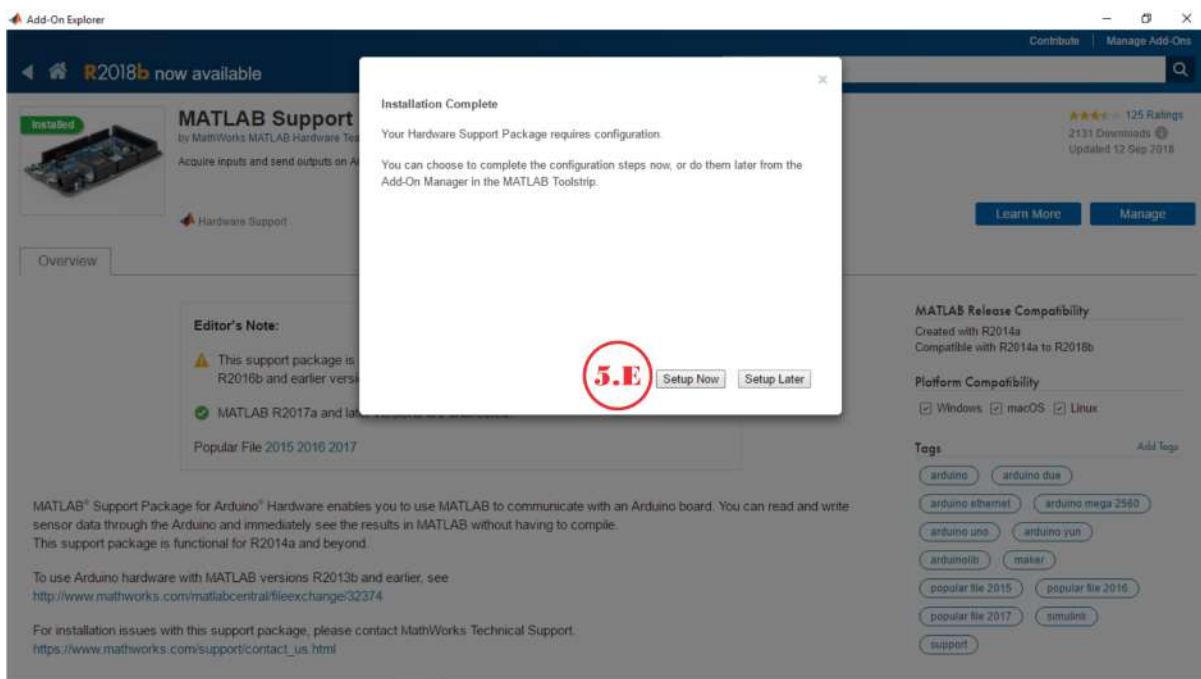


Figura D.9: Instalacion de Matlab Suport Package.

10. Se habilita la instalación del controlador de la placa Arduino.

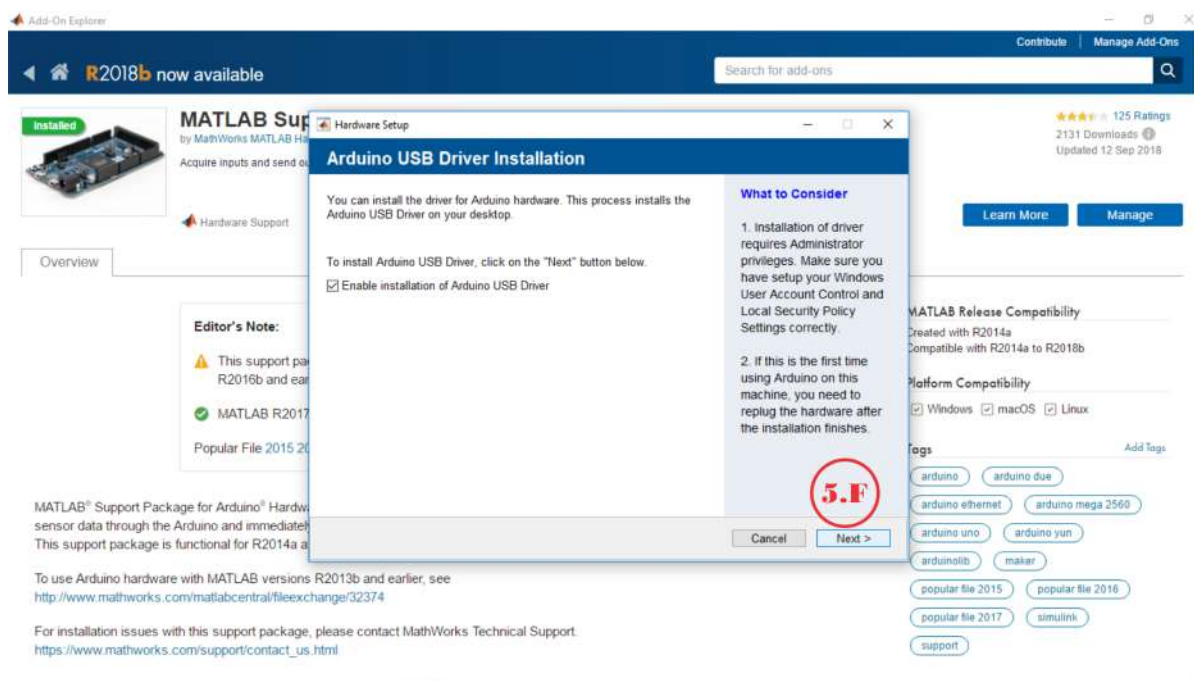


Figura D.10: Instalacion de Matlab Suport Package.

11. Se habilita la configuración de la placa Arduino.

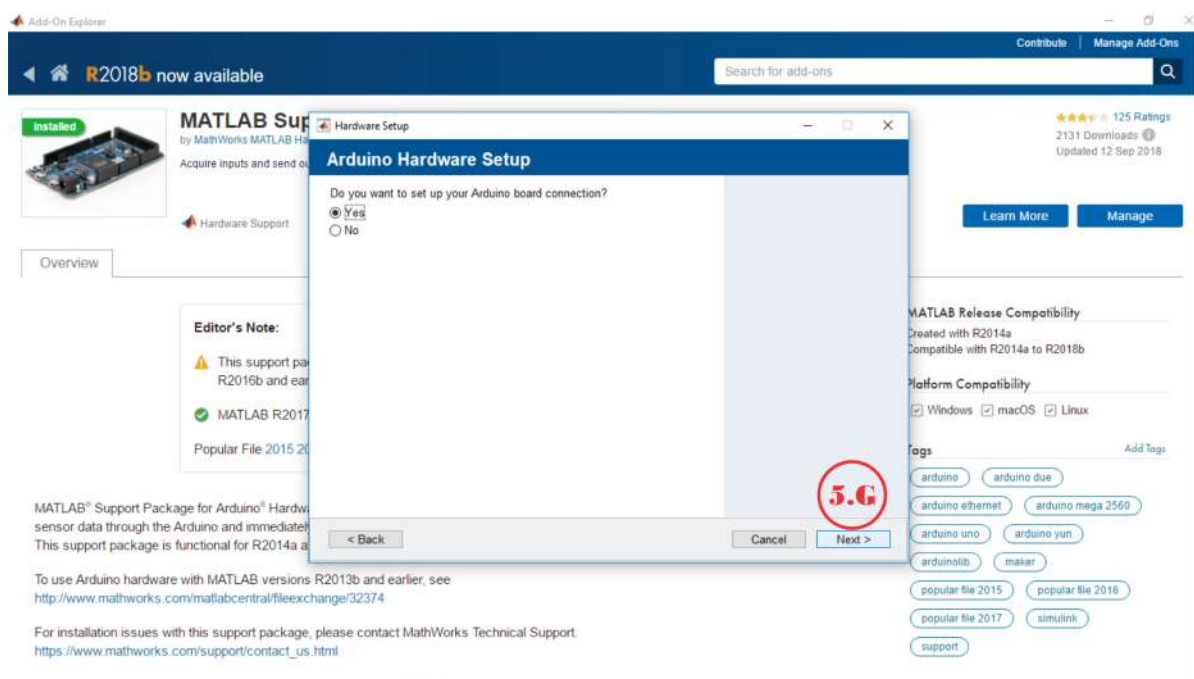


Figura D.11: Instalacion de Matlab Suport Package.

12. Se selecciona el tipo de conexión de la placa Arduino, en este caso es vía USB.

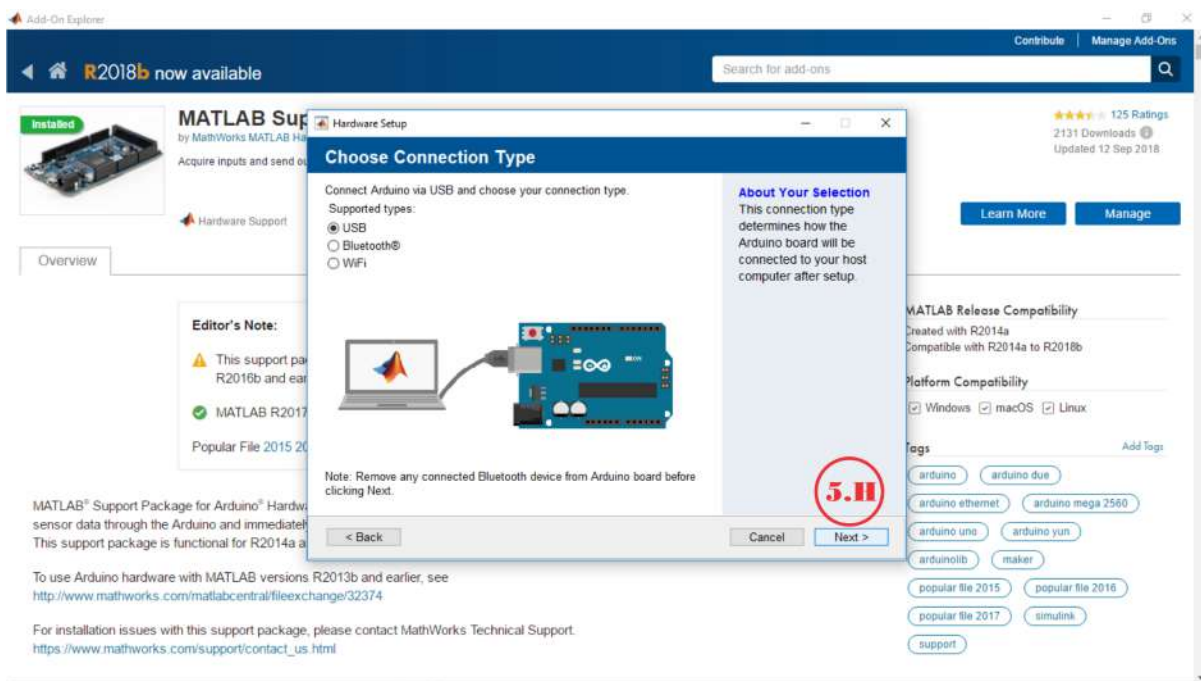


Figura D.12: Instalacion de Matlab Suport Package.

13. Se eligen las opciones, como son: la placa a conectar (punto 5.I),seleccionar el puerto de conexión (punto 5.J), elegir las librerías (punto 5.K) y finalmente programar la placa arduino.

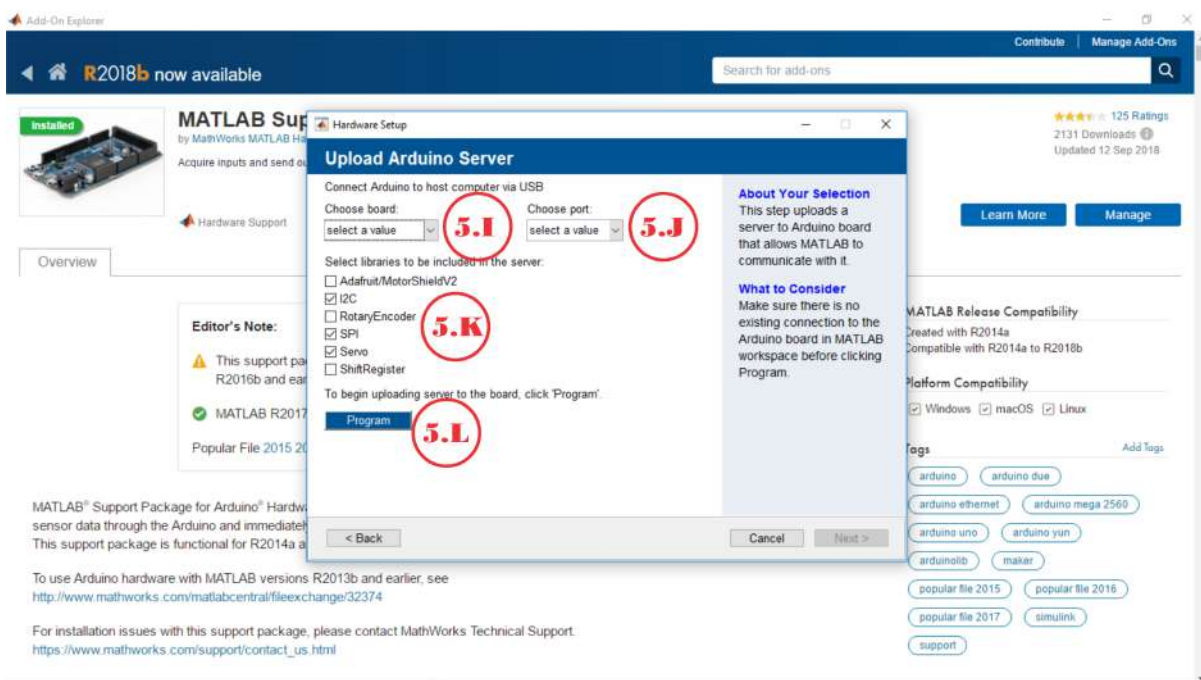


Figura D.13: Instalacion de Matlab Suport Package.

14. Después de la programación se procede a continuar (punto 5.M).

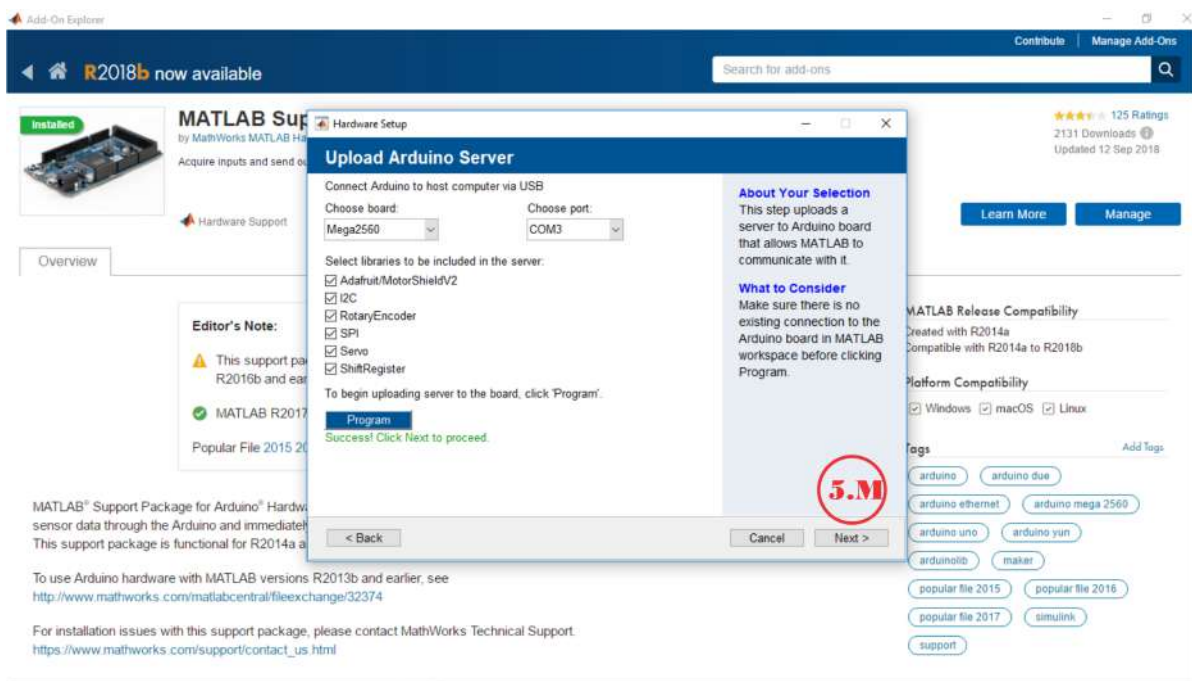


Figura D.14: Instalacion de Matlab Suport Package.

15. Al continuar se tiene que probar la conexión (punto 5.N).

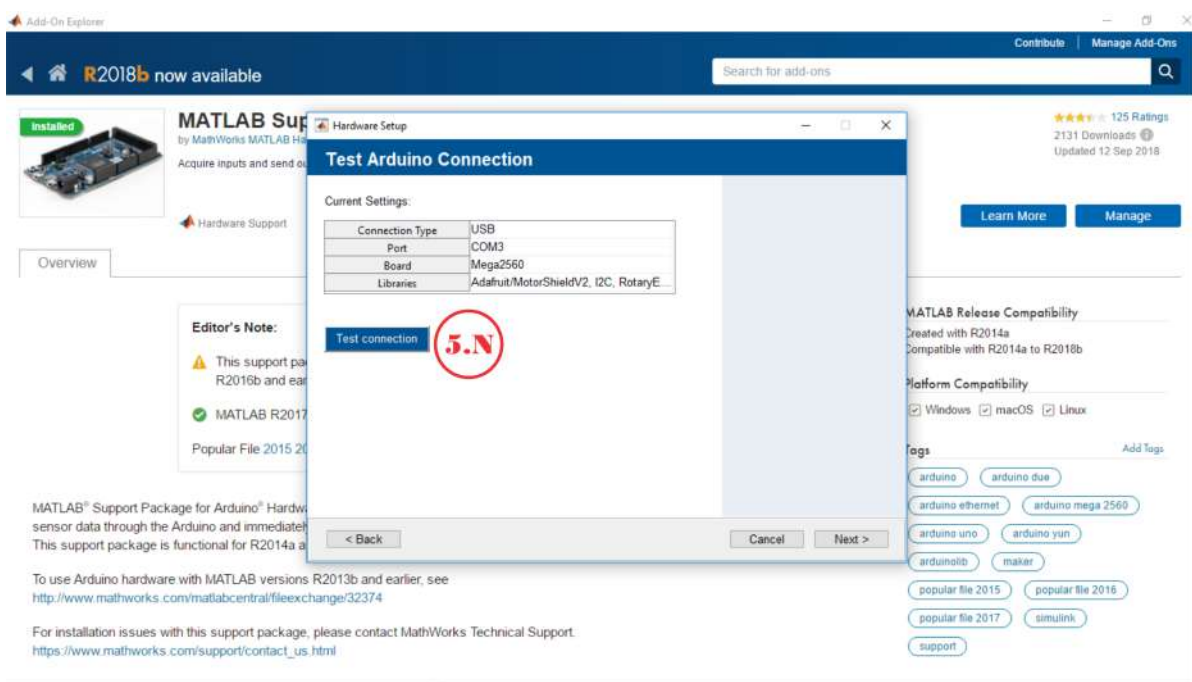


Figura D.15: Instalacion de Matlab Suport Package.

16. Proceso de conexión.

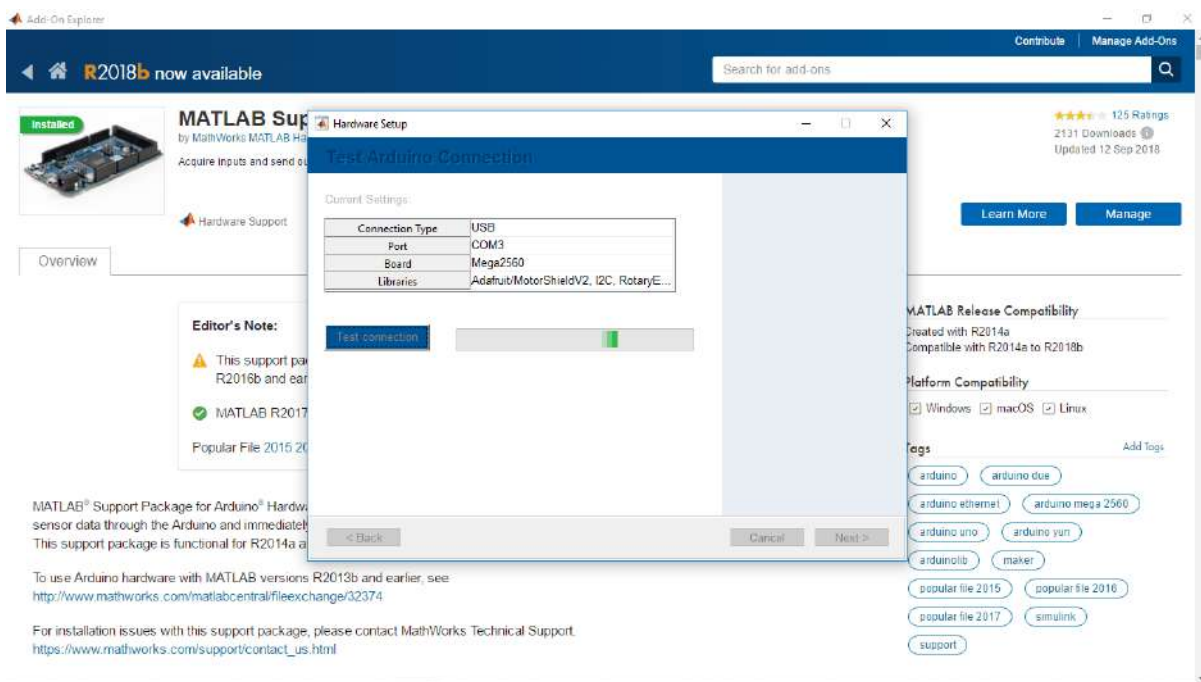


Figura D.16: Instalacion de Matlab Suport Package.

17. Al probar la conexión se procede a continuar (punto 5.Ñ).

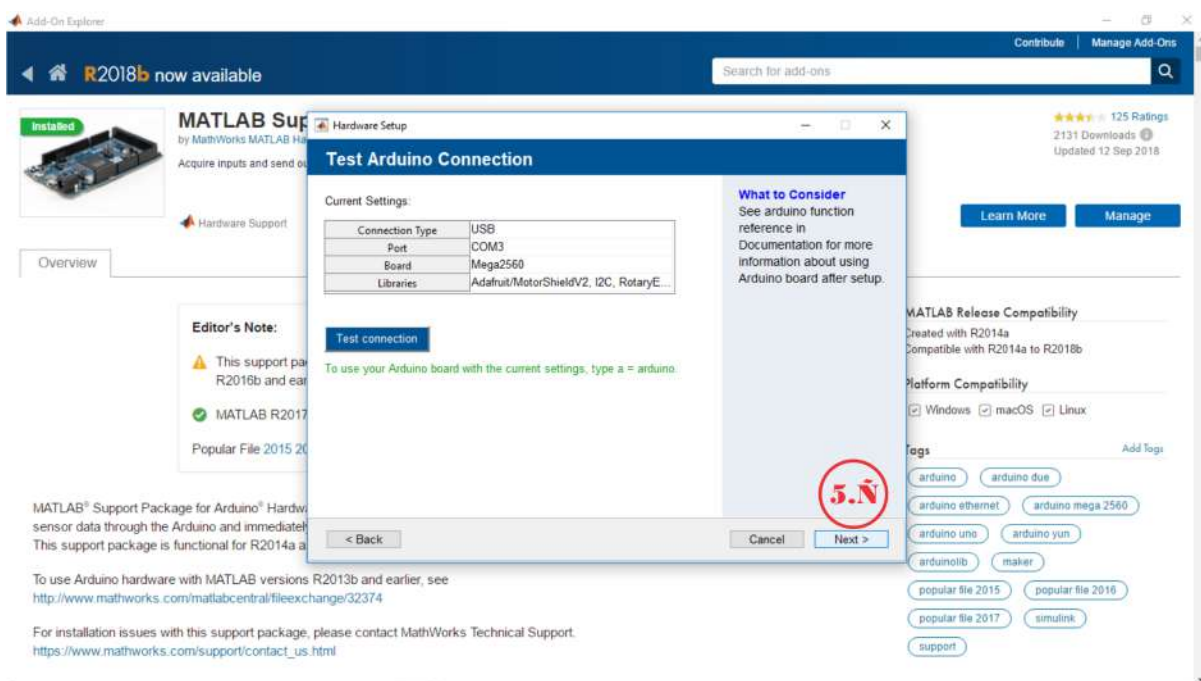


Figura D.17: Instalacion de Matlab Suport Package.

18. Se termina la instalación (punto 5.O).

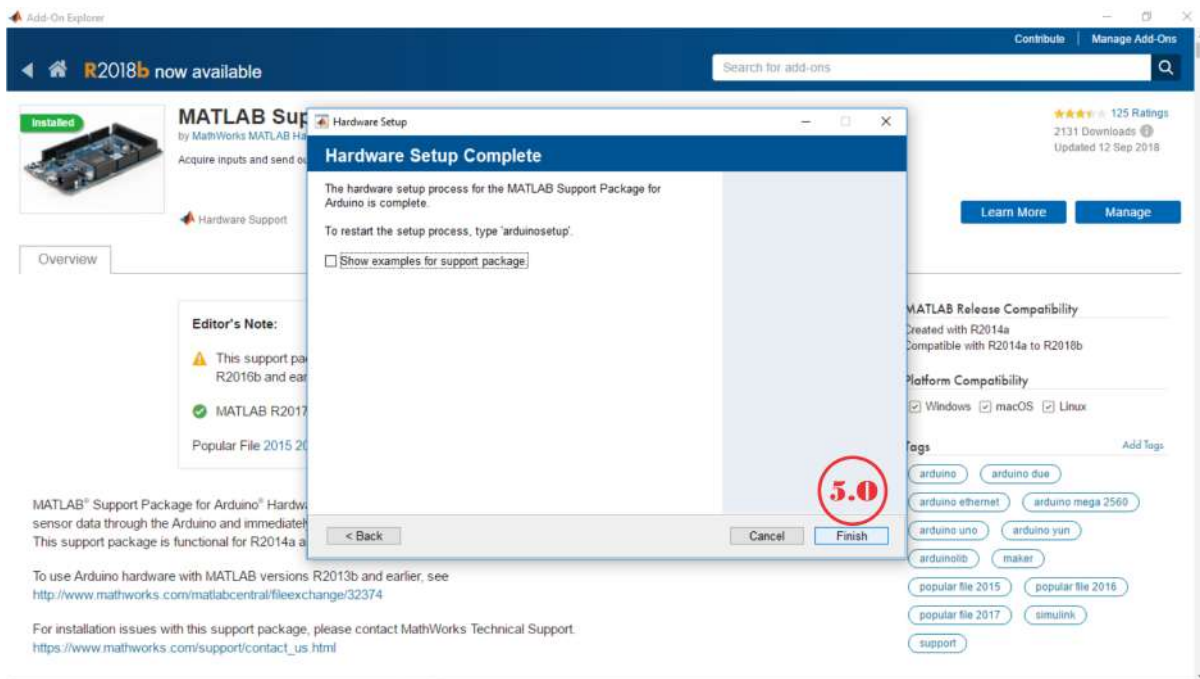


Figura D.18: Instalacion de Matlab Suport Package.

19. Se muestra que ya esta instado el paquete (punto 5.P).

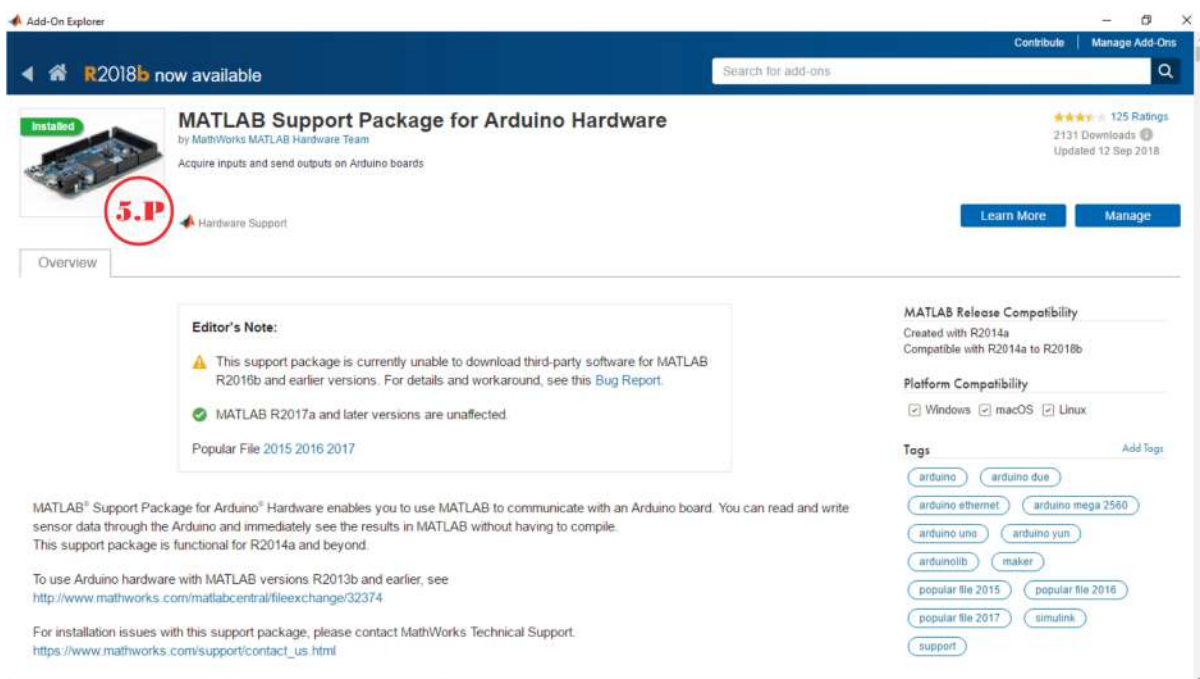


Figura D.19: Instalacion de Matlab Suport Package.

20. Se muestra la intalación de los paquetes necesarios en Matlab Simulink (punto 6.A).

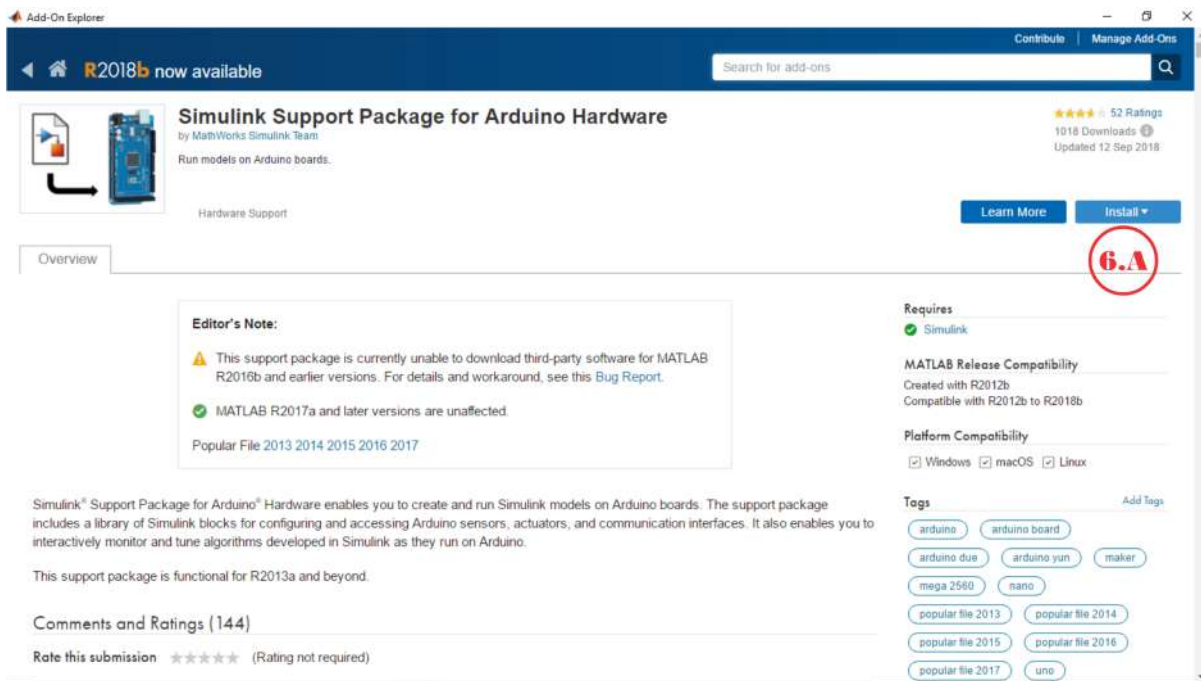


Figura D.20: Instalación de Simulink Support Package for Arduino Hardware.

21. Se selecciona Install (punto 6.B) para instalar este paquete.

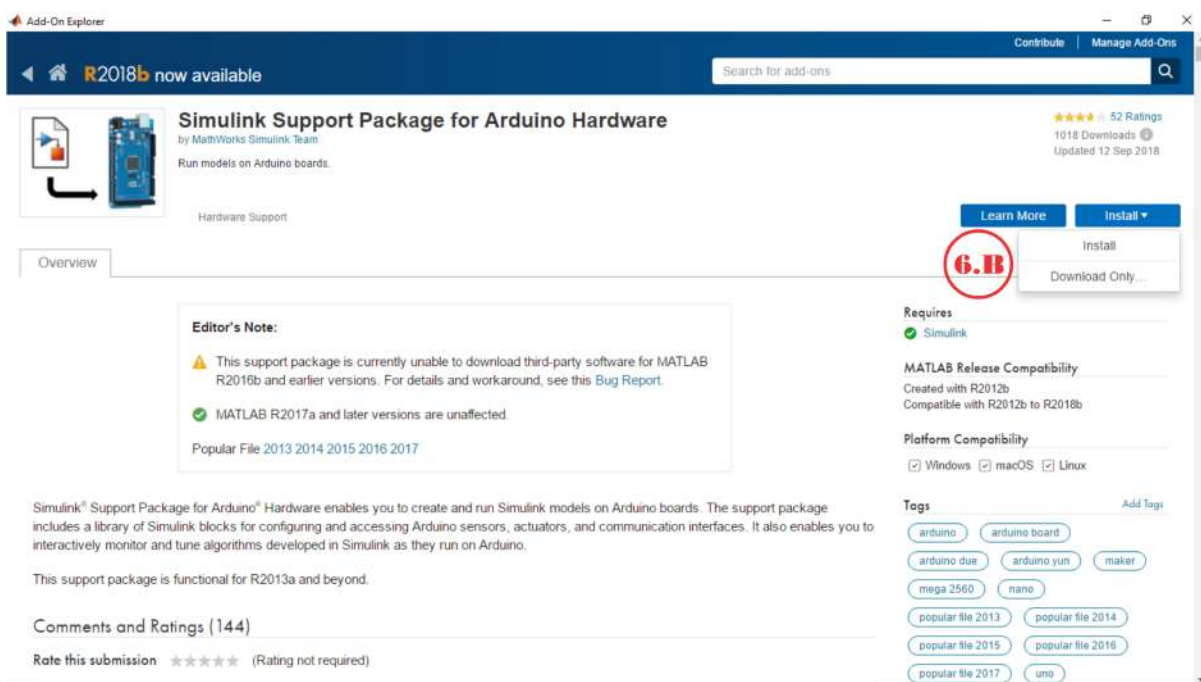


Figura D.21: Instalación de Simulink Support Package for Arduino Hardware.

22. Se aceptan los términos y condiciones de instalación de los paquetes (Punto 6.C)

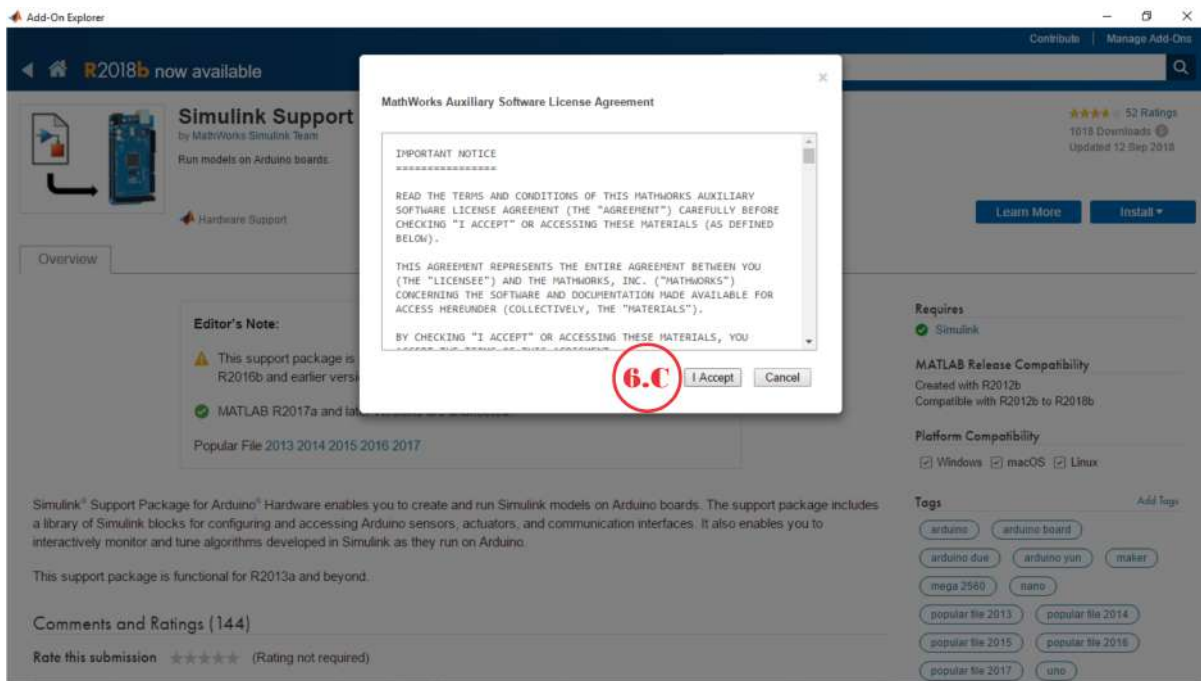


Figura D.22: Instalación de Simulink Support Package for Arduino Hardware.

23. Se aceptan las licencias de los paquetes (Punto 6.D)

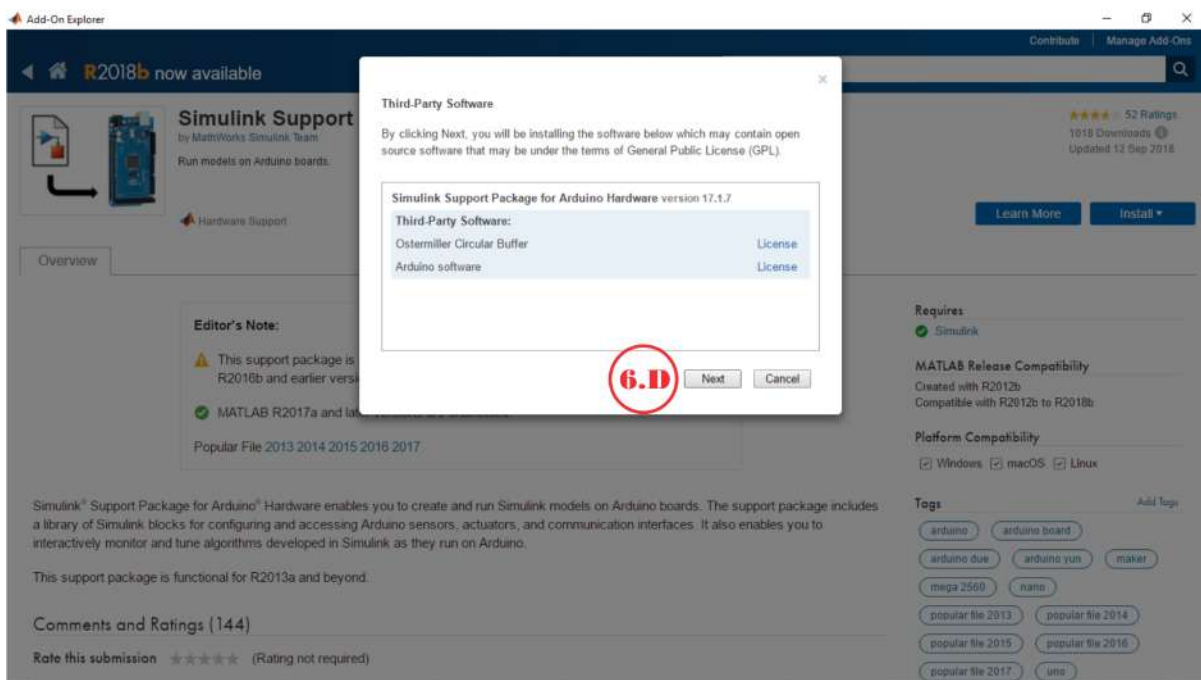


Figura D.23: Instalación de Simulink Support Package for Arduino Hardware.

24. Al aceptar las licencias, se muestra el estatus de instalación.

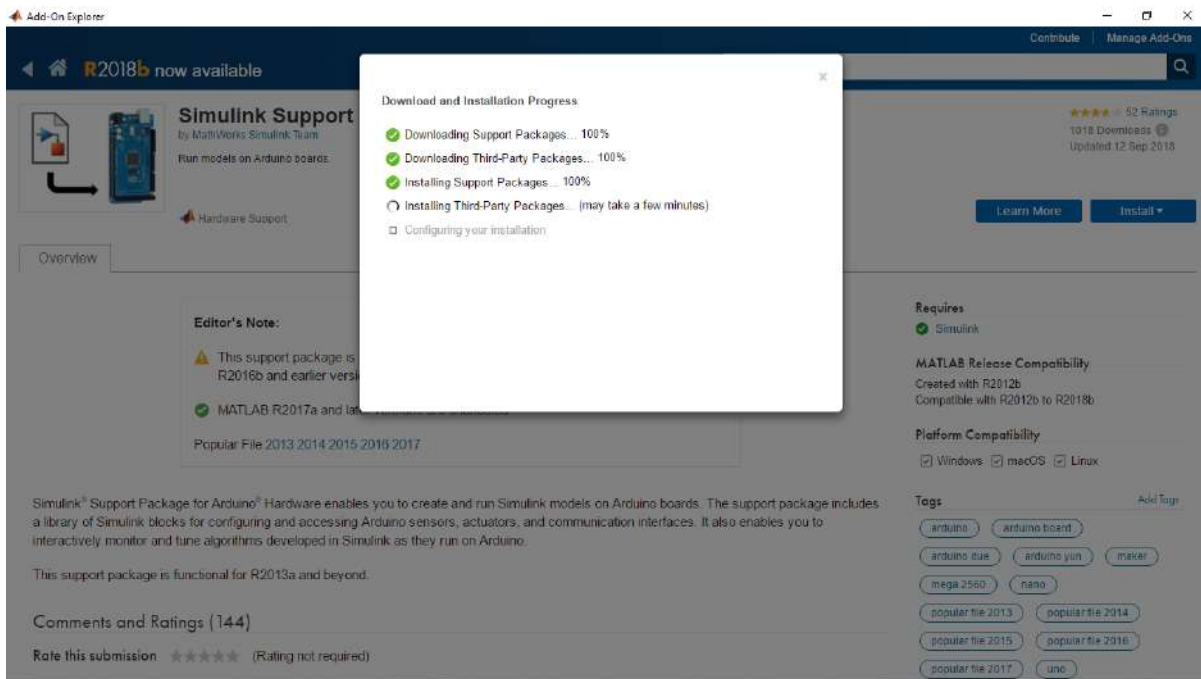


Figura D.24: Instalación de Simulink Support Package for Arduino Hardware

25. La instalación esta completada, se da en finalizar.(Punto 6.E)

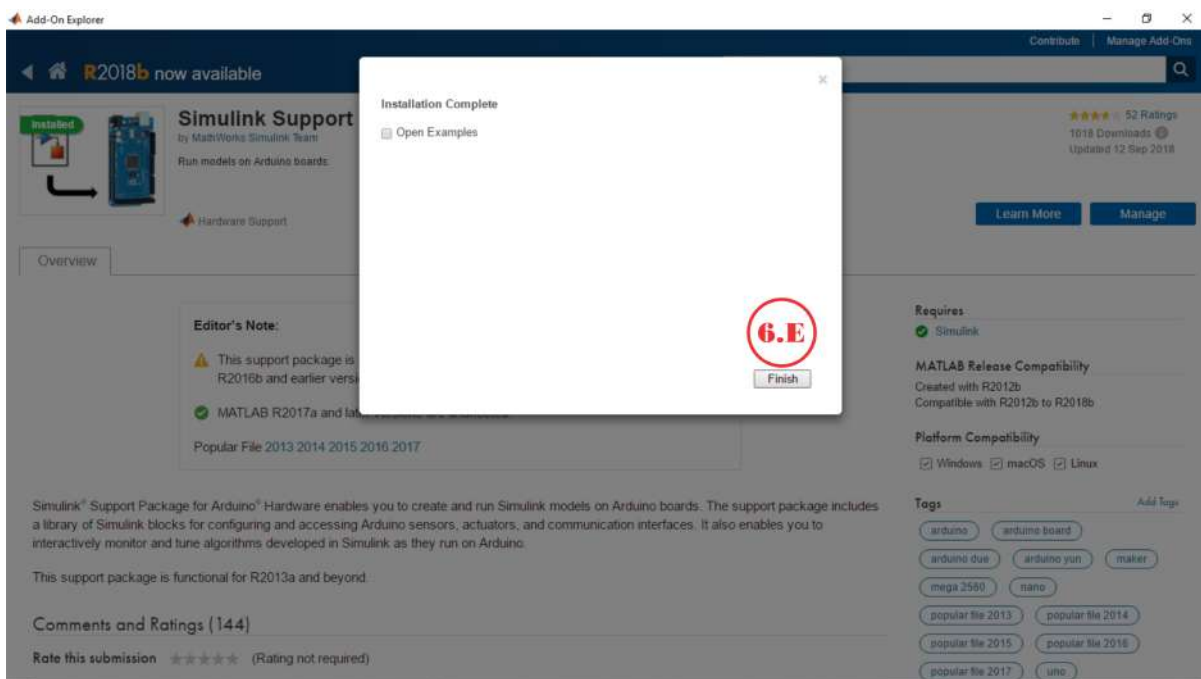
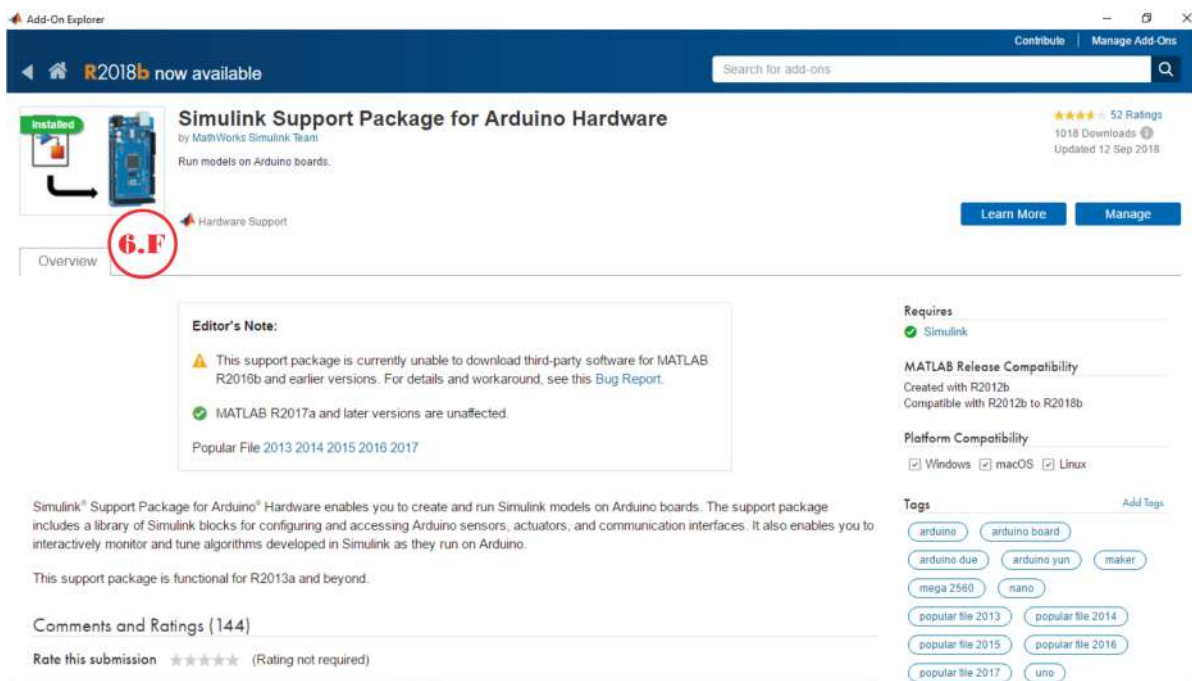


Figura D.25: Instalación de Simulink Support Package for Arduino Hardware.

26. Se observa que el paquete esta instalado(Punto 6.F)



The screenshot shows the 'Add-On Explorer' window in MATLAB. At the top, a banner reads 'R2018b now available'. Below this, the 'Simulink Support Package for Arduino Hardware' is displayed as 'Installed'. The package is by the MathWorks Simulink Team and is used to 'Run models on Arduino boards.' It has 52 ratings (4.5 stars) and 1018 downloads, last updated on 12 Sep 2018. A red circle with '6.F' is drawn around the 'Overview' tab. The 'Editor's Note' section contains a warning that the package is currently unable to download third-party software for MATLAB R2018b and earlier versions, and a green checkmark indicating that MATLAB R2017a and later versions are unaffected. The 'Requires' section shows 'Simulink' is installed. The 'MATLAB Release Compatibility' section indicates it was created with R2012b and is compatible with R2012b to R2018b. The 'Platform Compatibility' section shows it is compatible with Windows, macOS, and Linux. The 'Tags' section includes 'arduino', 'arduino board', 'arduino due', 'arduino yun', 'maker', 'mega 2560', 'nano', 'popular file 2013', 'popular file 2014', 'popular file 2015', 'popular file 2016', 'popular file 2017', and 'uno'. At the bottom, there is a 'Comments and Ratings (144)' section and a 'Rate this submission' button with a star rating and '(Rating not required)'.

Figura D.26: Instalación de Simulink Support Package for Arduino Hardware.

D.2. Verificación de paquetes instalados

Una vez instalados los paquetes es necesario validar su instalación mediante los siguientes pasos:

1. Ejecutar Simulink de la plataforma Matlab (punto 1) y seleccionar black model de la nueva pestaña (punto 2).

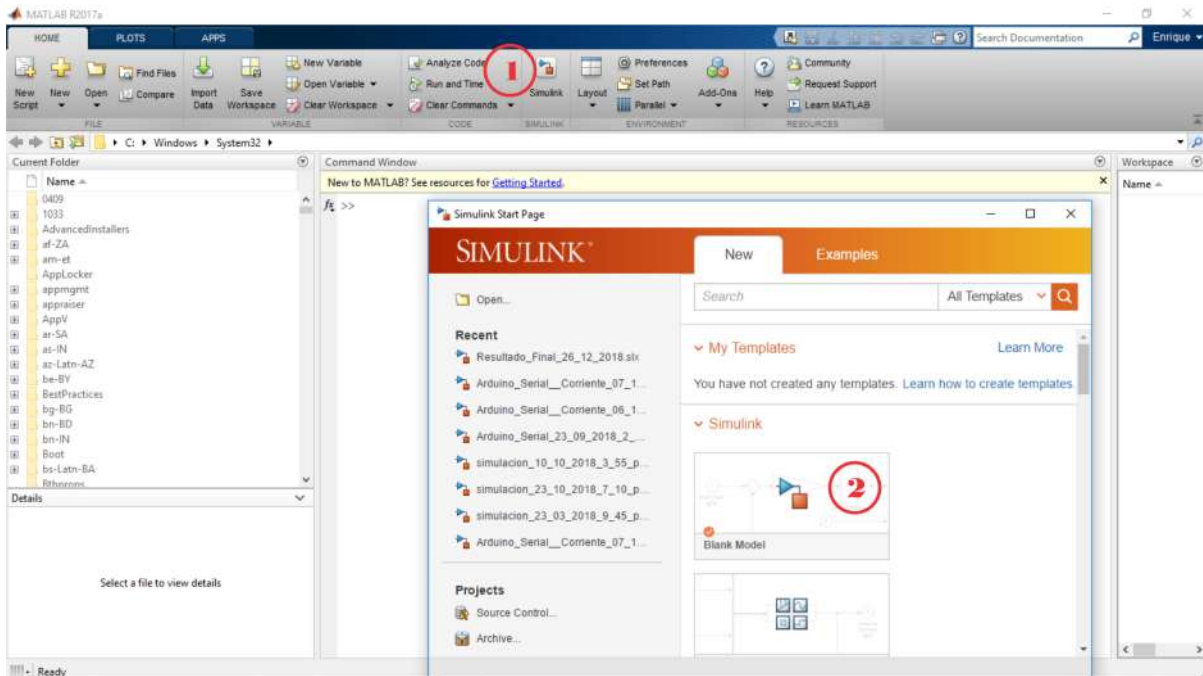


Figura D.27: Comprobación de paquetes instalados.

2. Al abrirse el documento en blanco, es necesario abrir las librerías de Simulink, es por ello que se selecciona el recuadro mostrado en el punto 3

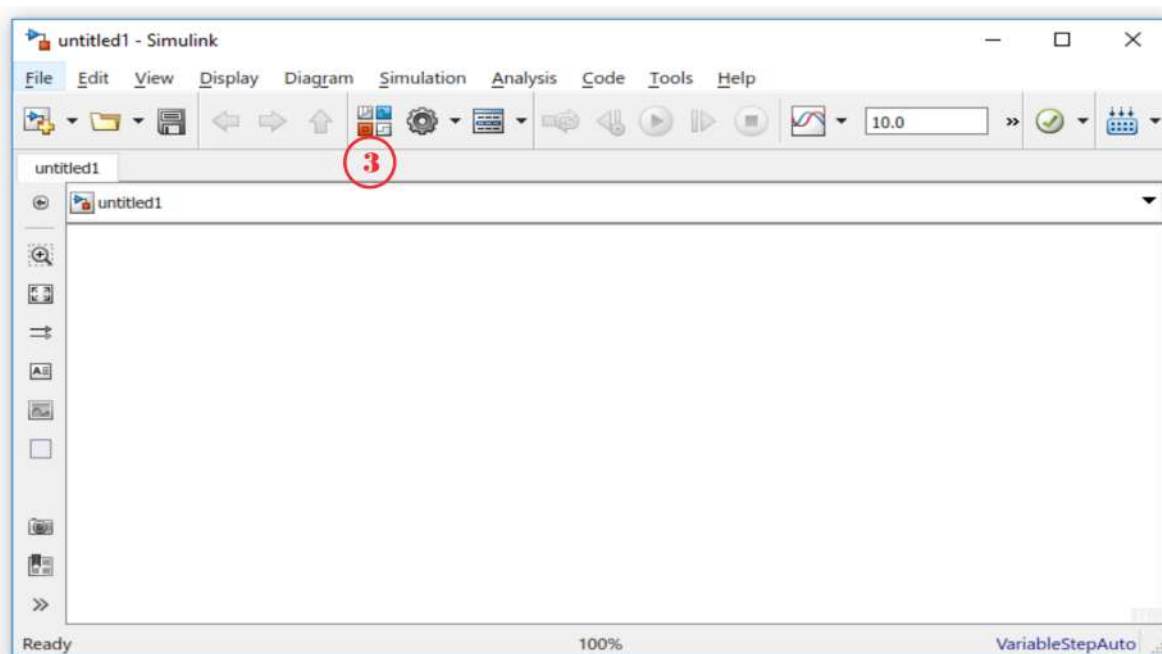


Figura D.28: Comprobación de paquetes instalados.

3. La nueva pestaña nos muestra el conjunto de librerías, Sin embargo tenemos que hallar las librerías requeridas, como se muestra en el punto 4, y esta constituido por los bloques señalados por el punto 5.

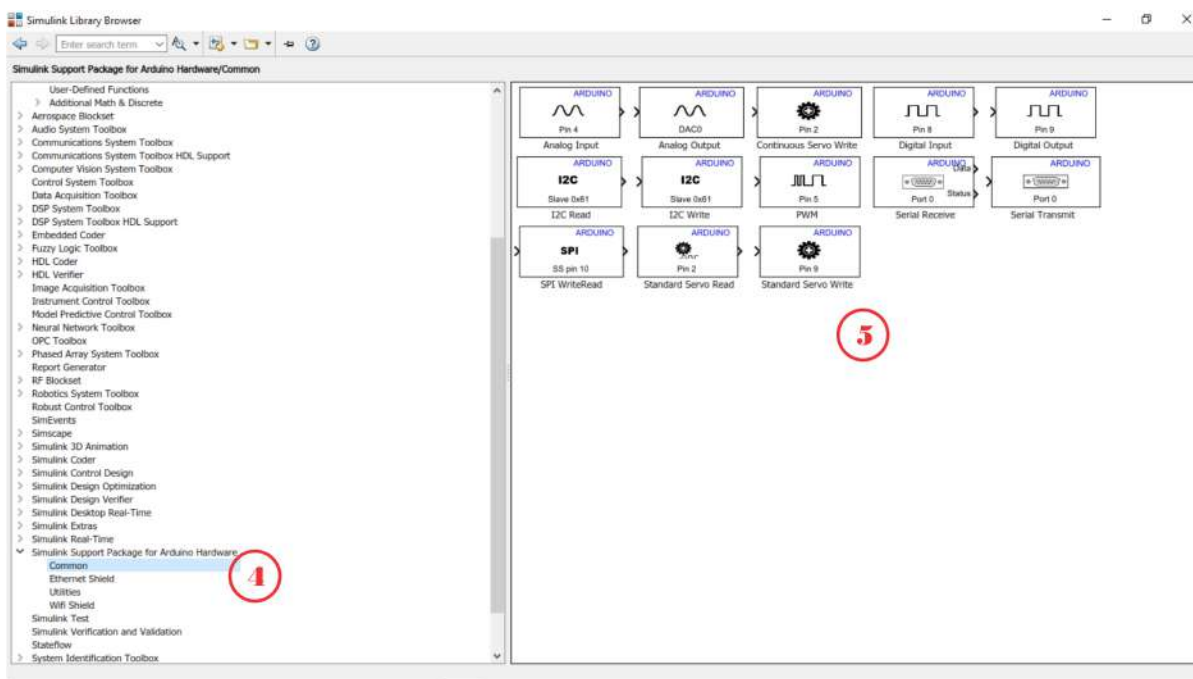


Figura D.29: Comprobación de paquetes instalados.

Apéndice E

Código para decodificación de los datos de posición

```
653 

---


654 // Variables de entrada y salidas.
655 

---


656 function [sa2, sal] = fcn(theta_actual)
657 persistent millares;
658 persistent centenas;
659 persistent decenas;
660 persistent unidades;
661 persistent value;
662 persistent bandera;
663 persistent counter;
664 persistent signo;
665 persistent numero;
666 

---


667 // Inicializacion de variables
668 

---


669 if isempty(millares)
670     millares=0;
671 end
672 if isempty(centenas)
673     centenas=0;
674 end
675 if isempty(decenas)
676     decenas=0;
677 end
678 if isempty(unidades)
679     unidades=0;
680 end
681 if isempty(value)
682     value=0;
683 end
684 if isempty(bandera)
685     bandera=0;
686 end
687 if isempty(signo)
```

```

688     signo=1;
689 end
690 if isempty(numero)
691     numero=0;
692 end
693
694 //          Validacion y decodificacion de datos recibidos
695
696 if(theta_actual==111 && bandera==0) // Verificar si se recibio una "o".
697     bandera=1;
698 end
699 if(theta_actual==112) // Verificar si se recibio una "p".
700     bandera=0;
701 end
702 if(theta_actual==45) // Verificar si se recibio el signo "-".
703     signo=-1;
704     bandera=2;
705 end
706 if(theta_actual==43) // Verificar si se recibio el signo "+".
707     signo=1;
708     bandera=2;
709 end
710     if(theta_actual<58 && theta_actual>47 && 1<bandera && bandera<6)
711         if(bandera==2) // Generacion de los millares.
712             value=(theta_actual-48)*1000;
713             bandera=3;
714         else
715             if(bandera==3) // Generacion de las centenas.
716                 value=(theta_actual-48)*100 + value;
717                 bandera=4;
718             else
719                 if(bandera==4) // Generacion de las decenas.
720                     value=(theta_actual-48)*10 + value;
721                     bandera=5;
722                 else
723                     if(bandera==5) // Generacion de las unidades.
724                         value=(theta_actual-48) + value;
725                         bandera=6;
726                     end
727                 end
728             end
729         end
730     end
731 if(bandera==6) // Se recibio el dato entero. Si existe "-", cambiar el signo.
732     numero=signo*value;
733 end
734 if(bandera==1 && theta_actual==111) // Se recibio un dato de inicio, "o"
735     bandera=2; // Incrementar bandera para recibir el primer digito.
736 end
737 counter=numero; // Asignar el valor recolectado a la prevariable de salida.
738 sa2=signo;
739 sal=counter; // Se le asigna la salida a "sal".

```

Apéndice F

Código para codificación de PWM

```
740
741 // Variables de entrada y salidas.
742
743 function [sali , sal , sal1 , sal2 , sal3 , total] = fcn(theta_actual , derivada)
744 persistent desfase;
745 persistent bandera;
746 persistent centenas;
747 persistent decenas;
748 persistent unidades;
749 persistent valor;
750
751
752 // Inicializacion de variables
753
754 if isempty(valor) // Primera vez que inicia el programa, se hacen
755     inicializaciones
756     valor=0;
757     end
758     if isempty(desfase)
759         desfase=0;
760     end
761     if isempty(bandera)
762         bandera=0;
763     end
764
765 // Toma de decisiones
766
767 if((-0.0001<derivada<0.0001) && theta_actual>255 && bandera==0) // Ubicar
768     la razon de cambio.
769     bandera=1;
770     desfase=theta_actual;
771     end
772 if(theta_actual>255 && derivada>0) // Truncar el valor de PWM sin cambio
773     de pendiente.
774     valor=255;
775 else
776     if(theta_actual>255 && derivada<0) // Truncar el valor de PWM con
```

```

774         cambios de pendiente.
775         valor=255+(theta_actual-desfase);
776         if(valor < 0)
777             valor=-1*valor + 300;
778             if(valor > 554)
779                 valor = 554;
780             end
781         else
782             if(theta_actual < 0) %PWM negativo.
783                 valor=-1*theta_actual + 300; // Interpretar el PWM negativo.
784                 if(valor > 554)
785                     valor = 554;
786                 end
787             else
788                 valor=theta_actual; // Valor de PWM dentro del rango -255 a
789                                     255.
790             end
791         end
792         if(valor > 554)
793             valor = 255;
794         end
795
796 //      Validacion y codificacion de datos recibidos
797
798     centenas=valor - mod(valor,100); // Se restan las decenas con unidades,
799         centenas.
800     centenas=centenas/100;
801     sobrante= mod(valor,100); // Son las decenas y unidades.
802     decenas=sobrante-mod(sobrante,10);
803     decenas=decenas/10;
804     sobranteunidades=mod(sobrante,10); // Unidades con decimales.
805     unidades=sobranteunidades-mod(sobranteunidades,1);
806     sali=10;
807     sal=centenas+48;
808     sal1=decenas+48;
809     sal2=unidades+48;
810     sal3=46;
811     total=centenas*100 + decenas*10 + unidades;

```

Apéndice G

Configuración de los parámetros de simulación y simulación en tiempo real

La configuración de los parámetros se muestra a continuación:

1. Se da click en el punto 1 mostrado en la Figura G.1

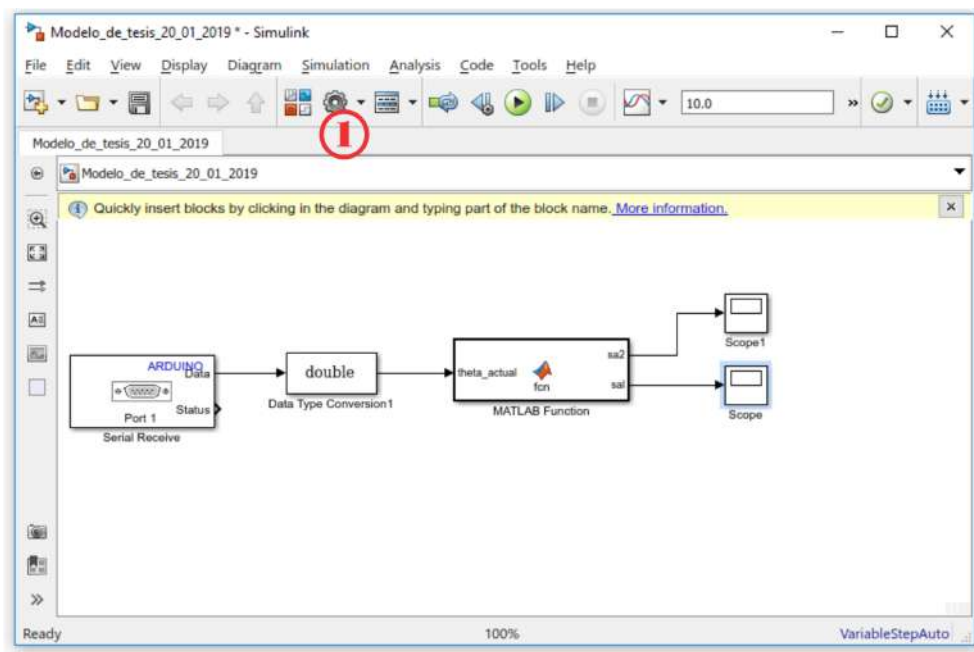


Figura G.1: Modificación de parámetros.

2. Se eligen las opciones mostradas.

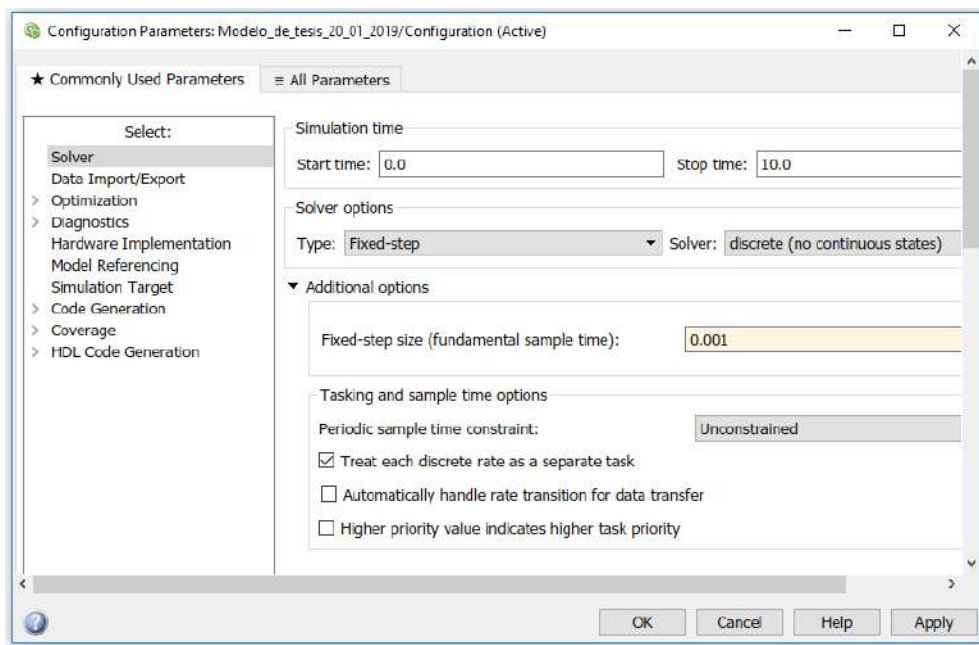


Figura G.2: Modificación de parámetros.

3. Se eligen las opciones mostradas.

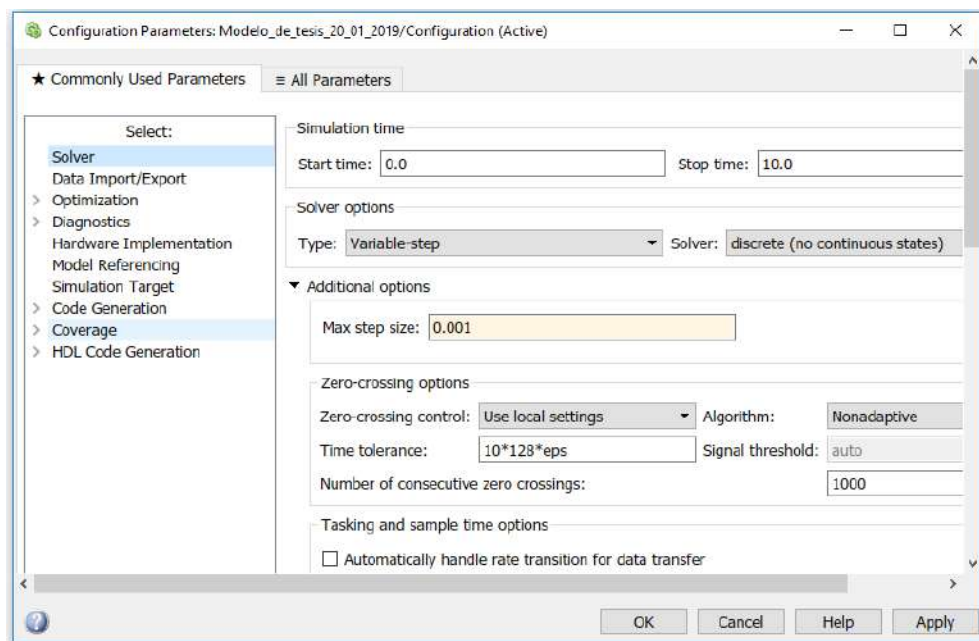


Figura G.3: Modificación de parámetros.

4. Se eligen las opciones mostradas.

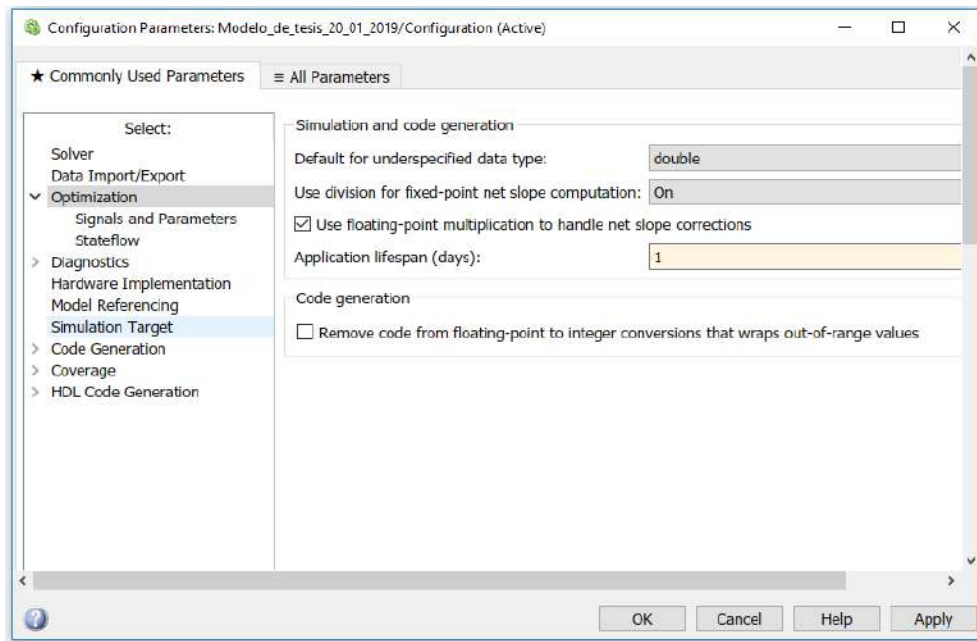


Figura G.4: Modificación de parámetros.

5. Se eligen las opciones mostradas.

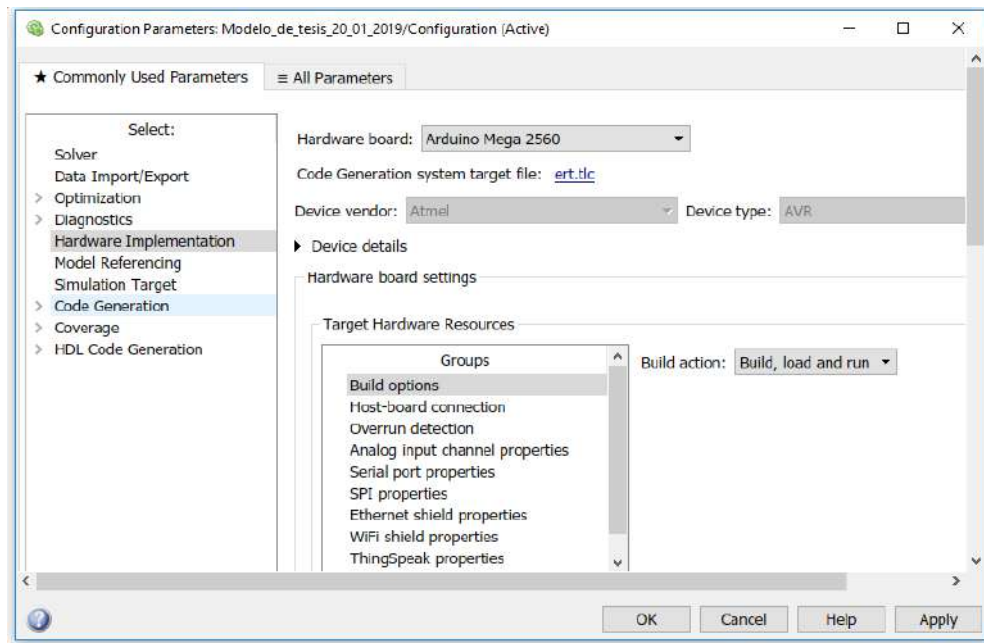


Figura G.5: Modificación de parámetros.

6. Se eligen las opciones mostradas.

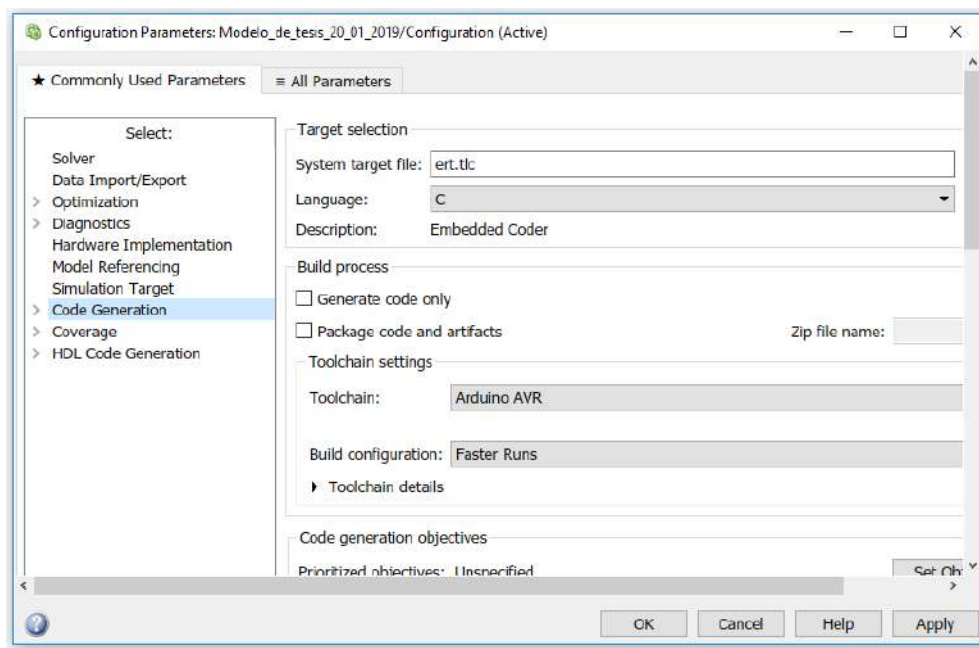


Figura G.6: Modificación de parámetros.

7. Se termina la configuración (ver el punto 2)

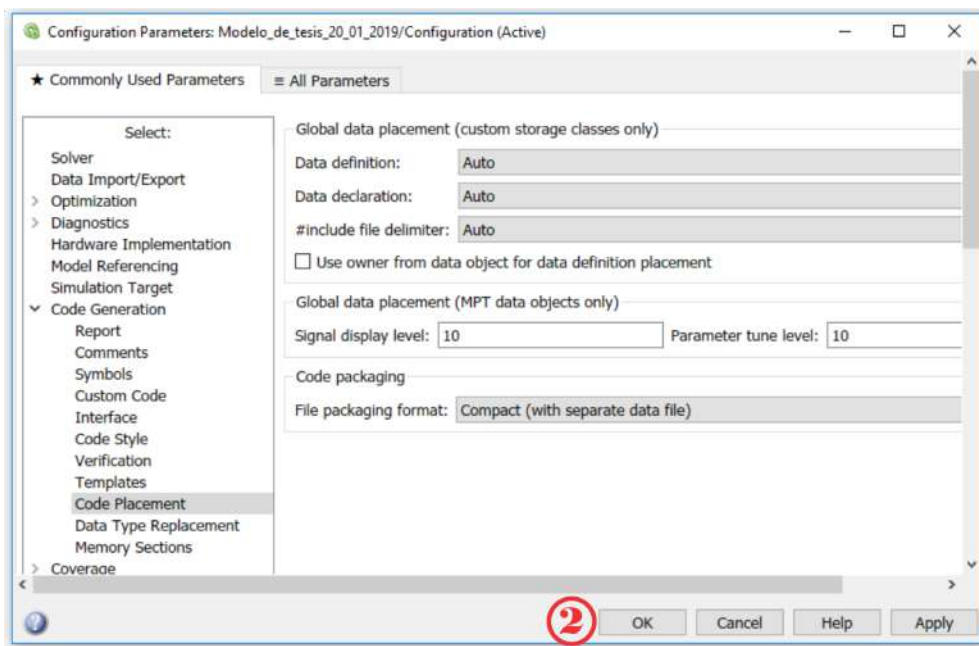


Figura G.7: Modificación de parámetros.

Debido al apartado mostrado por la Figura G.2 de la configuración de los parámetros, se abordo que el tiempo del solver discreto es de 0.001s es decir una frecuencia de 1KHz, por

lo tanto la frecuencia máxima para adquisición de los datos deberá ser esa. Siendo esto así, es necesario cambiar el parámetro de la adquisición de datos de Arduino, tal y como se logra visualizar en la Figura G.8.

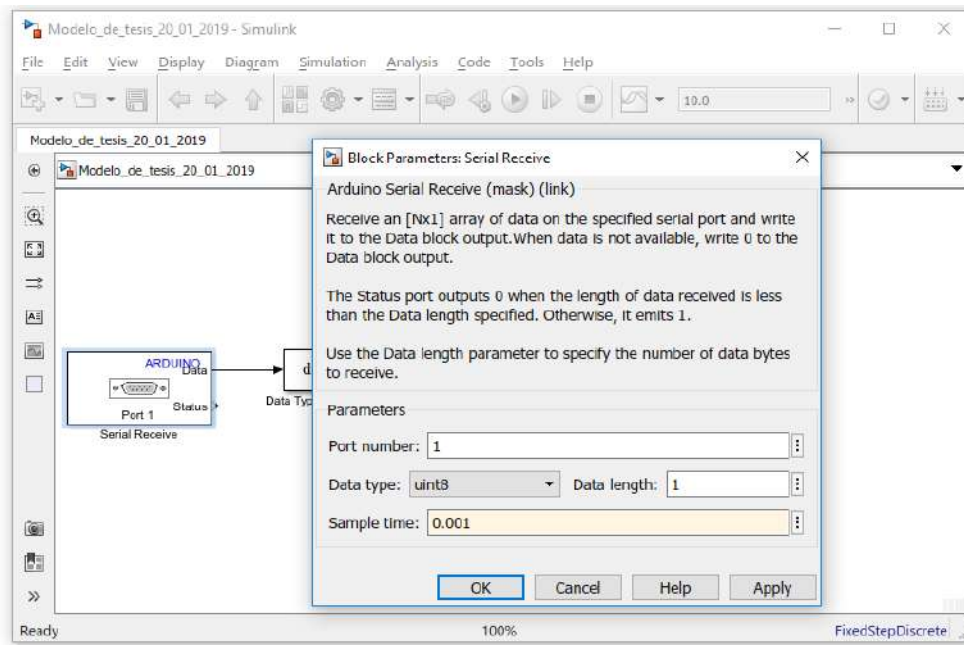


Figura G.8: Cambio de la velocidad de muestro de Arduino.

Debido a que se necesita trabajar en tiempo real, es necesario simular todos los diagramas a bloques dentro de la placa Arduino, es por ello que se hace la siguiente modificación en la forma de simulación, esto es mostrado en la Figura G.9 en donde el punto 1 muestra el submenú anexo y el punto 2, se muestra el tipo de simulación que en este caso es externo, y el tiempo de simulación ahora sera infinito.

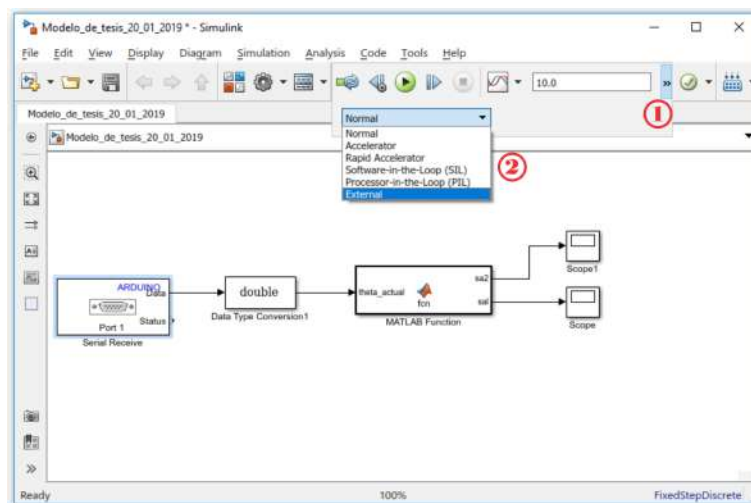


Figura G.9: Modificación de parámetros.

Por lo tanto el diagrama a bloque con su debido tipo de simulación queda dado por la Figura G.10.

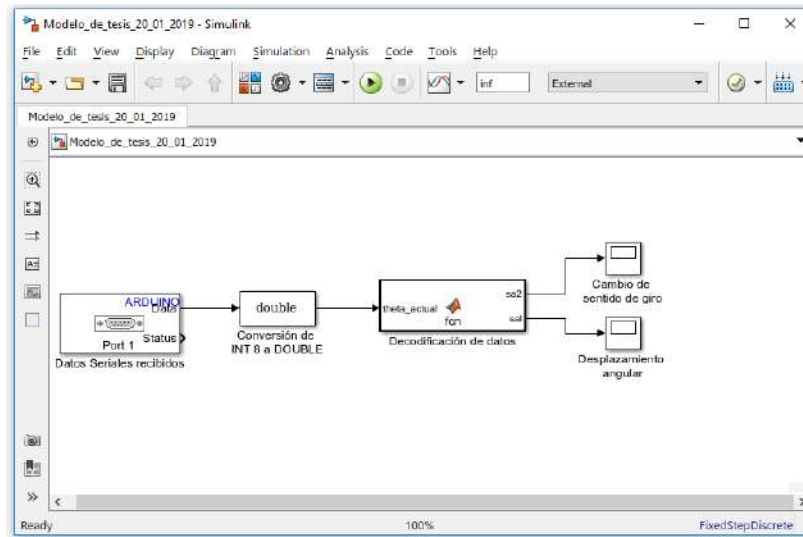
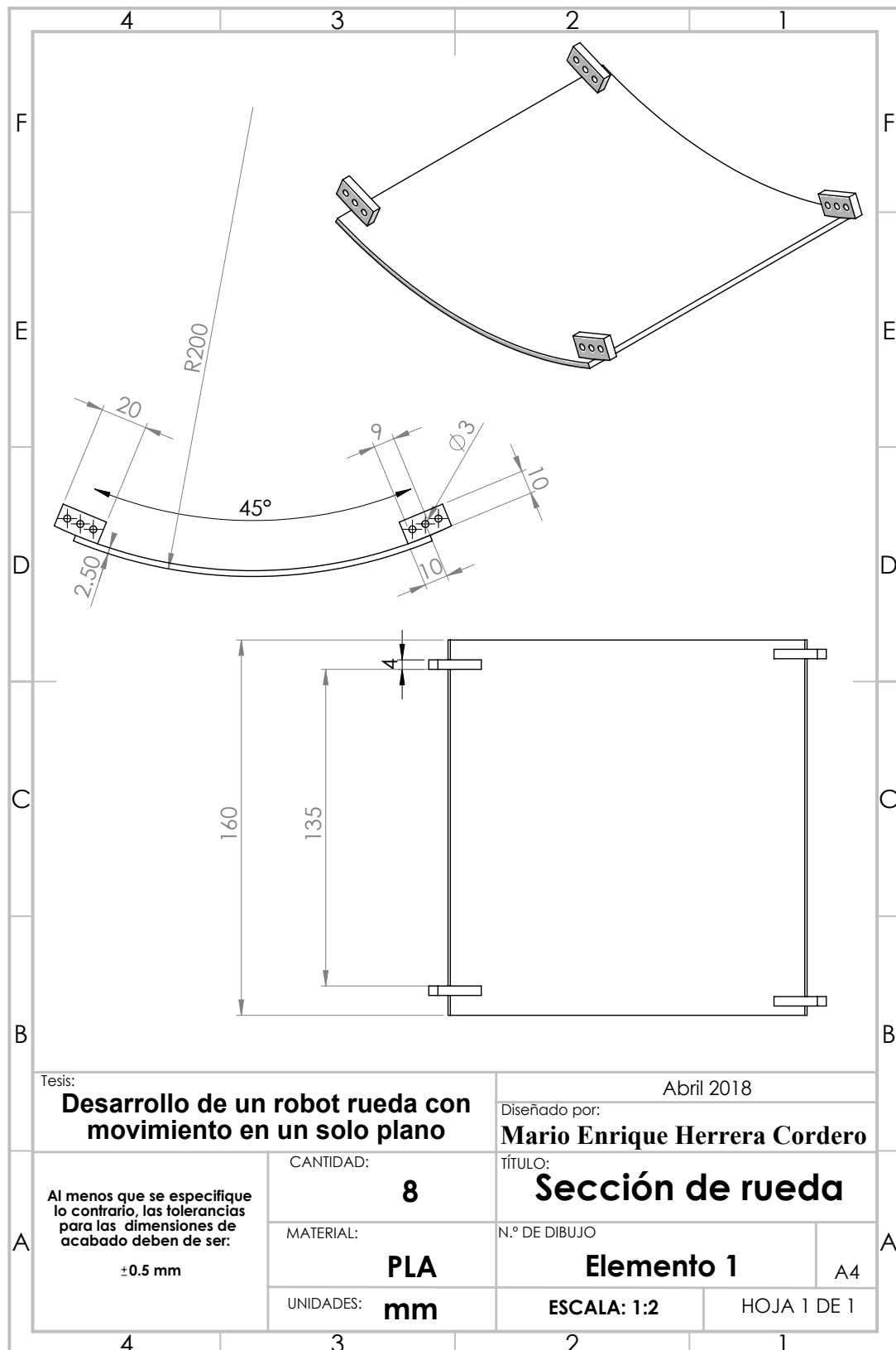


Figura G.10: Diagrama bloques completo.

Apéndice H

Dibujos técnicos de implementación del
robot rueda



Tesis:

Desarrollo de un robot rueda con movimiento en un solo plano

Abril 2018

Diseñado por:

Mario Enrique Herrera Cordero

TÍTULO:

Sección de rueda

Al menos que se especifique lo contrario, las tolerancias para las dimensiones de acabado deben de ser:

±0.5 mm

CANTIDAD:

8

MATERIAL:

PLA

UNIDADES:

mm

N.º DE DIBUJO

Elemento 1

ESCALA: 1:2

HOJA 1 DE 1

A

A

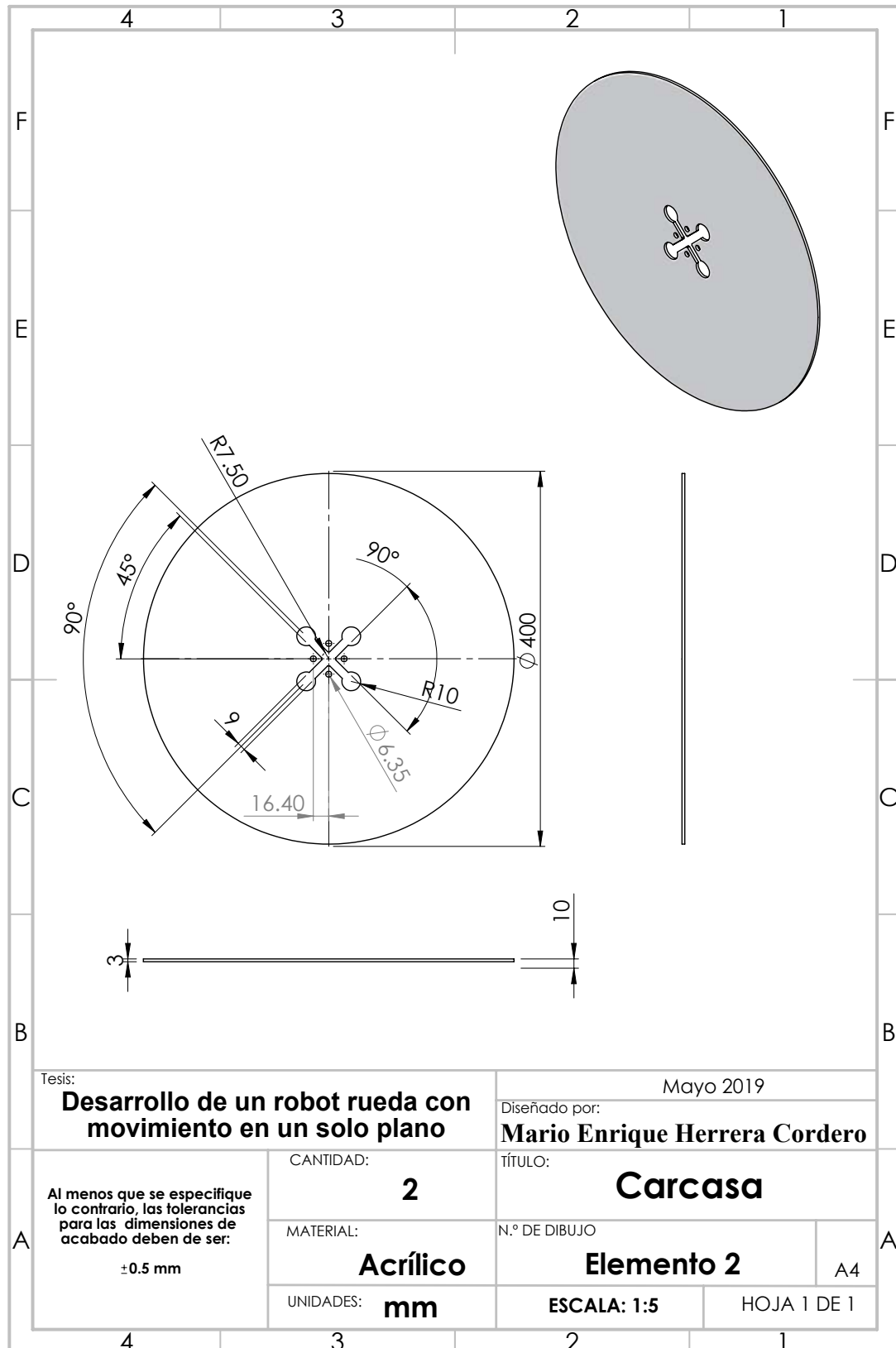
4

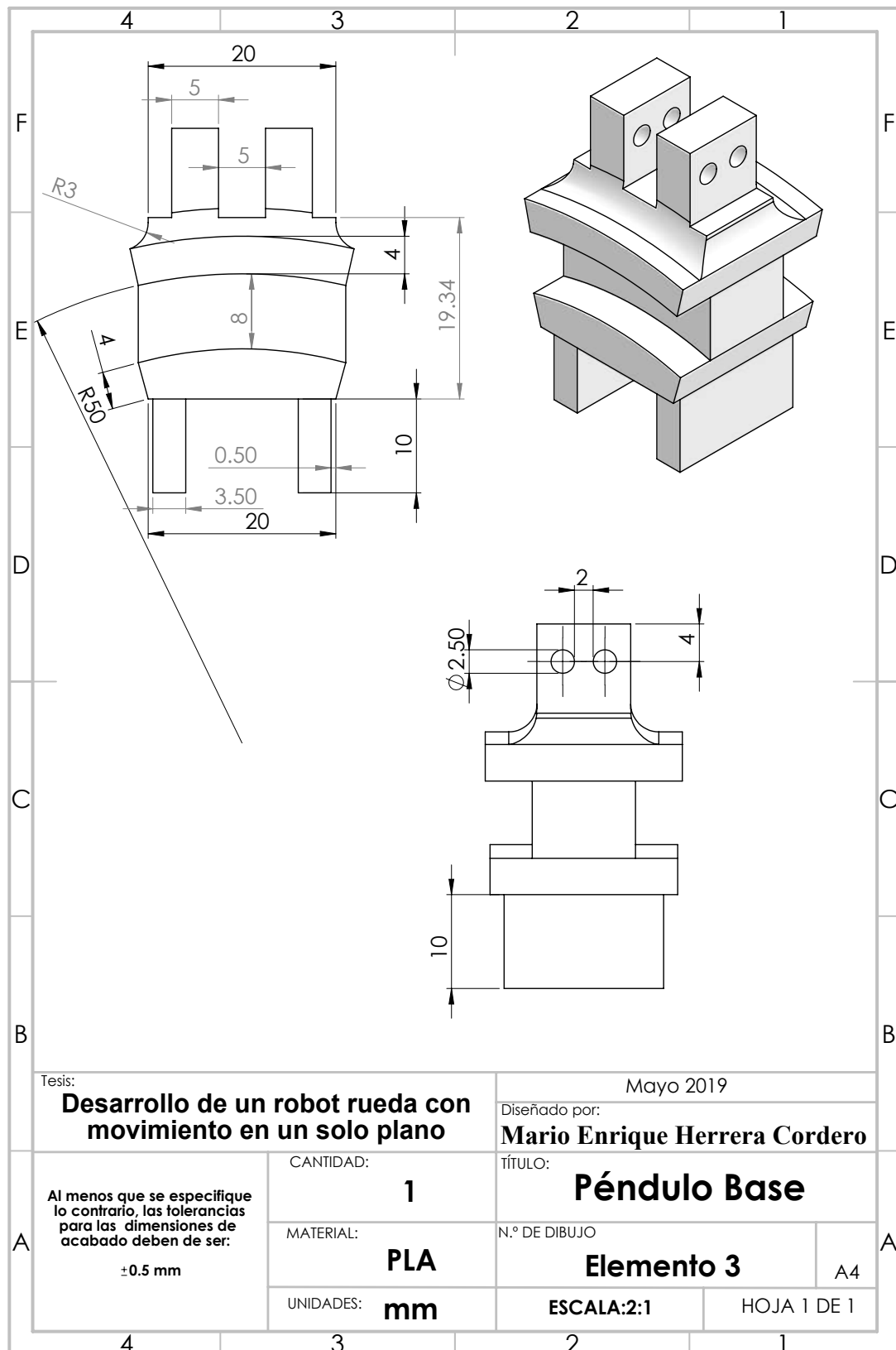
3

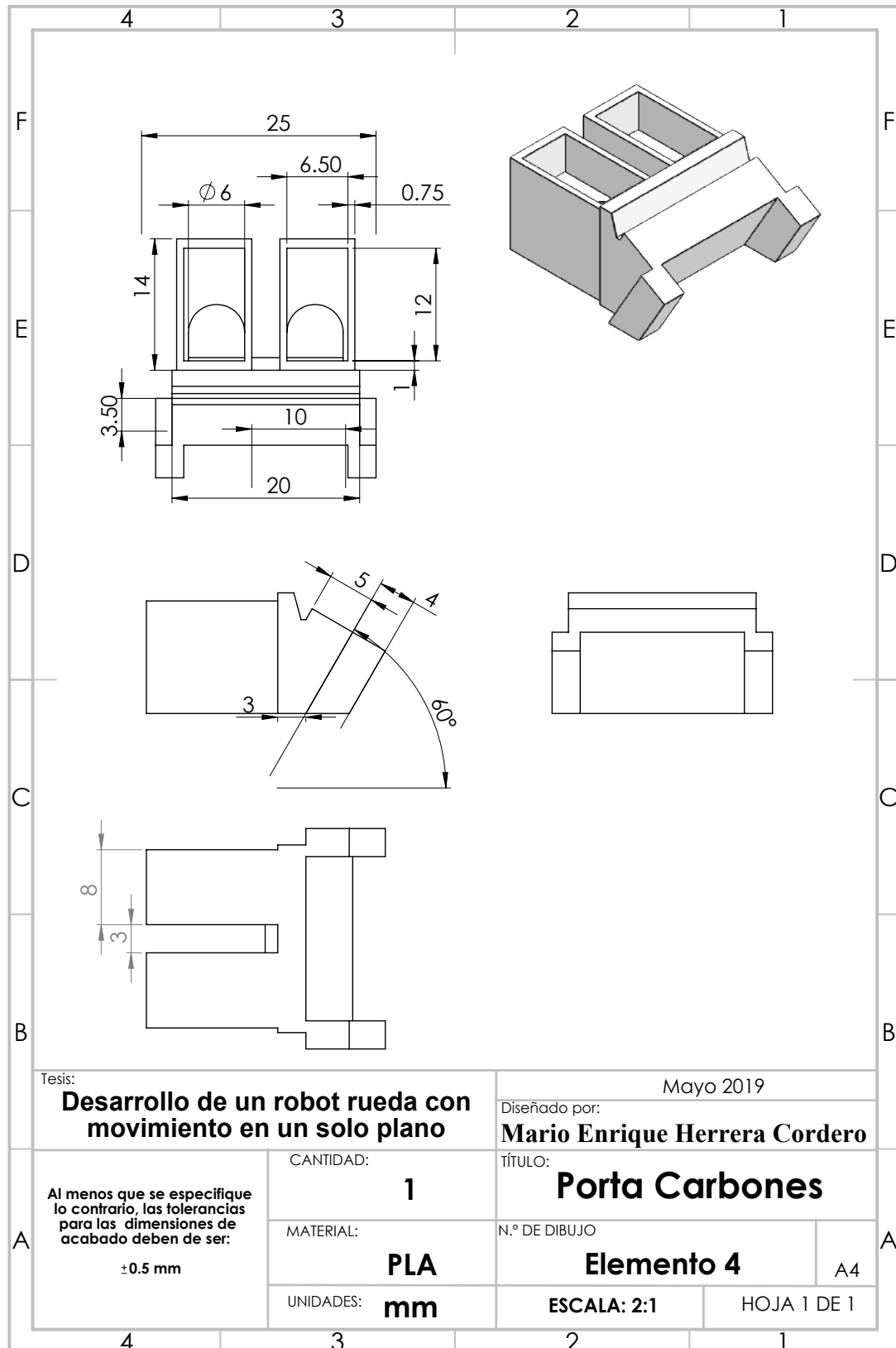
2

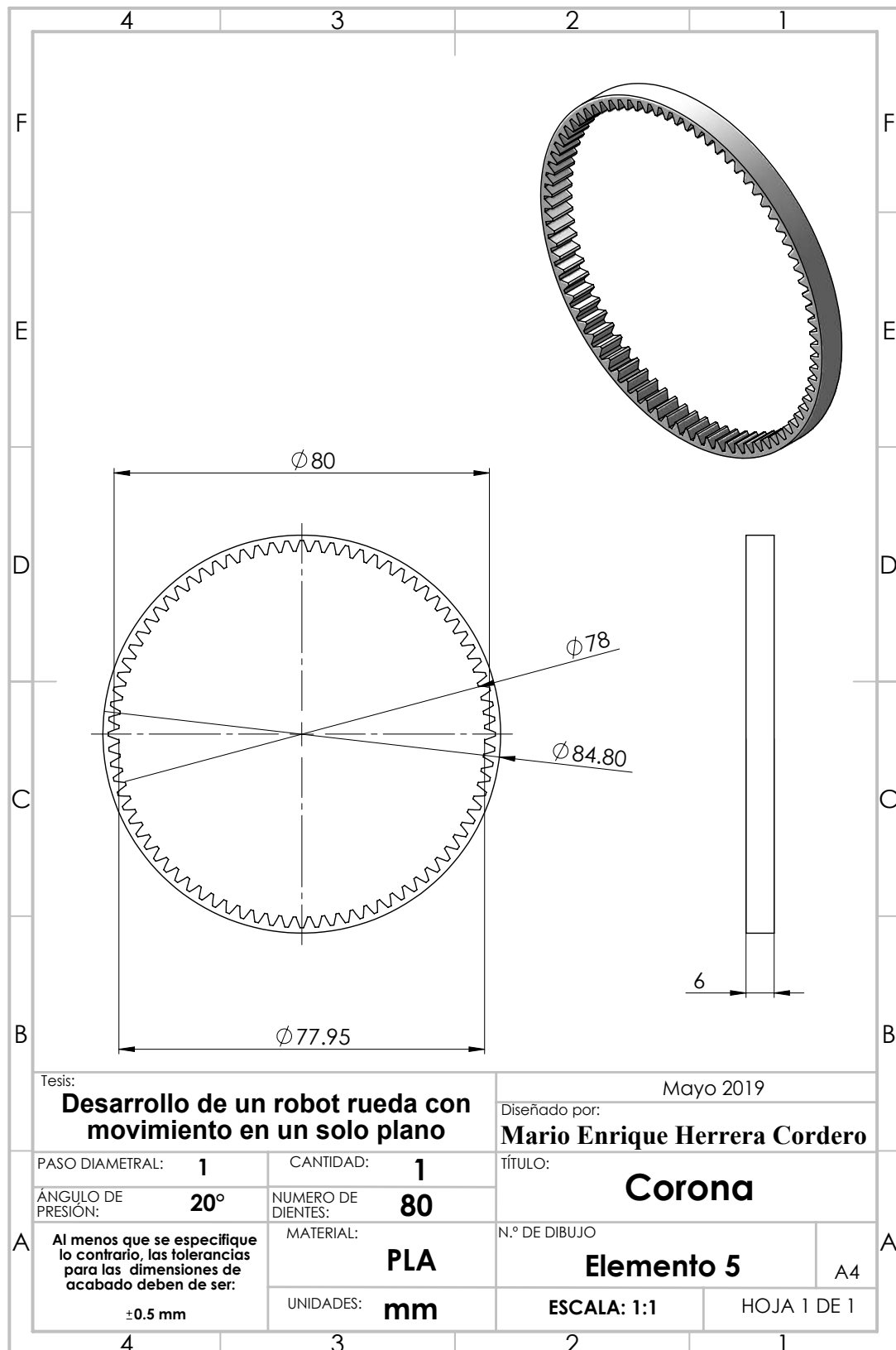
1

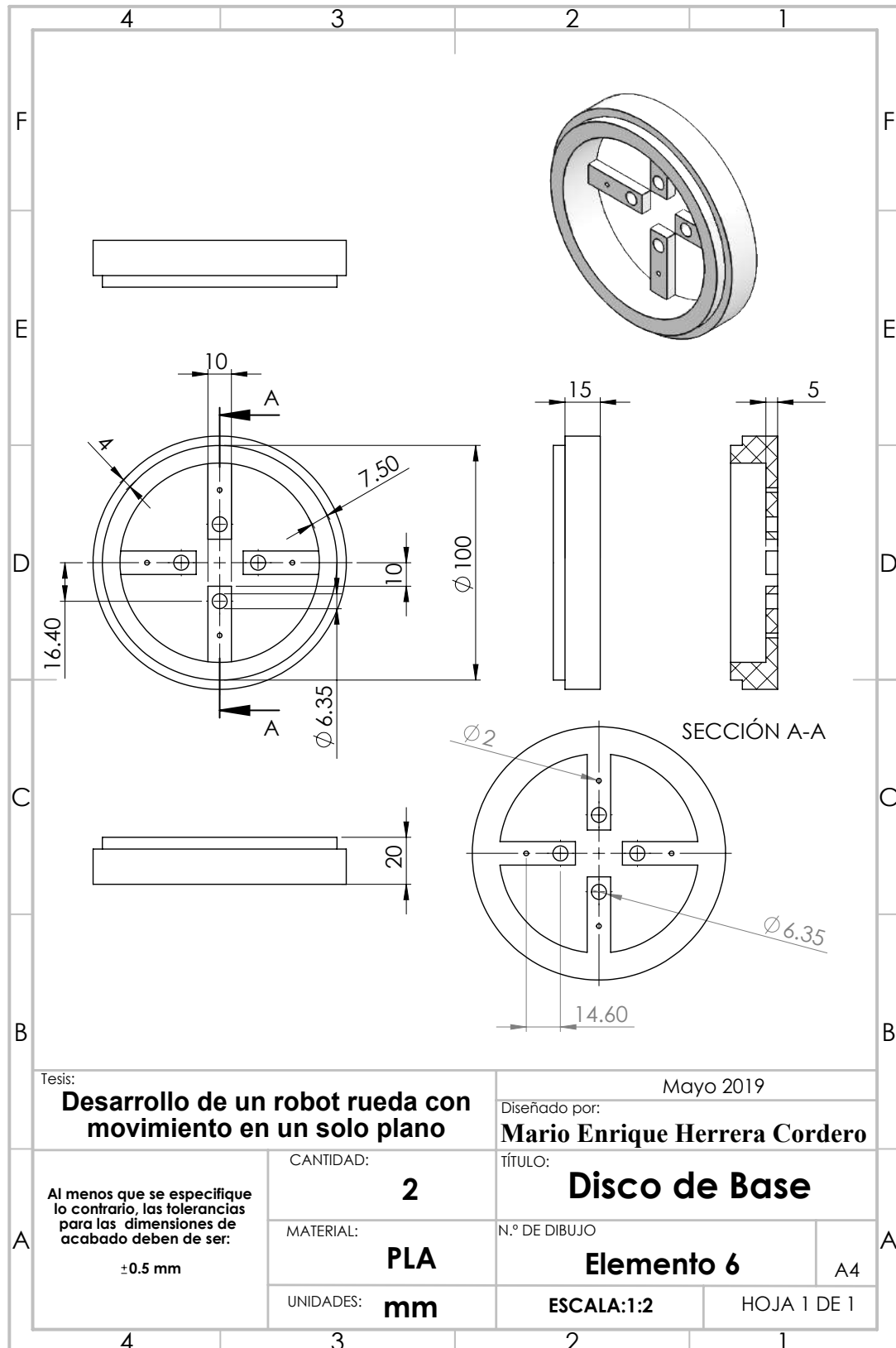
APÉNDICE H. DIBUJOS TÉCNICOS DE IMPLEMENTACIÓN DEL ROBOT RUEDA61

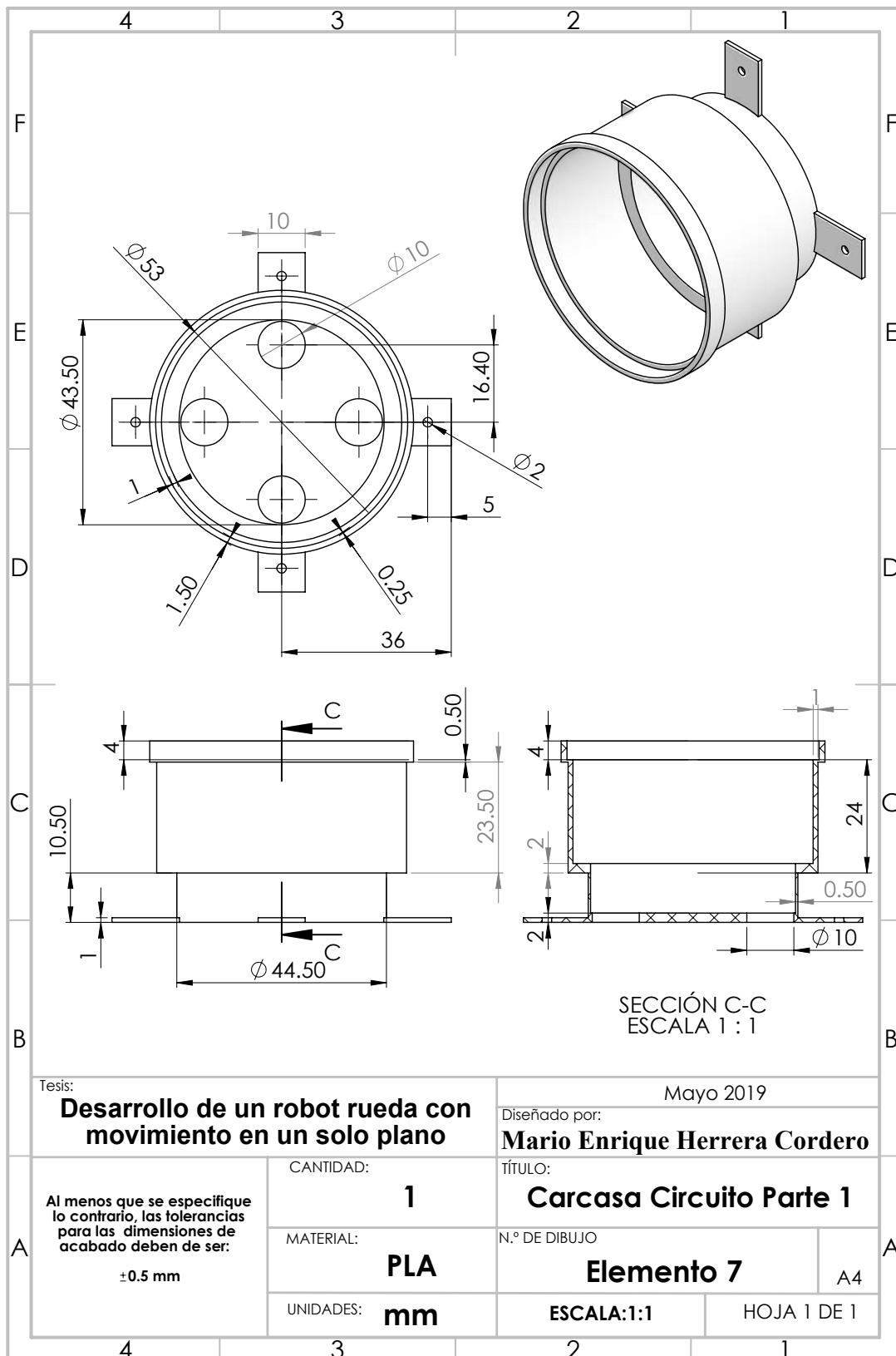


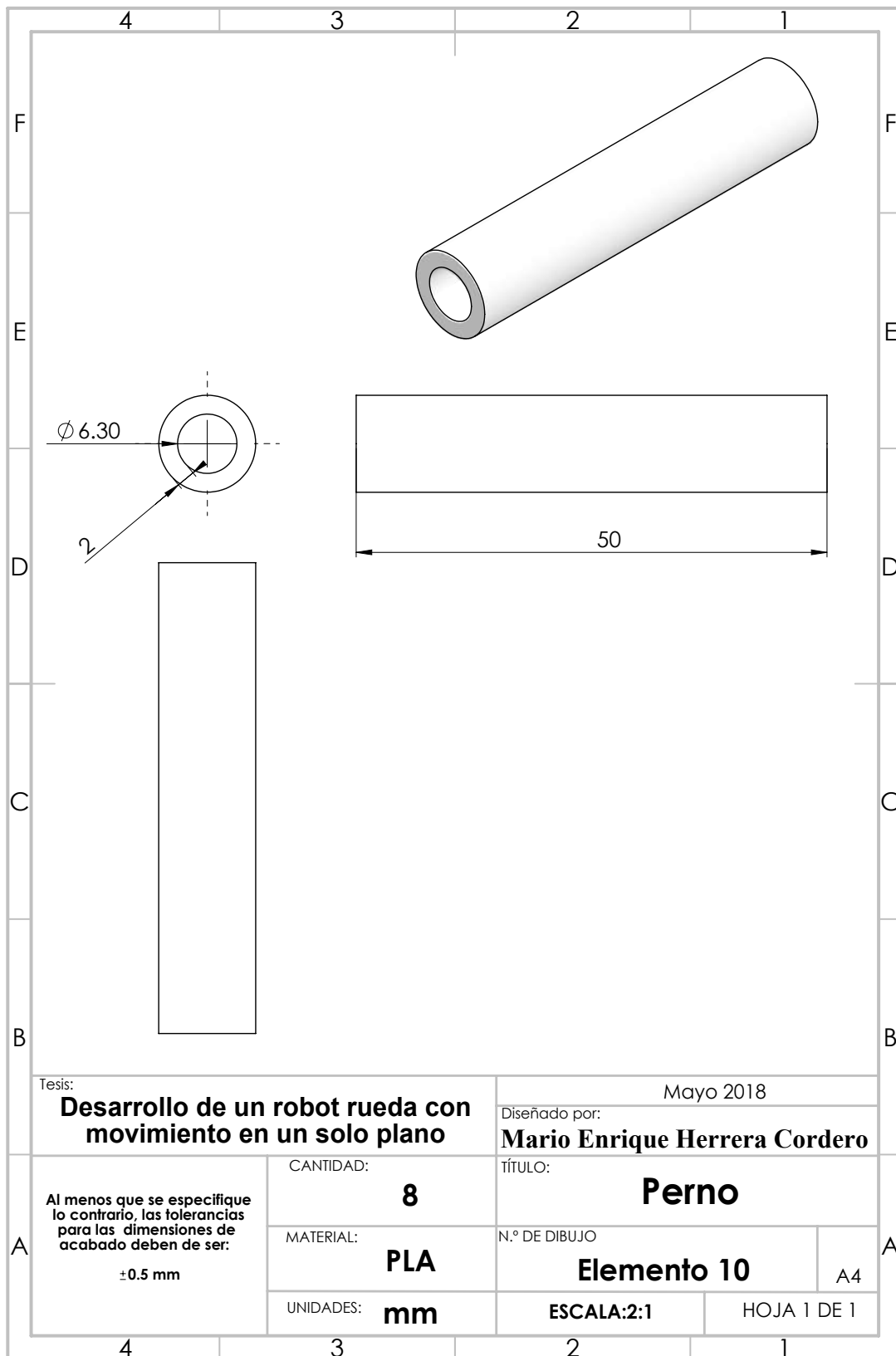












Tesis:

Desarrollo de un robot rueda con movimiento en un solo plano

Mayo 2018

Diseñado por:

Mario Enrique Herrera Cordero

TÍTULO:

Perno

Al menos que se especifique lo contrario, las tolerancias para las dimensiones de acabado deben de ser:
±0.5 mm

CANTIDAD:

8

MATERIAL:

PLA

UNIDADES:

mm

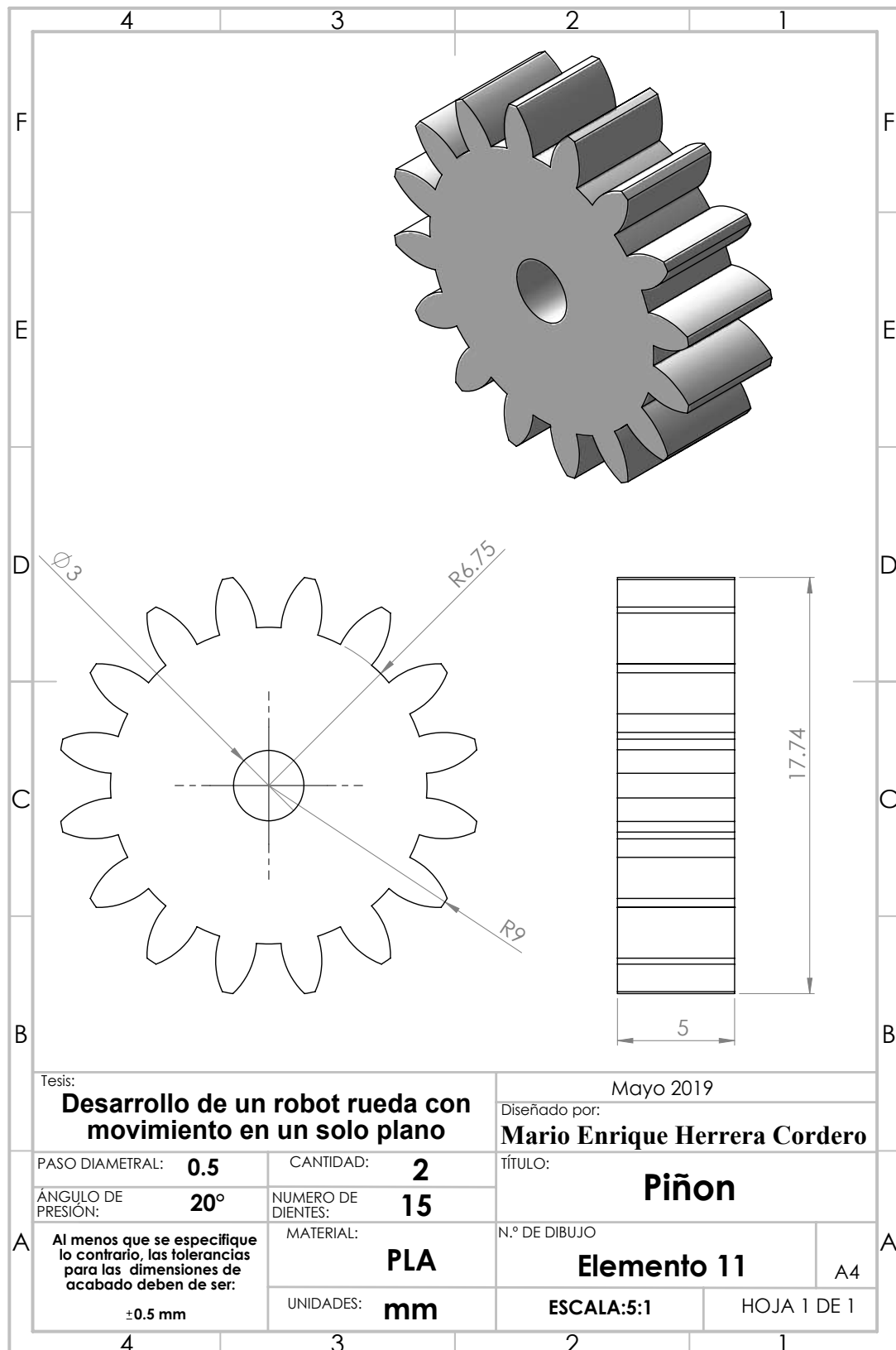
N.º DE DIBUJO

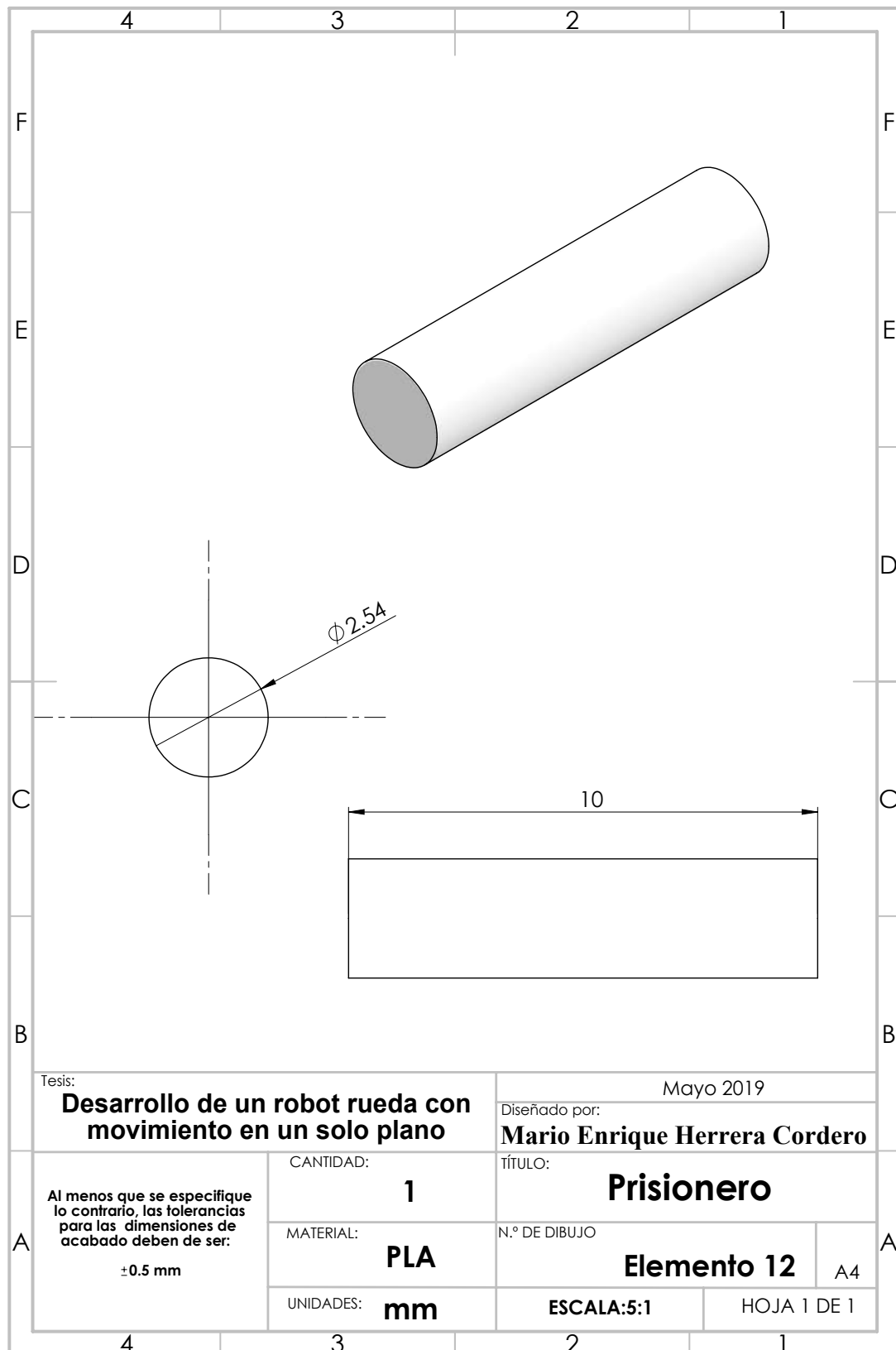
Elemento 10

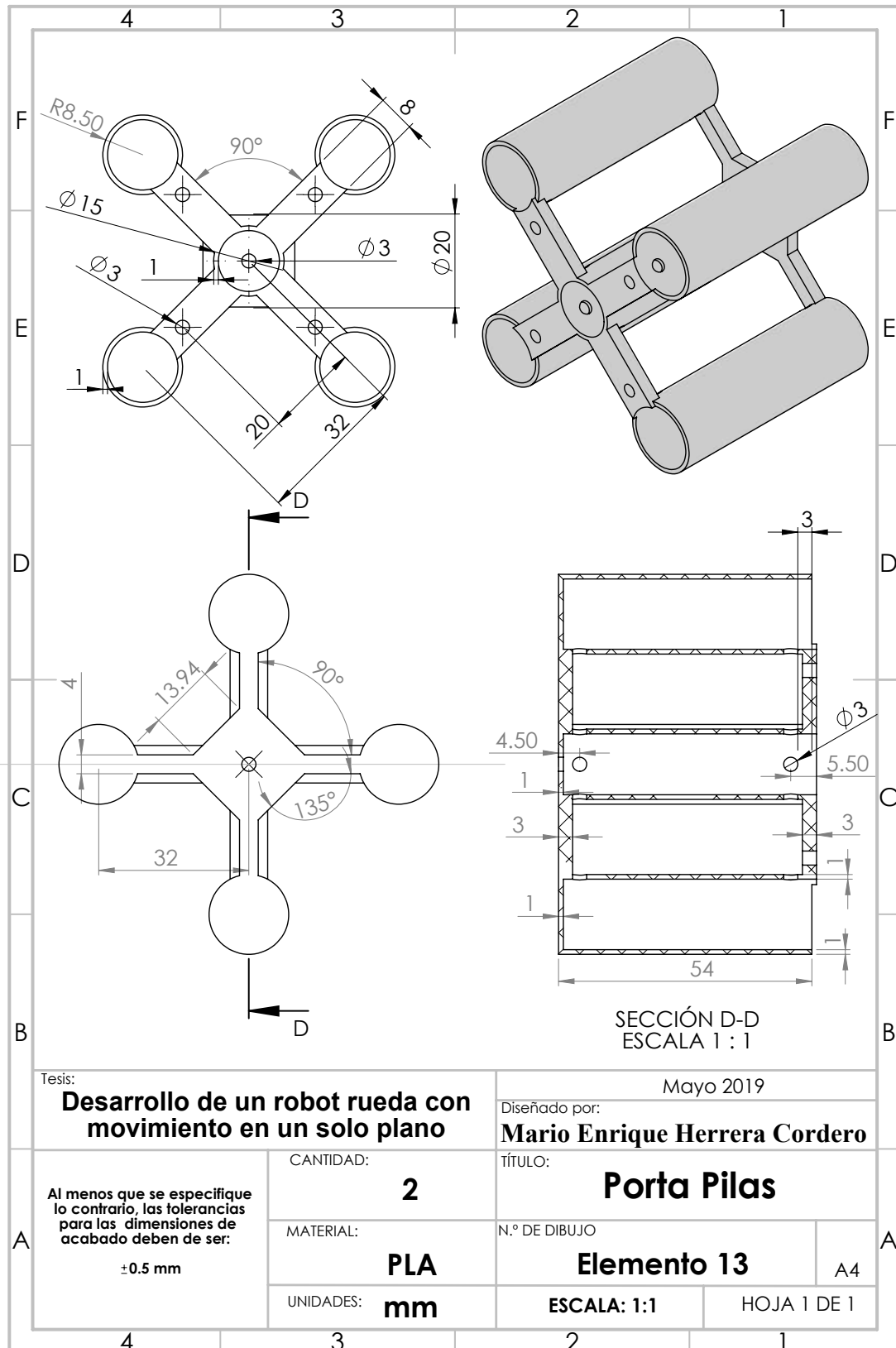
ESCALA:2:1

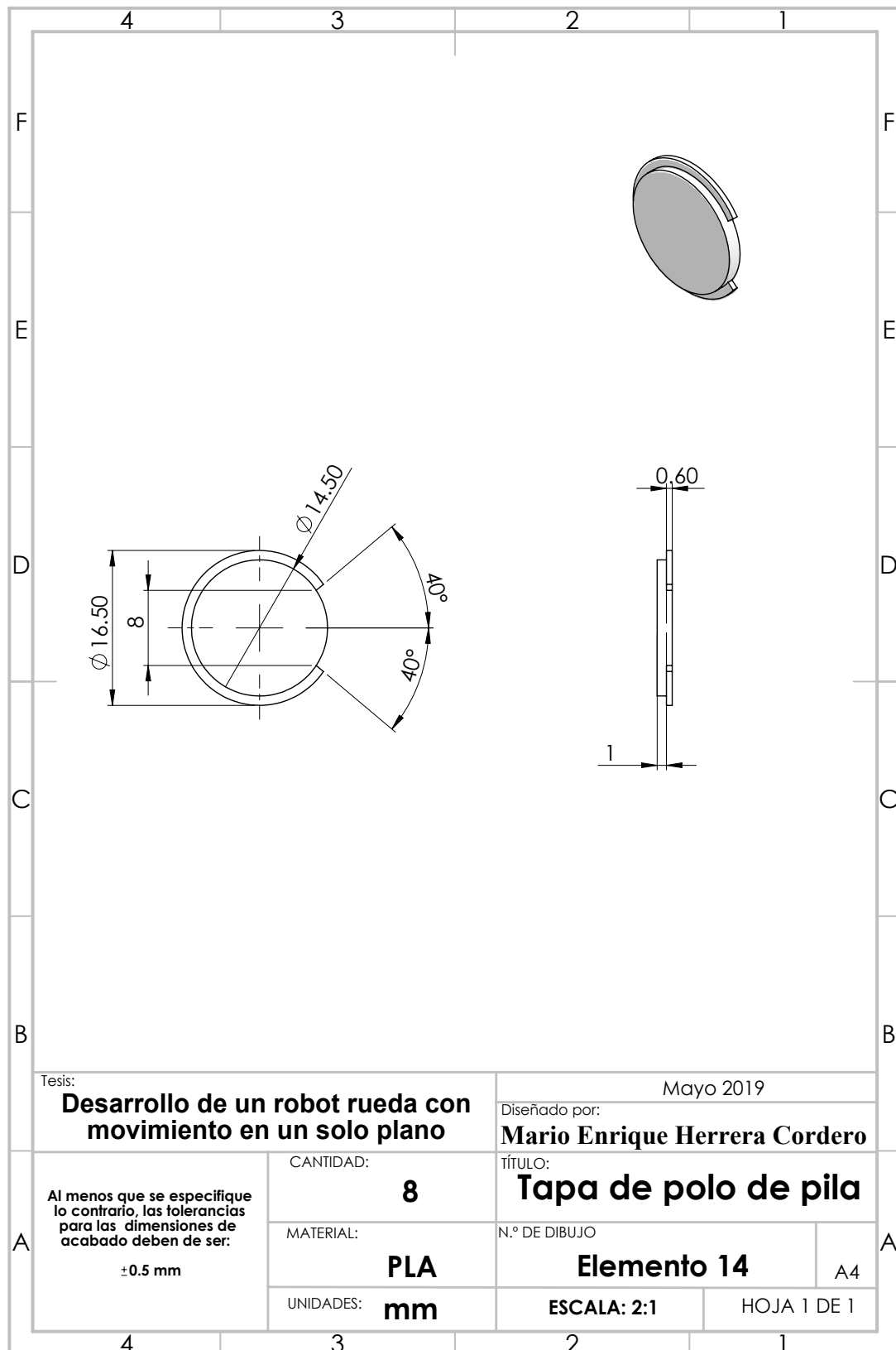
HOJA 1 DE 1

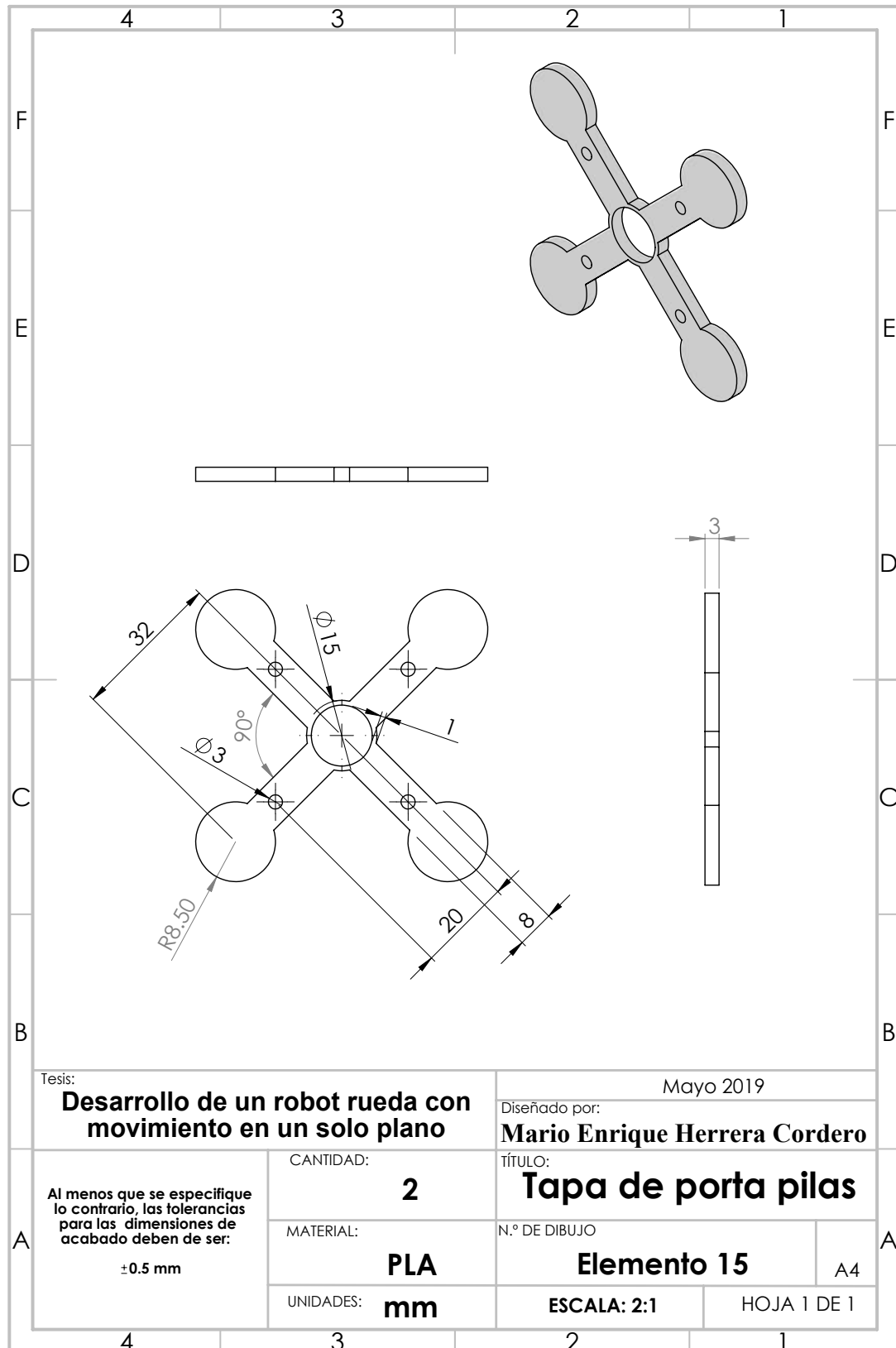
A4











Apéndice I

Presencia del trabajo de tesis en congresos, memorias y artículos

1. Mario E. Herrera Cordero, Manuel Arias Montiel, Esther Lugo González, "On the Dynamics and Control of a Single-Wheel Robot with Inertial Locomotion", *Latin American Symposium On Industrial And Robotic Systems*, Tecnológico de Monterrey, October 2019.
2. Mario E. Herrera Cordero, Manuel Arias Montiel, Esther Lugo González, "Design and Dynamic Modeling of a Novel Single-Wheel Pendulum Robot", *Mechanism Design for Robotics, Proceedings of the 4th IFToMM Symposium on Mechanism Design for Robotics*, pp. 353 – 360, Springer, 2018.
3. Mario E. Herrera Cordero, Manuel Arias Montiel, Esther Lugo Gonzalez, "Control de velocidad de un Robot Rueda con Locomoción Inercia por Péndulo", *17º Congreso Nacional de mecatrónica*, Instituto Tecnológico de Tepic, Octubre 2018.
4. Mario E. Herrera Cordero, Manuel Arias Montiel, Esther Lugo Gonzalez, "Modelo dinámico de un robot rueda con locomoción inercial", *3rd International Conference on Mathematical Modelling*, Universidad Tecnológica de la Mixteca, Octubre 2018.

Apéndice J

Glosario

- 1.– **Control** Es el poder manipular un sistema para que este llegue a tener cierto comportamiento en su respuesta, que es la deseada.
- 2.– **Giroscopio** Dispositivo que tiene la utilidad de sensar el ángulo de inclinación.
- 3.– **Linealizar** Proceso de transformación de una ecuación no lineal a una lineal en cierto rango numérico.
- 4.– **Puntos Críticos** Son los puntos difíciles de alcanzar por el sistema, estos pueden ser ciertos valores de temperatura, ciertos valores de desplazamiento, etc.
- 5.– **Microcontrolador** es un Circuito Integrado con tecnología VLSI que contiene una Unidad Central de Procesamiento (CPU), memoria para código, memoria para datos, además de otros recursos necesarios para el desarrollo de aplicaciones, por lo general con un propósito específico. Un microcontrolador es de hecho una computadora completa situada en un único chip.
- 6.– **Grado de Libertad** se refiere al número mínimo de parámetros que necesitamos especificar para determinar completamente la velocidad de un mecanismo o el número de reacciones de una estructura.

Bibliografía

- [1] J. Angeles, *Fundamentals of robotic mechanical systems: theory, methods, and algorithms*, vol. 124. Springer Science & Business Media, 2° ed ed., 2013.
- [2] A. K. Solutions, *Robotics: Appin knowledge solutions*. 1 er. ed., 2007.
- [3] K. Nozaki and T. Murakami, “A motion control of two-wheels driven mobile manipulator for human-robot cooperative transportation,” in *Proceedings of Industrial Electronics, 2009. IECON’09. 35th Annual Conference of IEEE*, pp. 1574–1579, IEEE, 2009.
- [4] A. Duarte, “Control de un robot autónomo tipo péndulo invertido,” masther thesis, Instituto Tecnológico de la Paz, Aug. 2016.
- [5] R. Chase and A. Pandya, “A review of active mechanical driving principles of spherical robots,” *Robotics*, vol. 1, no. 1, pp. 3–23, 2012.
- [6] M. R. Matthews, C. F. Gauld, and A. Stinner, *The pendulum: Scientific, historical, philosophical and educational perspectives*, vol. 13. Springer Science & Business Media, 2005.
- [7] R.Garcia, “Desarrollo de un robot tipo ballbot para aplicaciones de control.” Bachelor’s Thesis,Universidad Tecnológica de la Mixteca,pp. 1-30, 2014.
- [8] J. L. C. Miranda, “Application of kalman filtering and pid control for direct inverted pendelum control,” master thesis, California State University, Chico, Mar. 2009.
- [9] “<http://www.boletin.upiita.ipn.mx/index.php/ciencia/553-cyt-numero-41/840-sistemas-mecanicos-subactuados-pendulos-invertidos1>.”
- [10] “<http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/695/a4.pdf;sequence=4>.”
- [11] O. Alvarado, R. Campa, and J. Ollervides, “Modelado y control de un mecanismo tipo pendulo invertido esférico,” in *proceedings of Congreso Anual 2010 de la Asociación de México de Control Automático*, no. 1, pp. 1–6, 2010.
- [12] A. Valera, M. Vallés, and M. Cardo, “Desarrollo y control de un péndulo de furuta,” *Dpto. Ingeniería de Sistemas y Automática. Universidad Politécnica de Valencia. Camino de Vera*, vol. 14, p. 46022, 2002.

-
- [13] I. Fantoni, R. Lozano, and M. W. Spong, "Energy based control of the pendubot," *IEEE Transactions on Automatic Control*, vol. 45, no. 4, pp. 725–729, 2000.
- [14] S. C. Brown and K. M. Passino, "Intelligent control for an acrobot," *Journal of Intelligent & Robotic Systems*, vol. 18, no. 3, pp. 209–248, 1997.
- [15] P. V. M. Maalini, G. Prabhakar, and S. Selvaperumal, "Modelling and control of ball and beam system using PID controller," in *Proceedings of International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pp. 322–326, May 2016.
- [16] J. Hauser, S. Sastry, and G. Meyer, "Nonlinear control design for slightly non-minimum phase systems: Application to v/stol aircraft," *Automatica*, vol. 28, no. 4, pp. 665–679, 1992.
- [17] R. García and M. Arias, "Linear controllers for the nxt ballbot with parameter variations using linear matrix inequalities," *IEEE Control Systems Magazine*, vol. 33, pp. 121–136, June 2016.
- [18] M. Jankovic, D. Fontaine, and P. V. Kokotovic, "Tora example: cascade-and passivity-based control designs," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 3, pp. 292–297, 1996.
- [19] Y. Zhu, Y. Gao, C. Xu, J. Zhao, H. Jin, and J. Lee, "Adaptive control of a gyroscopically stabilized pendulum and its application to a single-wheel pendulum robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2095–2106, 2015.
- [20] J. L. Meriam and L. G. Kraige, *Engineering mechanics: dynamics*, vol. 2, pp. 736. John Wiley & Sons, 5^o ed ed., 2012.
- [21] "<http://en.wikipedia.org/wiki/gyroscope..>".
- [22] H. Fourati, N. Manamanni, L. Afilal, and Y. Handrich, "Complementary observer for body segments motion capturing by inertial and magnetic sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 1, pp. 149–157, 2014.
- [23] H. Jin, J. Zhao, J. Fan, and J. Lee, "Gain-scheduling control of a 6-dof single-wheeled pendulum robot based on dit parameterization," in *Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3511–3516, IEEE, 2011.
- [24] B. Bapiraju, K. Srinivas, P. P. Kumar, and L. Behera, "On balancing control strategies for a reaction wheel pendulum," in *Proceedings of India Annual Conference, 2004. The IEEE INDICON 2004. First*, pp. 199–204, IEEE, 2004.
- [25] J. Lee, S. Han, and J. Lee, "Decoupled dynamic control for pitch and roll axes of the unicycle robot," *IEEE Transactions on Industrial Electronics*, vol. 60, pp. 3814–3822, July 2012.

-
- [26] H. Jin, T. Wang, F. Yu, Y. Zhu, J. Zhao, and J. Lee, "Unicycle robot stabilized by the effect of gyroscopic precession and its control realization based on centrifugal force compensation," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 6, pp. 2737–2745, 2016.
- [27] A. Schoonwinkel, *Desing and Test of a Computer stabilized Unicycle*. Ph. d. thesis, Stanford University, Dec. 1987. pp. 1-398.
- [28] D. W. Vos and A. H. Von Flotow, "Dynamics and nonlinear adaptive control of an autonomous unicycle: Theory and experiment," in *Proceedings of the 29th IEEE Conference on Decision and Control, 1990.*, pp. 182–187, IEEE, 1990.
- [29] H. Jin, J. Hwang, and J. Lee, "A balancing control strategy for a one-wheel pendulum robot based on dynamic model decomposition: Simulations and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 4, pp. 763–768, 2011.
- [30] Z. Sheng and K. Yamafuji, "Postural stability of a human riding a unicycle and its emulation by a robot," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 709–720, 1997.
- [31] Y. Naveh, P. Z. Bar-Yoseph, and Y. Halevi, "Nonlinear modeling and control of a unicycle," *Dynamics and Control*, vol. 9, no. 4, pp. 279–296, 1999.
- [32] J. Hwang, S. Kim, and J. Lee, "Vision-based path planning for the rotation control of unicycle robot," in *Proceedings of Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, pp. 1408–1412, IEEE, 2010.
- [33] J. Zhao, M. Xiong, and H. Jin, "Dynamics and a convenient control design approach for a unicycle robot," in *Proceedings of Information and Automation (ICIA), 2010 IEEE International Conference on*, pp. 706–711, IEEE, 2010.
- [34] D. Gong, Q. Pan, G. Zuo, and W. Deng, "Lqr control for a self-balancing unicycle robot," in *Proceedings of Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, pp. 1424–1429, IEEE, 2012.
- [35] S. D. Lee and S. Jung, "Experimental verification of stability region of balancing a single-wheel robot: An inverted stick model approach," in *Proceedings of IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pp. 004556–004561, 2015.
- [36] K. Pathak, J. Franch, and S. K. Agrawal, "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 505–513, 2005.
- [37] Y. Takahashi, N. Ishikawa, and T. Hagiwara, "Soft raising and lowering of front wheels for inverse pendulum control wheel chair robot," in *Proceedings Intelligent Robots and Systems, 2003. (IROS 2003). 2003 IEEE/RSJ International Conference on*, vol. 4, pp. 3618–3623, IEEE, 2003.

-
- [38] H. G. Nguyen, J. Morrell, K. D. Mullens, A. B. Burmeister, S. Miles, N. Farrington, K. M. Thomas, and D. W. Gage, "Segway robotic mobility platform," in *Proceedings of SPIE 5609, Mobile Robots XVII*, pp. 207–220, International Society for Optics and Photonics, 2004.
- [39] R. P. M. Chan, K. A. Stol, and C. R. Halkyard, "Review of modelling and control of two-wheeled robots," *Annual Reviews in Control*, vol. 37, no. 1, pp. 89–103, 2013.
- [40] T. Takaki, T. Aoyama, and I. Ishii, "Development of inverted pendulum robot capable of climbing stairs using planetary wheel mechanism," in *Proceedings of Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 5618–5624, IEEE, 2013.
- [41] W.-J. W. C.-H. Huang and C.-H. Chiu, "Design and implementation of fuzzy control on a two-wheel inverted pendulum," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 2988–3001, 2011.
- [42] S. B. Cardini, "A history of the monocycle stability and control from inside the wheel," *IEEE Control Systems Magazine*, vol. 26, pp. 22–26, Sept. 2006.
- [43] E. Mumm, K. Davis, M. Mahin, D. Neal, and R. Hayes, "Miniature control moment gyroscope development," in *Proceedings of 2014 IEEE Aerospace Conference*, (Big Sky, MT, USA), pp. 1–9, Mar. 2014.
- [44] M. W. Spong, P. Corke, and R. Lozano, "Nonlinear control of the inertia wheel pendulum," *Automatica*, vol. 37, pp. 1845–1851, 1999.
- [45] Y. Xu, H. B. Brown, and K. W. Au, "Dynamic mobility with single-wheel configuration," *The International Journal of Robotics Research*, vol. 18, pp. 728–738, July 1999.
- [46] Y. Ou and Y. Xu, "Stabilization and line tracking of the gyroscopically stabilized robot," in *Proceedings of 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2, pp. 1753–1758.
- [47] J. H. Park and S. Jung, "Development and control of a single-wheel robot: Practical mechatronics approach," *Mechatronics*, vol. 23, pp. 594–606, May 2013.
- [48] A. Halme, T. Schonberg, and Y. Wang, "Motion control of a spherical mobile robot," in *Proceedings of 4th International Workshop on Advanced Motion Control, 1996. AMC '96-MIE.*, vol. 1, pp. 259–264.
- [49] A. Halme, J. Suomela, T. Schonberg, and Y. Wang, "A spherical mobile micro-robot for scientific applications," in *Proceedings of 4th ESA Workshop on Advanced Space Technologies for Robot Applications*, vol. 1, pp. 1–3, 1996.
- [50] A. Crossley, "A literature review on the design of spherical rolling robots," *Carnegie Mellon University*, pp. 1–15, Mar. 2006.

-
- [51] A. Bicchi, A. Balluchi, D. Prattichizzo, and A. Gorelli, “Introducing the sphericle: an experimental testbed for research and teaching in nonholonomy,” in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2620–2625, 1997.
- [52] “www.rotundus.se.”
- [53] R. Mukherjee, M. A. Minor, and J. T. Pukrushpan, “Simple motion planning strategies for spherobot: a spherical mobile robot,” in *Proceedings of the 38th IEEE Conference on Decision and Control*, vol. 3, pp. 2132–2137, 1999.
- [54] A. H. J. A and P. Mojabi, “Introducing august: a novel strategy for an omnidirectional spherical rolling robot,” in *Proceedings of 2002 IEEE International Conference on Robotics and Automation.*, vol. 4, pp. 3527–3533, 2002.
- [55] J. D. Hernández Vega, “ROSPHERE: Diseño, construcción y aplicación de una esfera robótica,” 2012.
- [56] H. U. of Technology, Ariadna, P. Jakubik, J. Suomela, M. Vainio, and T. Ylikorpi, *Biologically inspired solutions for robotic surface mobility*, vol. 1. Helsinki University of Technology, 2004.
- [57] B. P. DeJong, E. Karadogan, K. Yelamarthi, and J. Hasbany, “Design and analysis of a four-pendulum omnidirectional spherical robot,” *Journal of Intelligent & Robotic Systems*, vol. 86, no. 1, pp. 3–15, 2016.
- [58] “Hojas de datos l293d.”
- [59] “Hoja de datos mpu6050.”
- [60] “Hojas de datos mpu6050 registros.”
- [61] “Hoja de datos acs712-05a.”