UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"ROBOT MÓVIL AUTÓNOMO (AMR)"

TRABAJO FIN DE GRADO

Junio - 2024

AUTOR: Juan Miguel Oñate Tévar

DIRECTOR: Carlos Pérez Vidal

Contenido

1 INTRODUCCIÓN

Desde principios de esta década se esta producción una revolución en la industria con el objetivo de interconectar y mejorar procesos, para mejorar la cadena de valor. En Alemania surgió el concepto de Industria 4.0 en 2011, este concepto hace referencia al cambio que se está produciendo en la industria al implementar sensores, máquinas, componentes y sistemas informáticos los cuales están conectados entre sí a lo largo de la cadena de valor, estos sistemas están conectados entre sí con el objetivo de analizar procesos, para mejorar tiempo en ellos, reducir errores en el proceso y predecir eventos.

La automatización de estos procesos para crear una industria inteligente que se pueda comunicar con otras máquinas y personas lleva a la integración de nuevas máquinas capaces de interactuar y recopilar información para mejorar procesos, capaces de ser modificadas en tiempo real y de una forma descentralizada. Estas máquinas pueden ser autónomas o colaborativas con las personas, las cuales pueden ayudar a reducir la carga de trabajo de las personas. Aquí es donde entran los robots colaborativos, robots industriales, vehículos de guiado automático (AGV) y robots móviles autónomos (AMR) como solución de algunos de estos problemas.

En este trabajo se desarrollará un robot móvil colaborativa con idea de que se pueda adaptar a cualquier industria su movimiento será sobre dos ruedas, su único contacto con el suelo, todo su cuerpo; parte electrónica, eléctrica y de chasis crecerá hacia arriba con idea de que pueda moverse en espacios reducidos, pudiendo girar sobre sí mismo.

1.1 Objetivos del trabajo

El objetivo general de este trabajo se focaliza en el desarrollo y producción de un robot, tipo robot móvil autónomo, con el hándicap de que este robot se mueve sobre dos ruedas, con las consiguientes dificultades técnicas que ello conlleva. Los pasos seguidos para este trabajo han sido:

- Investigaciones de material y equipos necesarios para el proyecto, algunos de estos son; sensores, actuadores, chasis, fuentes alimentación, etc.
- Calibración se sensores y actuadores, con su respectivo escalado para una correcta actuación y lectura de estos.
- Diseño del chasis.
- Simulaciones, para poder optimizar el chasis y poder tener un punto de partida en el control de nuestro AMR.
- Construcción física y programación de nuestro AMR.
- Comprobación del funcionamiento de nuestro AMR, con las correspondientes modificaciones en el diseño del chasis y del programa.

2 ESTADO DEL ARTE

2.1 Industria 4.0 y tecnologías asociadas

La Industria 4.0 se basa en la interconexión de los sistemas de producción, la integración de la tecnología de la información y la automatización de los procesos. Con esto se pretende lograr una producción más eficiente y flexible, una mejor calidad de los productos, una reducción de los costos y una mayor personalización de estos, en la Figura 1 se puede ver algunas de las partes de la industria 4.0.

Figura 1. Partes de la industria 4.0

La industria 4.0 engloba un gran número de tecnologías que se están implementando en los últimos años, a nivel industrial, estas nuevas tecnológicas son:

- Internet de las cosas (IoT, por sus siglas en inglés): La conexión de máquinas, sensores y dispositivos a través de Internet permite la recopilación de datos en tiempo real, algunas de las aplicaciones de la IoT en la industria son:
    - Sensores: capaces de medir diferentes variables, como temperatura, humedad, presión, vibración y otros parámetros, en las máquinas y equipos. Todos esos datos se pueden recopilar y enviar a través de la red a un servidor o plataforma para su posterior procesamiento y análisis.
    - Redes de comunicación: Para conectar los dispositivos y sensores a la red, se utilizan diferentes tecnologías de comunicación, como Wi-Fi, Bluetooth, Zigbee, LoRaWAN o Sigfox.
    - Plataformas de análisis: Estas plataformas utilizan algoritmos y técnicas de análisis de datos recopilados por los sensores y dispositivos, con el objetivo de informar y alertar en tiempo real.
- Fabricación aditiva: También conocida como impresión 3D, permite la creación de objetos tridimensionales a partir de archivos digitales, algunas de sus ventajas son la capacidad de producir piezas de formas complejas y personalizadas con una precisión muy alta, la reducción del tiempo de producción y el costo de materiales. Sus aplicaciones más importantes son:

    - Prototipado rápido: La posibilidad de crear prototipos sin molde y la posibilidad de hacer un prototipo en pocas horas o minutos permite a los ingenieros la realización de varios prototipos si fuera necesario, en poco tiempo.
    - Producir piezas personalizadas: Por su alto detalle se pueden hacer desde prótesis médicas a componentes de un avión.
    - Fabricación de herramienta o útiles: Herramientas o útiles los cuales tienen una función muy específica y para un uso en particular, fabricarlos con otros métodos sería muy costoso.
    - Producción de piezas de repuesto: Piezas que se necesitan con urgencia o piezas que ya están descatalogadas y no se pueden conseguir, una manera fácil de conseguir estas piezas podría ser la fabricación aditiva.
    - Fabricación de producto final: Cada vez más usado por su personalización, bajo costo de fabricación para pequeños lotes y rapidez en la fabricación.
- Realidad aumentada (RA): La RA se utiliza para superponer información digital en el entorno físico, lo que ayuda a los trabajadores en tareas de montaje, mantenimiento y reparación al proporcionar instrucciones visuales y guías interactivas.
- Big Data y análisis predictivo: La recopilación masiva de datos y su análisis en tiempo real permiten la optimización de los procesos de producción, la detección temprana de fallas o problemas, y la toma de decisiones basadas en datos para mejorar la eficiencia y calidad.
    - Mantenimientos predictivos: La recopilación de datos del funcionamiento de una maquina sirve para identificar patrones y anomalías lo que podría usarse para predecir cuándo va a fallar esa máquina.
    - Monitorización de rendimiento: La recopilación de datos de una maquina o proceso, permite a los gerentes identificar oportunidades de mejora y optimización.
    - Gestión de inventario: El Big Data se utiliza para recopilar y analizar información sobre los niveles de inventario, la demanda del mercado y otros factores que afectan la gestión de inventario. Lo cual se puede usar para la gestión de esta.
    - Control de calidad: Los sensores y dispositivos recopilan los datos los cuales se pueden usar para detectar defectos y errores de calidad, permitiendo corregir esos problemas antes de que llegue al mercado.
    - Personalización del producto: El Big Data puede recopilar información de los clientes, pudiendo las empresas modificar su producto, ofreciéndoles a los clientes un producto o servicio que satisfagan sus necesidades.
- Inteligencia Artificial (IA): Con idea de hacer un proceso específico para el cual ha sido entrenado. La IA puede ser utilizada en diferentes procesos de la industria, como en robótica industrial para automatizar tareas repetitivas y peligrosas. La IA puede mejorar la eficiencia y tiene capacidad para adaptarse a la producción. Sus usos principales son:
    - Aprendizaje automático (Machine learning): Mediante algoritmos la IA es capaz de analizar gran cantidad de datos para encontrar patrones. En la industria, el aprendizaje automático se utiliza para optimizar la producción, mejorar la calidad y reducir los costos.
    - Redes neuronales artificiales: Un tipo de aprendizaje automático que usa la IA inspirado en el cerebro humano, usa nodos los cuales se interconectan con otros nodos estos están agrupados en diferentes capas, con una jerarquía.
    - Procesamiento de lenguaje natural (Natural language processing - NLP): Se utiliza para comprender y analizar el lenguaje humano. En la industria, se utiliza

para la gestión de clientes, la gestión de redes sociales y la optimización de la cadena de suministro.
    - Robótica inteligente: Los robots o líneas automatizadas capaces de tomar decisiones en función de un aprendizaje anterior son conocidos como robots inteligentes y se usa para optimizar procesos y mejorar su calidad.
    - Análisis predictivo: Capaz de predecir el comportamiento futuro en función de patrones históricas se puede usar para predecir la demanda del mercado, el mantenimiento preventivo y la gestión de inventarios.
- Automatización y robótica: La automatización y la robótica son muy utilizados en la industria ya que son precisos, rápidos y constantes. Algunos de sus usos pueden ser:
    - Fabricación en línea: son muy usados en procesos repetitivos, en los cuales es fundamental la rapidez de fabricación y la precisión en la misma.
    - Control de calidad: gracias a diferentes sensores que pueden incluir nuestro robot o automatismo somos capaces detectar errores en la fabricación, como podría ser, un mal par de apriete en un tornillo, reconocimiento de algún desperfecto mediante la visión artificial o por ejemplo detectar un mal curado de la resina en un horno mediante sondas de temperatura.
    - Logística y cadena de suministros: Los robots se pueden usar para la gestión del almacén, la organización, empaquetado, transporte y entrega de productos y todo de forma autónoma reduciendo mano de obrar, errores y siendo más rápidos y efectivos.
    - Seguridad en el lugar de trabajo: los robots pueden ser robustos y aguantar condiciones ambientales adversas lo cual es ideal para trabajar en lugares en los que una persona podría sufrir alguna lesión, ya sea por fática al ser un trabajo repetitivo o por estas en condiciones adversas, altas temperaturas, lugares con productos tóxicos o inflamables.

2.2 El rol de la robótica en la industria 4.0

Aunque el término “robot” fue acuñado por Karel Čapek en su obra de teatro de ciencia ficción R.U.R. (Robots Universales Rossum) en 1921, no fue hasta la década de los 50 cuando se desarrolló el primer robot industrial, conocido como Unimate. Este primer robot, que realizaba tareas de manipulación y soldadura, era voluminoso y pesaba más de 1000 kg.

Desde entonces, la robótica ha evolucionado a pasos agigantados. Los robots de hoy en día son más compactos, ligeros y versátiles. Además, gracias a los avances tecnológicos, las nuevas generaciones de robots están equipadas con sensores avanzados, sistemas de visión y aprendizaje automático. Esto les permite adaptarse a entornos cambiantes, colaborar con los humanos y tomar decisiones en tiempo real.

La robótica juega un papel crucial en la Industria 4.0 al permitir la automatización inteligente de procesos, la colaboración entre humanos y máquinas, y la integración con otros sistemas como la inteligencia artificial. Esto conduce a entornos de fabricación más interconectados, seguros y productivos.

A lo largo de los años se ha tenido un salto de tecnología, año tras año, dentro del sector de la robótica industriales, creando diferentes generaciones asociadas a estas innovaciones tecnológicas:

- **Primera generación**: Son los robots manipuladores. Su sistema de control está basado en “paradas fijas”. Se utilizan principalmente para mover objetos, pero están muy limitados en número de movimientos. Surgieron alrededor de la década de los años 50 y 60. Estos robots eran principalmente utilizados en la industria automotriz.
- **Segunda generación**: Conocidos como robots de aprendizaje, surgieron alrededor de los años 70 y 80. Estos robots son capaces de memorizar y repetir secuencias de movimientos que han sido previamente realizadas por un operador humano. Estos robots tienen la capacidad de percibir parte de su entorno mediante sensores especializados, lo que les permite desplazarse en posiciones fijas. Surgieron alrededor de los años 70 y 80.
- **Tercera generación**: Son reprogramables y son capaces de adquirir alguna percepción de su entorno mediante sensores. Estos robots se controlan mediante procesadores y controladores. Se desarrollaron entre los años 80 y 90.
- **Cuarta generación**: Conocidos como robots inteligentes comenzaron a desarrollarse a finales de los años 90. Estos robots están equipados con sensores avanzados y sistema básico de auto aprendizaje, que les permiten interpretar su entorno en tiempo real.
- **Quinta generación**: Se caracteriza por incorporar la inteligencia artificial, se empezaron a desarrollar sobre la década de 2010. La incorporación de la inteligencia artificial dota a los robots de un aprendizaje automático, sin intervención humana, siendo capaces de imitar y desarrollar modelos de actuación, razonamiento y de conducta, en los entornos.

A la vez que se mejoraba la tecnología en estos robots se iban creando diferentes robots con diferentes cinemáticas y grados de libertad en función del trabajo que fueran a desempeñar, pudiendo ser este movimiento:

- **Cartesiano**: Los robots cartesianos (Figura 2) se mueven a lo largo de tres ejes cartesianos (X, Y, Z) y son utilizados a menudo en la automatización de procesos industriales, como el ensamblaje, la soldadura y la manipulación de materiales. Son muy versátiles y se utilizan comúnmente en industrias como la electrónica, la automotriz, la de alimentos y la farmacéutica.

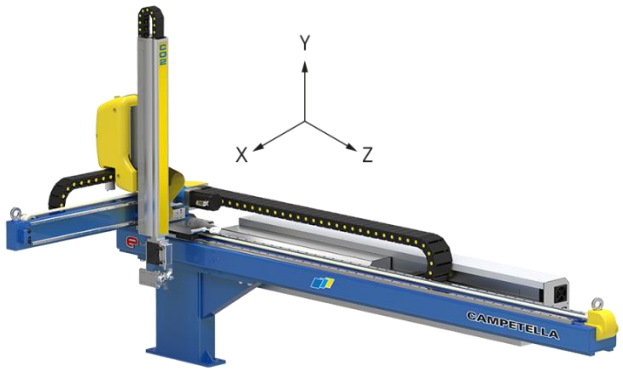


Figura 2. Robot cartesiano de 3 ejes C01-C02 Series EVO Campetella Robotic Center

- **Cilíndrico**: Los robots cilíndricos (Figura 3) tienen al menos una junta giratoria en la base y al menos una junta prismática para conectar los eslabones. Son ideales para operaciones de ensamblaje y manipulación con garras robóticas.

Figura 3. Robot cilíndrico RZY-RTLA-90

- **SCARA**: Los robots SCARA (Selective Compliance Assembly Robot Arm) (Figura 4) son principalmente cilíndricos en diseño, se caracterizan por disponer normamente de 4 grados de libertad y de realizar trabajos con posicionamiento horizontal. Se utilizan comúnmente en aplicaciones de montaje. Son extremadamente versátiles y destacan por trabajar a grandes velocidades para aplicaciones Pick & Place y de ensamblaje.

Figura 4. Robot Scara IXP-3N4515

- **Delta**: Los robots delta (Figura 5) están construidos a partir de paralelogramos unidos conectados a una base común. Se utilizan frecuentemente para aplicaciones de empaquetado y de manufactura, gracias a las altas velocidades que alcanzan.

Figura 5. Robot delta ABB IRB 360

- **Articulado**: Este diseño de robot (Figura 6) cuenta con juntas giratorias y puede variar desde dos estructuras simples de juntas a 10 o más articulaciones. Los robots articulados son muy útiles en el dominio de la industria. Permitiendo realizar operaciones de soldadura, montaje, mecanizado, y pintura entre otras.

Figura 6. Robot Polar Fanuc LR Mate 200iD

- **Móviles**: Estos robots (Figura 7) se caracterizan por no tener una base fija en el espacio, sobre la que pueda girar o desplazarse, lo que les permite moverse libremente en el espacio. Son usados para el transporte de materiales o mercancías en almacenes o líneas de montaje. Existen de dos tipos comunes de robots móviles los vehículos de guiado automático AGV y los robots móviles autónomos AMR.

Figura 7. Robot móvil MOBOT AGV FlatRunner MW (004)

2.3 Robots móviles

Cuando nos referimos a robots móvil estamos haciendo referencia a un tipo de robot que no tiene un punto de unión con una base fija, sino que se pueden moverse libremente por el espacio (Figura 7). Este tipo de robots también son conocimos con el sustantivo de vehículos en vez de robots ya que la misión principal para la que fueron creadas fue para transportar cosas de un punto a otro de un almacén o línea de montaje.

Dentro de los robots móviles podemos encontrar principalmente los robots móviles autónomos AMR y los vehículos de guiado automático AGV. Los robots AMR son una evolución de los robots AGV, y sus características y diferencias son:

- **Vehículos de guiado automático AGV**: funcionan con guías físicas preestablecidas, lo que les proporciona una alta precisión de navegación. Son ideales para aplicaciones que requieren movimientos repetitivos y precisos, como el transporte de productos en una línea de producción. Sin embargo, esta dependencia de las guías físicas limita su capacidad de adaptación y movilidad fuera de su zona predefinida.

- **Robots móviles autónomos AMR**: utilizan tecnologías de navegación autónoma, como sensores y cámaras, que les permiten planificar y reconfigurar su ruta en tiempo real. Esto les proporciona una gran flexibilidad y versatilidad, ya que pueden detectar y evitar obstáculos y personas, modificando la ruta de forma autónoma.

Aunque los AGV son precisos para trayectos repetitivos, los AMR son más versátiles y flexibles para cubrir las diferentes necesidades de la logística interna de una empresa.

3 NECESIDADES DEL PROYECTO

El objetivo del proyecto es la creación de un robot tipo AMR, este robot se crea con la idea de operar en espacios reducidos, para ello el robot contará de dos ruedas para desplazarse, el chasis del robot crecerá hacia arriba. Lo que hará que nuestro robot sea un AMR en vez de una AGV es la dotación de diferentes sensores para hacer a nuestro robot autónomo y asi tenga la capacidad de situarse en el espacio. Para ello el robot incorporará un lidar 360º, con ello se podrá posicionar los diferentes objetos que le rodean, también incorporara una cámara para que mediante la visión artificial este robot sea capaz de identificar objetos.

3.1 Dificultades técnicas

El simple hecho de mover el robot y lograr que mantenga la verticalidad ya supone un desafío considerable. Dado que nuestro robot se mantiene sobre dos ruedas, esto provoca una inestabilidad inherente. Ante cualquier perturbación, el robot puede alejarse de su estado de equilibrio (Figura 8). Definimos que nuestro robot está en estado de equilibrio cuando el chasis mantiene la verticalidad respecto al horizonte. Por lo tanto, la primera parte de este proyecto se centrará en lograr que nuestro robot sea estable en su estado de equilibrio. Una vez alcanzado este objetivo, el siguiente paso será permitir que el robot se mueva en la dirección y a la velocidad que indiquemos. Para lograr esto, nos apoyaremos en el uso de reguladores anidados.

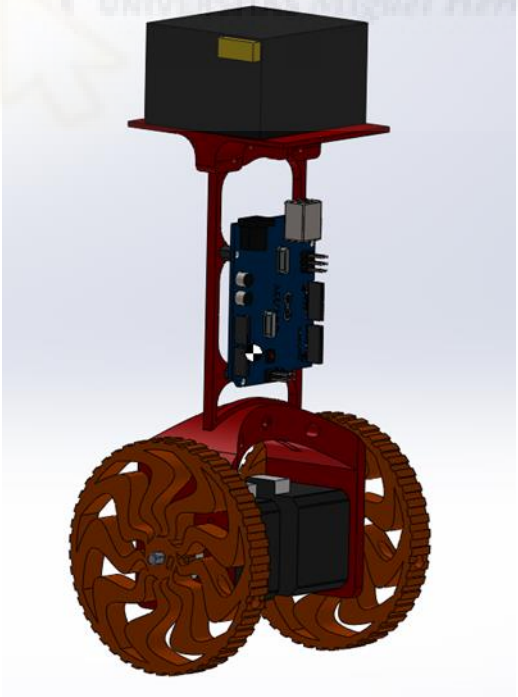


Figura 8. Diseño 2. Robot en su estado de equilibrio

La importancia de un buen controlador para nuestro robot es tan crucial como el diseño adecuado del chasis del robot. Al diseñar un controlador con el objetivo de controlar la posición, una variable muy importante a considerar es la masa y el centro de masas del robot. En mi caso,

esto es aún más relevante ya que estoy intentando controlar un robot que se mantiene sobre dos ruedas en un punto de equilibrio. La herramienta SolidWorks me ayudó a calcular el centro de masas de los diferentes diseños. Además, gracias a las simulaciones en Matlab, pude observar cómo se comportan estos diseños ante diferentes controladores. En la Figura 9, se pueden apreciar los dos primeros diseños y sus respectivos centros de masas.

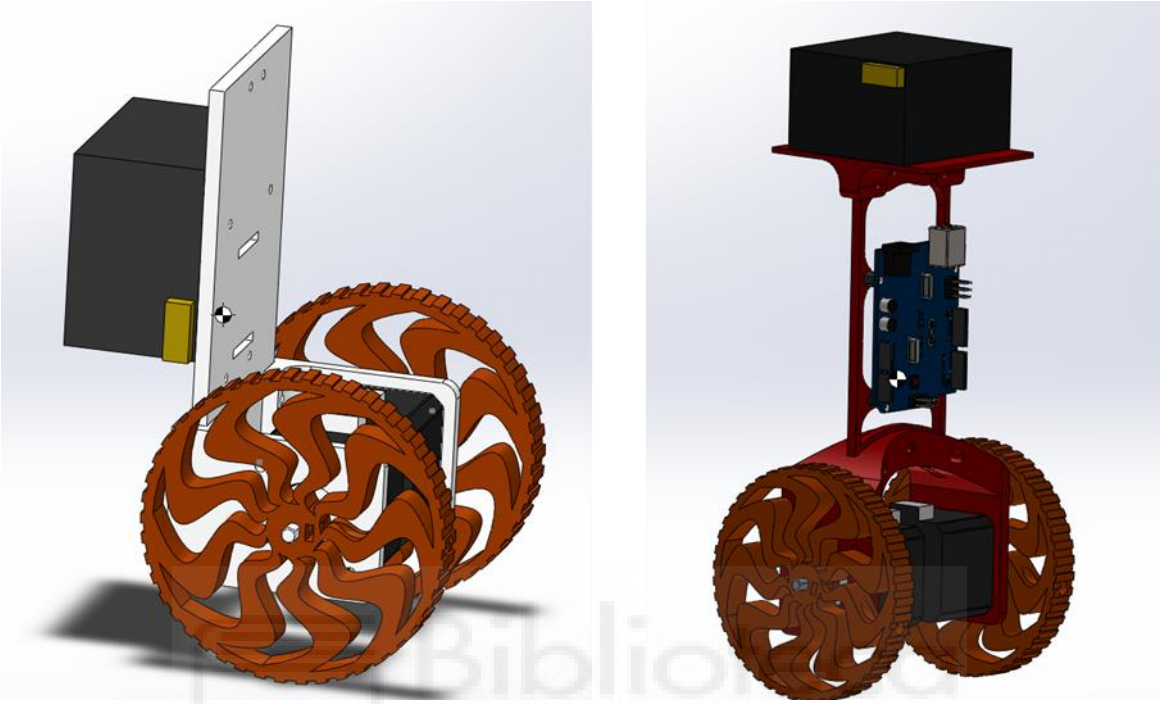


Figura 9. Diseño 1 y Diseño 2 del robot

Para poder cerrar el lazo de control de nuestros controladores necesitamos sensores, pero para poder usar nuestros sensores necesitamos escalarlos y calibrarlos correctamente, igual pasa con los motores necesitamos escalar la señal que vamos a introducir con lo correspondiente a la salida de movimiento de estos. Uno de los problemas que he encontrado en el desarrollo del proyecto es que tanto los sensores como equipos de medida usados para calibrar estos sensores y actuadores son de bajo coste, ya que este proyecto se ha realizado a nivel personal. Esto provoca que a la hora de hacer nuestro proyecto final tengamos que trabajar más en el ajuste final del programa tanto controladores como señales escaladas.

4 CARACTERIZACIÓN DE SENSORES Y ACTUADORES

Para que nuestro controlador pueda interpretar las señales que le llegan de los sensores y pueda enviar una señal correcta a los controladores, primero debemos caracterizar correctamente las diferentes señales que envían los sensores y conocer como responden nuestros actuadores, los motores, ante diferentes señales de salida de nuestro controlador, en la Figura 10 se puede ver un diagrama de cómo se comunica nuestro controlador con los sensores y actuadores.

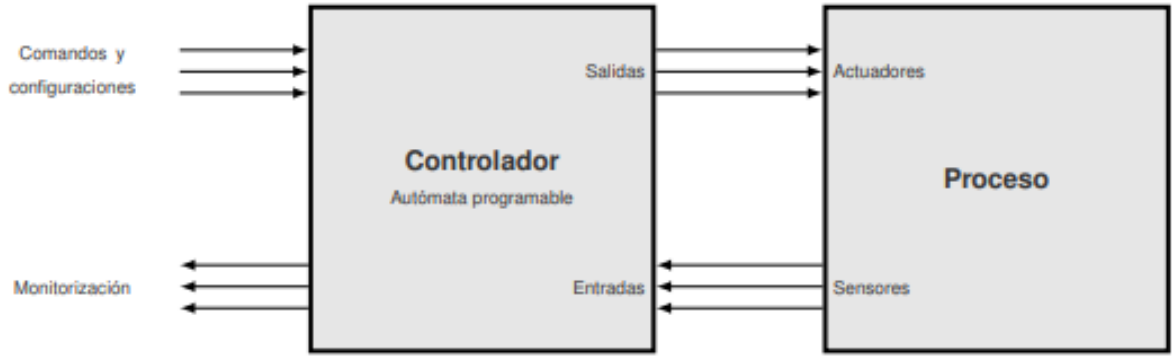


Figura 10. Diagrama de bloque de un proceso automatizado

La caracterización de algunos parámetros de nuestros actuadores y sensores nos servirá también en los siguientes apartados, para cálculos relacionados con el modelado del sistema y su simulación.

Para la caracterización, calibración y escalado de nuestras señales de entrada y salida en nuestro controlador he realizado unos experimentos con diferentes equipos de medida. Realizando varias medidas he conseguido la calibración de estas señales, a continuación, se explican estos experimentos.

4.1 Motor. Calculo relación par intensidad de nuestro motor DC12V200RPM

Para conocer la constante de la ecuación que relaciona la intensidad con el par "Ki" en la ecuación (1):

$$\Gamma_m = K_i * i \qquad (1)$$

Deberemos realizar un experimento para saber el par que realiza nuestro motor con una cierta intensidad a un voltaje constante, para ellos mediante el microcontrolador Atmega328 haremos un programa que grafique la relación Par-Intensidad de nuestro motor a voltaje constante y diferentes pares, con ello podremos sacar la constante par-intensidad K_i de la ecuación (1). Para ello necesitamos usar 2 sensores uno para medir el par del motor y otro para medir la intensidad del motor.

Mediante el Código 1 vamos enviando, por el puerto serial de nuestro controlador, el valor transformado y escalado a par de la celda de carga, y la intensidad también transformada y

escalada del sensor ACS712. Mediante SolidWorks sacamos fácilmente la distancia a la que se aplica la fuerza en la galga estequiométrica y el centro del eje del motor, necesario para sacar el par.

Código 1. Par-Intensidad

```
#include <Arduino.h>
#include "HX711.h"

float Sensibilidad=0.18359; //sensibilidad en Voltios/Amperios
float Masa=0;
float Fuerza=0;
float Par=0;
const int DOUT=A2;
const int CLK=A1;
float get_voltage(int n_muestras);
HX711 balanza;

void setup() {
  Serial.begin(9600);
  balanza.begin(DOUT, CLK);
 // Serial.print("Lectura del valor del ADC:  ");
  Serial.println(balanza.read());
 // Serial.println("No ponga ningún objeto sobre la balanza");
 // Serial.println("Destarando...");
  //Serial.println("...");
  balanza.set_scale(113930.25); // Establecemos la escala, 439430.25
  balanza.tare(20);  //El peso actual es considerado Tara.

  //Serial.println("Listo para pesar");
}

void loop() {
   float voltajeSensor = get_voltage(7808);//obtenemos voltaje del
sensor(7808 muestras)
  // Serial.print("Corriente: ");
  //Serial.print(",");
  Serial.print(voltajeSensor ,3); //Vamos a tomar 3 decimales del valor
  Serial.println(",");
  //Serial.printl(" A");
  //Serial.print(",");
  //Serial.print("Par: ");
  //Serial.print(",");
  Masa=balanza.get_units(20),3;
  Fuerza=Masa*9.807;
  Par=Fuerza*0.042;
  Serial.print(Par);
  Serial.print(",");
  //Serial.print(" N.m");
  //Serial.print(",");
```

```
  //delay(500);
}
float get_voltage(int n_muestras)
{
  float voltajeSensor=0;
  float corriente=0;

  for(int i=0;i<n_muestras;i++)
  {
      voltajeSensor = analogRead(A0) * (5.0 / 1023.0);//Las entradas
analogicas tiene un valor en tre 0 y 1023 esto corresponte a una tension
en suentrada de 0 a 5 voltios
                  corriente=corriente+(voltajeSensor-2.494)/Sensibilidad;
//Ecuación  para obtener la corriente
  }
     corriente=corriente/n_muestras; //Sacamos el valor medio de 7808
muestras
    return(corriente);
}
```

Lo resultado de par e intensidad que obtuvimos en el experimento lo pudimos grabar directamente (Tabla 1), en Excel mediante el complemento de Excel "Microsoft Data Streamer for Excel" que se comunica con el microcontrolador por el puerto serial de este, en tiempo real.

La prueba consto de ir aumentado la tensión en el motor de 0 a 12V, el motor tenía un vástago el cual estaba conectado al eje del motor, este vástago estaba apoyado sobre la galga extensiométrica (Figura 21), al ir aumentando la tensión en el motor también aumenta la intensidad en este y el par que se aplica en la galga. Tras varias medidas de intensidad y par grabadas en la tabla de Excel paramos la prueba. Cuando ya tuvimos estos valores pudimos realizar la gráfica de la Figura 11, en la que mediante una línea de tendencia de esta, pudimos sacar la constante "Ki" de la ecuación (1).

$$Ki=0.156$$

Tabla 1. Experimento Par-intensidad

	Fuerza	Intensidad (A)	Fuerza (N)
15:43:29,95	0,587	1,544	0,242
15:43:26,93	0,592	1,544	0,244
15:43:23,91	0,59	1,554	0,243
15:43:20,89	0,598	1,467	0,246
15:43:17,87	0,618	1,481	0,255
15:43:14,85	0,597	1,463	0,246
15:43:11,83	0,627	1,451	0,258
15:43:08,81	0,549	1,463	0,226
15:43:05,79	0,554	1,233	0,228

15:43:02,77	0,555	1,236	0,229
15:42:59,75	0,561	1,167	0,231
15:42:56,74	0,567	1,182	0,234
15:42:53,72	0,573	1,105	0,236
15:42:50,70	0,469	1,123	0,193
15:42:47,68	0,439	0,901	0,181
15:42:44,66	0,399	0,904	0,164
15:42:41,64	0,377	0,817	0,155
15:42:38,62	0,376	0,813	0,155
15:42:35,60	0,359	0,755	0,148
15:42:32,59	0,313	0,587	0,129
15:42:29,57	0,31	0,575	0,128
15:42:26,55	0,21	0,53	0,086
15:42:23,53	0,171	0,446	0,070
15:42:20,51	0,168	0,209	0,069
15:42:17,49	0,17	0,205	0,070
15:42:14,47	0,171	0,209	0,070
15:42:11,36	0,084	0,157	0,035
15:42:08,26	0,002	-0,002	0,001
15:42:05,15	0,002	-0,002	0,001

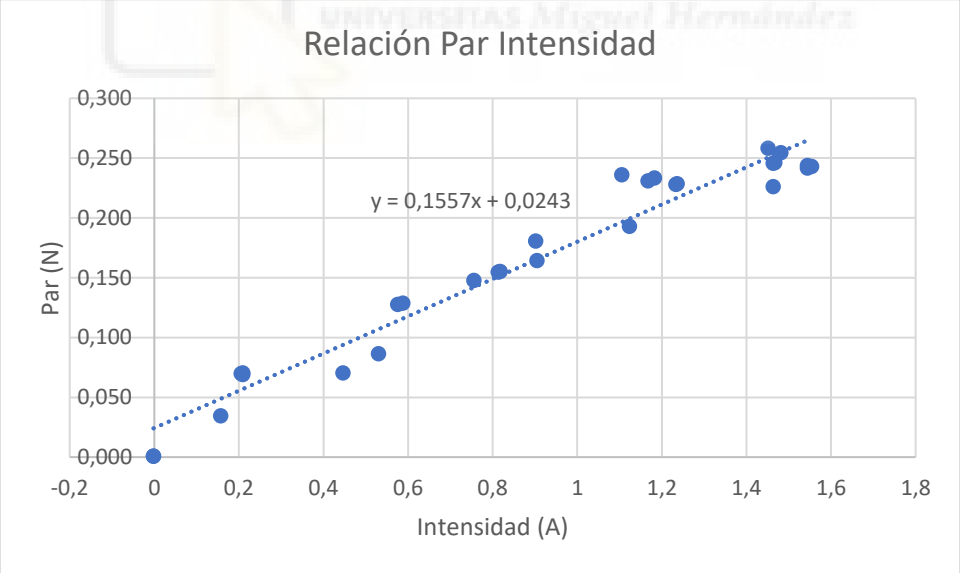


Figura 11. Grafica relación par intensidad

4.1.1 Par (Γ_m)

Para el cálculo de la fuerza aplicada, usamos una celda de carga, Figura 12.

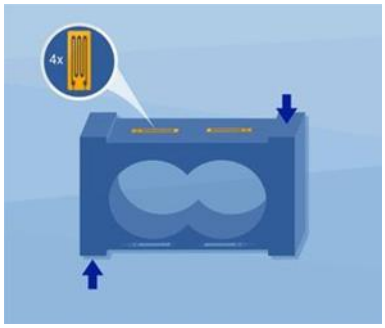


Figura 12. Célula de carga monoplato

Una celda de carga está compuesta por 1 o más galgas extensiométricas, Figura 13, colocadas en un material con cierta flexibilidad, en nuestro caso es un bloque de aluminio, el cual tiene una cierta elasticidad conocida. Según donde coloquemos las galgas extensiométricas podremos medir la fuerza en diferentes direcciones, en nuestro caso tenemos una célula de carga monoplato con 4 galgas extensiométricas, las galgas son conductores eléctricos firmemente unidos a una película, siguiendo un patrón sinuoso. Cuando se tira o se comprime esta película, esta, junto con los conductores, se alarga o se comprimen. Esta deformación hace tener una pequeña variación en la resistencia de los conductores eléctricos, que forman la galga, esta deformación es proporcional a la variación de la resistencia. Al estar las galgas colocadas sobre un material con una cierta elasticidad, en nuestro caso aluminio, las galgas llegaran a deformarse variando su resistencia.

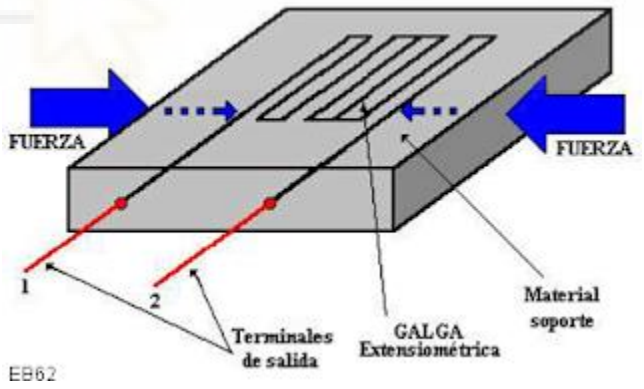
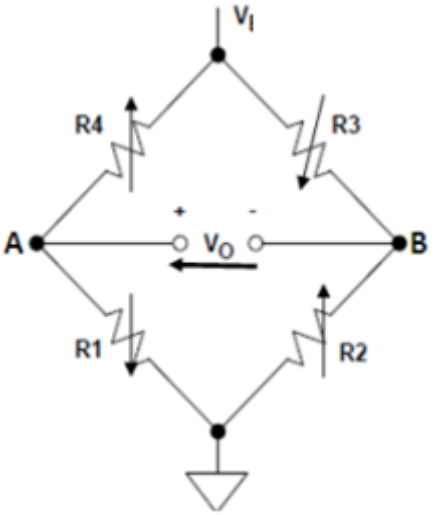


Figura 13. Galga extensiométrica

El problema es que la variación de resistencia de estas galgas es ínfima y nos hacer muy difícil medir directamente esta variación. Para solucionar estos problemas usamos un puente de Wheatstone, Figura 14, en el cual mediremos directamente la diferencia de tensión en un cierto punto del puente, mediante esta diferencia de tensión podremos saber la fuerza que hemos aplicado a nuestro bloque de metal. Por ejemplo, si no aplicamos ninguna fuerza al bloque de aluminio el valor de las resistencias será siempre el mismo y no tendremos diferencia de potencial a la salida de puente, es decir fuerzas igual a cero, en el momento que apliquemos una fuerza en un sentido la diferencia de potencial a la salida ira variando.



$$V_0 := V_A - V_B$$

$$V_A := \frac{V_I \cdot R_1}{R_4 + R_1}$$

$$V_B := \frac{V_I \cdot R_2}{R_3 + R_4}$$

$$V_0 := V_I \cdot \left(\frac{R_1}{R_4 + R_1} - \frac{R_2}{R_3 + R_4} \right)$$

Figura 14. Puente de Wheatstone y las ecuaciones que lo caracterizan

Al igual que cualquier material las galgas extensiométricas y el aluminio se comprimen y se expanden con el cambio de temperatura. Al ser tan pequeña la variación en la resistencia de las galgas estequiométricas, una pequeña diferencia de temperatura haría que este valor de resistencia variara, sin aplicar ninguna fuerza en la galga extensiométrica. Para solucionar este problema colocamos ciertas resistencias junto al puente de Wheatstone para compensar. También colocaremos resistencias para compensar la variación que se produce entre las galgas por el simple hecho de que el bloque de metal tiene un peso, en la Figura 15 se pueden ver las resistencias de compensación.

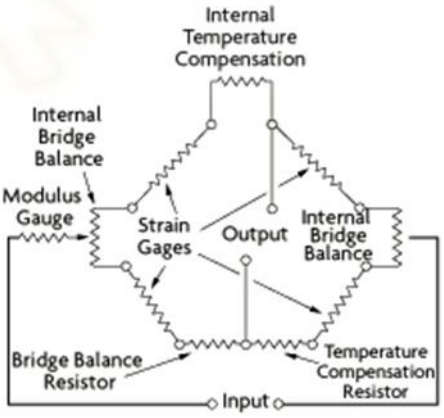


Figura 15. Resistencias de compensación del puente de Wheatstone

Tanto el puente de Wheatstone como las resistencias de compensación están integradas en la propia celda de carga.

Para poder comunicarse el puente de Wheatstone con el microcontrolador usaremos el integrado HX711, este es un conversor analógico a digital de 24 bits, este recibe las tensiones de VA y VB, Figura 14, y mediante un amplificador diferencial es capaz de medir y amplificar esta señal, podremos modificar la ganancia mediante la señal de reloj del microcontrolador. El Figura 16 se ve el diagrama del conversor amplificador HX711.

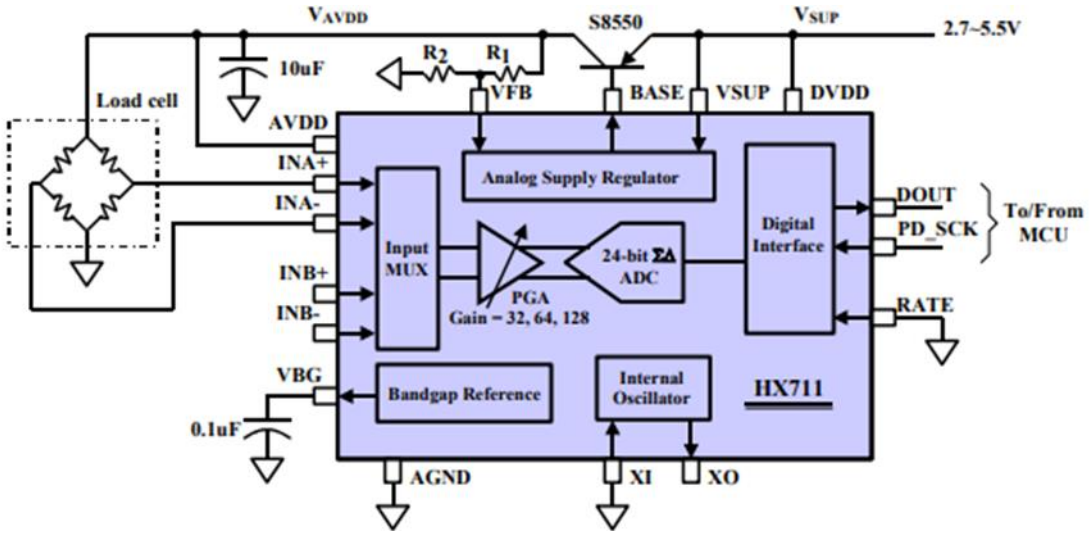


Figura 16. Diagrama del conversor amplificador HX711

Antes de poder usar la célula de carga en el experimento tenemos que escalar y calibra su medida, para ello hemos realizado el Código 2, en C++:

Código 2. Escalado célula de carga

```
#include <Arduino.h>
#include "HX711.h"

const int DOUT=A1;
const int CLK=A0;
float Masa=0;
float Fuerza=0;


HX711 balanza;

void setup() {
  Serial.begin(9600);
  balanza.begin(DOUT, CLK);
  Serial.print("Lectura del valor del ADC:  ");
  Serial.println(balanza.read());
  Serial.println("No ponga ningun  objeto sobre la balanza");
  Serial.println("Destarando...");
  Serial.println("...");
  balanza.set_scale(113930.25); // Establecemos la sensibilidad.

  balanza.tare(20);  //El peso actual es considerado Tara.

  Serial.println("Listo para pesar");
}

void loop() {
  Serial.print("Peso: ");
  Masa=balanza.get_units(20),3;
```

```
  Fuerza=Masa*9.807;
  Serial.print(Fuerza);
  Serial.println(" kg");
  delay(500);
}
```

Para calibrar la galga fuimos colocando unos pesos de control, los cuales conocemos su masa, en la célula de carga, en función del valor que nos da el controlador y el peso real de los pesos de control vamos ajustando el valor de "balanza.set_scale()" para que se acerque lo máximo posible al valor que tendría que estar marcando. Esta medida la realizamos varias veces con diferentes pesos, ajustando el valor de "balanza.set_scale()" hasta que se aproxime lo máximo posible al valor correcto.

4.1.2 Intensidad (i)

Para medir la intensidad que está consumiendo el motor use el sensor de corriente ACS712. El ACS712, es un dispositivo que puede medir la corriente eléctrica que pasa por su interior, tanto alterna como continua, hasta un máximo de 30A, en mi caso use un sensor que puede llegar a medir hasta 5A, que para mi caso es más que suficiente. La corriente pasara por dentro del sensor a través de un conductor de cobre el cual tiene una resistencia interna muy baja, 1.2mOhm, no afecta a nuestra medida, fue despreciada. Su tecnología está basada en el efecto Hall, Figura 17, la corriente aplicada que fluye a través de esta ruta de conducción de cobre genera un campo magnético que es detectado por el Hall IC integrado y convertido en un voltaje proporcional. Gracias a esta tecnológica de efecto Hall, que explico más adelante, nuestro sensor puede "transformar" la corriente en voltaje, el sensor está aislado galvánicamente, separando la parte de potencia y de medida, pudiendo hacer que el controlador trabaje fácilmente con grandes intensidades, sin poner en riesgo su integridad.

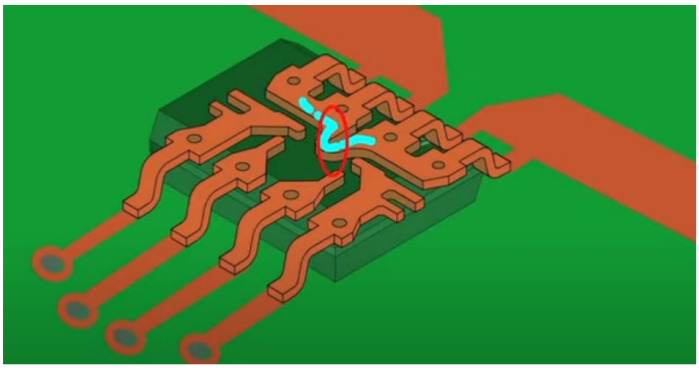


Figura 17. Sensor ACS712, la línea azul representa la corriente y en ovalo rojo representa campo creado

En la Figura 18 podemos ver una representación del efecto Hall el cual es un fenómeno físico que consiste en la aparición de un campo eléctrico (campo Hall, V_{Hall}) en un conductor por el que circula una corriente eléctrica, I_{Hall}, cuando se aplica un campo magnético, $B_{Magnetic}$, perpendicular a la dirección de la corriente. Este campo eléctrico, V_{Hall}, se debe a la separación de cargas que se produce por la fuerza magnética que actúa sobre los portadores de corriente. El campo Hall es proporcional al producto de la intensidad de la corriente, I_{Hall}, y la del campo magnético $B_{Magnetic}$.

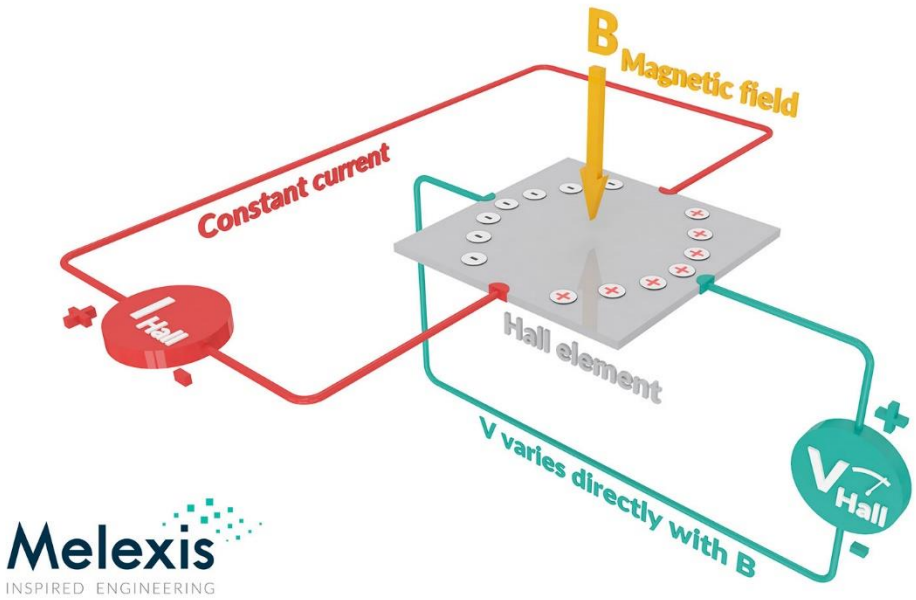


Figura 18. Representación del efecto Hall

Antes de empezar el experimento para calcular la relación entre Par e intensidad, tuve que conocer la sensibilidad del sensor ACS712 y su offset, para ello coloqué en serie diferentes resistencias de potencia y un multímetro, para conocer la intensidad real. Con un voltaje constante fui colocando diferentes resistencias para obtener diferentes intensidades, medí mediante programación y un microcontrolador el voltaje que iba produciendo el sensor. Con estos valores realice una gráfica, con el voltaje del sensor en las ordenadas y con la intensidad medida en el multímetro en abscisas, y saque la línea de tendencia de esta grafica. La pendiente de esa línea de tendencia será nuestra sensibilidad y donde corte la recta el eje de ordenadas será nuestro offset. En el Código 3 se puede ver el programa creado para el calculo de la sensibilidad y offset del sensor ACS712.

Código 3. Medidas sensor ACS712

```
#include <Arduino.h>
// Prototipo de función
float get_voltage(int n_muestras);

void setup() {

  Serial.begin(9600);
}

void loop() {

    float voltajeSensor = get_voltage(7808);//obtenemos voltaje del
sensor(7808 muestras)
  Serial.print("Voltaje del sensor: ");
  Serial.println(voltajeSensor ,3);
}
```

```
float get_voltage(int n_muestras)
{
  float voltage=0;

  for(int i=0;i<n_muestras;i++)
  {
      voltage  =voltage+analogRead(A0)  *  (5.0  /  1023.0);//Las  entradas
analógicas tiene un valor entre 0 y 1023 esto corresponde a una tensión en
su entrada de 0 a 5 voltios
  }
  voltage=voltage/n_muestras;//Sacamos el valor medio de 10000 muestras
  return(voltage);
}
```

Los datos de voltaje que saquemos del sensor mediante el controlador los pasaremos a Excel con el complemento de Excel "Microsoft Data Streamer for Excel" que se comunica con el microcontrolador por el puerto serial de este, en tiempo real.

```
Serial.begin(9600)
```

Deberemos configurar el número de muestras de nuestro código en función del número de muestras que queramos en nuestro excel en nuestro caso será de 1 muestra por segundo. Conociendo que la función analogRead() tarda unos 127.43 microsegundos en leer un valor analógico. Si tomamos 7808 muestras como tenemos puesto en el código, llamaremos 7808 veces a analogRead() dentro de la función get_voltage(), el tiempo total será de 995000 microsegundos o 0.995 segundos. Además de esto debemos tener en cuenta a la velocidad que configuramos el puerto serial, Serial.begin(), en nuestro código lo ponemos a 9600, la velocidad será de 9600 bits por segundos. Sabiendo que la función Serial.println() usa el formato ASCII que envía 8 bits por cada carácter, más un carácter de fin de línea. Si se usan 3 decimales, por cada valor de voltaje este tendrá 5 caracteres (por ejemplo, 3.141), por lo tanto, el tiempo que tardará en enviar el valor será de (5 + 1) * 8 / 9600 = 0.005 segundos. Esto sumado al retardo de tomar 7808 muestra nos da que tardara 1 segundo en cada iteración del bucle principal.

Los datos de intensidad los cogeremos mediante el multímetro que estará conectado en serie al circuito, el multímetro que voy a usar, el UNI-T UT61E, este tiene la opción de conectarse y recopilar los datos en tiempo real en formato compatible con Excel, el software es el mostrado en Figura 19.

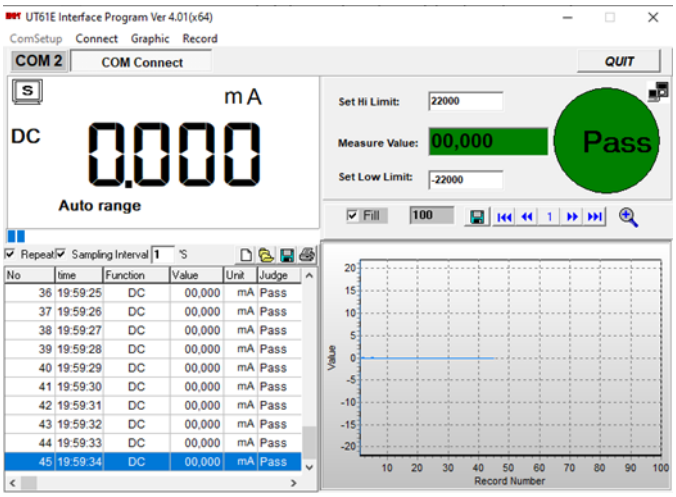


Figura 19. Software UT61E

Una vez que tengamos los datos en Excel del voltaje medido por el sensor ACS712 y los amperios medidos por el multímetro podremos fácilmente crear una gráfica, con la sensibilidad del sensor y su offset. La Figura 20 muestra la gráfica de los resultados de esos valores, donde la pendiente de la línea de tendencia es la sensibilidad 0.18359V/A y el offset es 2.4987V.

Tabla 2. Experimento Intensidad

Nº	Time	Multímetro (A)	Medida ACS712 (V)
1	21:03:00	0	2,501
2	21:03:01	0	2,501
3	21:03:02	0	2,501
4	21:03:03	0	2,501
5	21:03:04	0	2,501
6	21:03:05	0,03	2,502
7	21:03:06	0,03	2,502
8	21:03:07	0,03	2,502
9	21:03:08	0,03	2,502
10	21:03:09	0,03	2,502
11	21:03:10	0,049	2,506
12	21:03:11	0,049	2,506
13	21:03:12	0,049	2,506
14	21:03:13	0,049	2,506
15	21:03:14	0,049	2,506
16	21:03:15	0,085	2,514
17	21:03:16	0,085	2,514
18	21:03:17	0,085	2,514
19	21:03:18	0,085	2,514
20	21:03:19	0,085	2,514
21	21:03:20	0,13	2,522
22	21:03:21	0,13	2,522
23	21:03:22	0,13	2,522

24	21:03:23	0,13	2,522
25	21:03:24	0,13	2,522
26	21:03:25	0,195	2,535
27	21:03:27	0,195	2,535
28	21:03:28	0,195	2,535
29	21:03:29	0,195	2,535
30	21:03:30	0,195	2,535
31	21:03:31	0,259	2,548
32	21:03:32	0,259	2,548
33	21:03:33	0,259	2,548
34	21:03:34	0,259	2,548
35	21:03:35	0,259	2,548
36	21:03:36	0,386	2,571
37	21:03:37	0,386	2,570
38	21:03:38	0,386	2,571
39	21:03:39	0,386	2,571
40	21:03:40	0,386	2,571
41	21:03:41	0,722	2,630
42	21:03:42	0,721	2,630
43	21:03:43	0,721	2,630
44	21:03:44	0,721	2,630
45	21:03:45	0,721	2,630

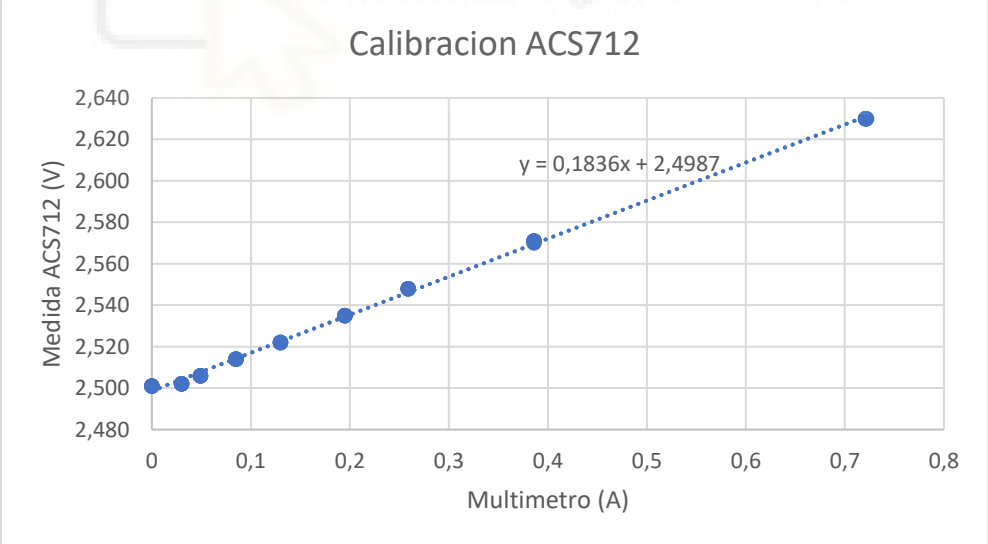


Figura 20. Grafica calibración sensor ACS712

Conociendo la sensibilidad y el offset del sensor pude medir correctamente la intensidad que pasaba por este sensor para obtener la relación par-intensidad del motor, con el Código 4 podemos hacer la lectura de la intensidad del sensor ACS712 con las pertinentes correcciones de sensibilidad y offset.

Código 4. Medida del sensor de Intensidad

```
#include <Arduino.h>

float Sensibilidad=0.18359; //sensibilidad en Voltios/Amperios
// Prototipo de función
float get_voltage(int n_muestras);

void setup() {

  Serial.begin(9600);
}

void loop() {

   float voltajeSensor = get_voltage(7808);//obtenemos voltaje del
sensor(7808 muestras)
   Serial.println(voltajeSensor ,3); //Vamos a tomar 3 decimales del
valor
}

float get_voltage(int n_muestras)
{
  float voltajeSensor=0;
  float corriente=0;

  for(int i=0;i<n_muestras;i++)
  {
      voltajeSensor = analogRead(A0) * (5.0 / 1023.0);//Las entradas
analogicas tiene un valor en tre 0 y 1023 esto corresponte a una tension
en suentrada de 0 a 5 voltios
                corriente=corriente+(voltajeSensor-2.5014)/Sensibilidad;
//Ecuación  para obtener la corriente
  }
     corriente=corriente/n_muestras; //Sacamos el valor medio de 7808
muestras
    return(corriente);
}
```

4.1.3 Banco de pruebas, experimento par-intensidad

Para la realización la medición de la variable par-intensidad realice un banco de pruebas para poder tener controladas todas las variables que pueden afectar a las medidas tomadas por los sensores, el diseño de este banco de prueba se ha realizado en SolidWorks Figura 21, tras realizar el diseño 3d se procedió a imprimir las piezas en una impresora FDM, Figura 22.

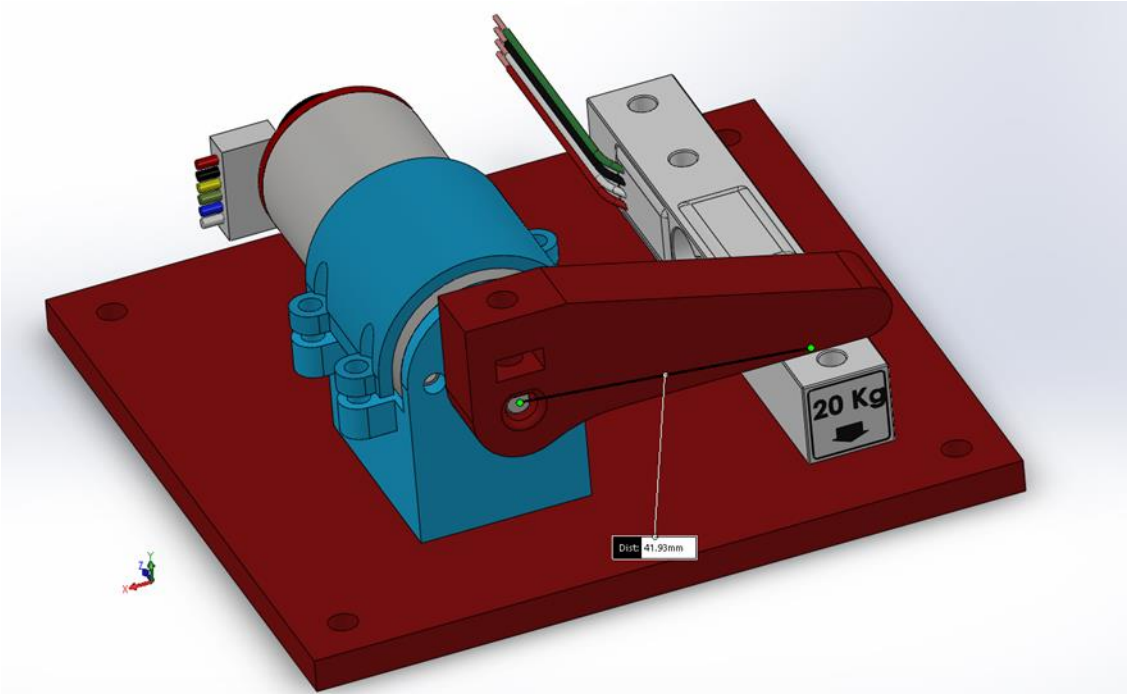


Figura 21. Diseño banco de pruebas par-intensidad

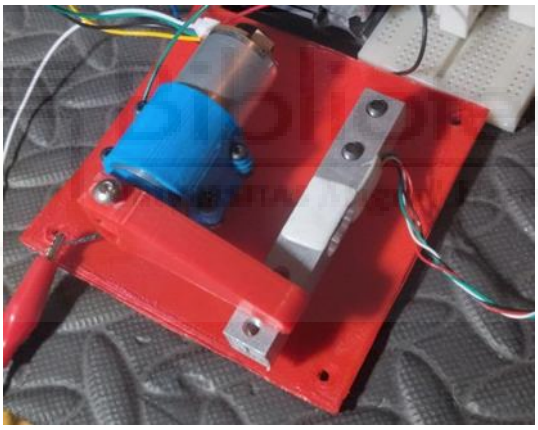


Figura 22. Banco de pruebas

4.2 Encoder. Calculo constante de velocidad DC12V200RPM

Para saber la constante de velocidad del motor, K_V, que aparece en la ecuación (2), y se explica más adelante, primero deberemos conocer los valores de velocidad angular y voltaje de alimentación del motor (ecuación (3)).

$$V_e = R_m * i + L_m * \frac{di}{dt} + K_V * W_1 \quad (2)$$

$$K_V = \frac{V_e}{\omega} \quad (3)$$

He realizado un experimento en el cual fui variando los valores de voltaje mediante una fuente de alimenta, a la vez iba registrado este valor de voltaje y de velocidad. Para ello mediante el microcontrolador Atmega328 he creado un programa bajo el entorno de desarrollo Visual Studio y la extensión PlatformIO. Este programa se encarga de registrar la velocidad de un enconder que lleva acoplado el motor DC a la vez que registra la tensión de entrada del motor, mediante un divisor de tensión, la tensión de entrada al motor la regule externamente con una fuente de tensión. Los datos de velocidad y tensión los he registrado mediante el complemento Teleplot de la extensión PlatformIO, el cual grafica todos los valores, en tiempo real desde el momento que le indiqué (Figura 23), estos valores los pude exportar directamente desde el complemento en formato CSV.

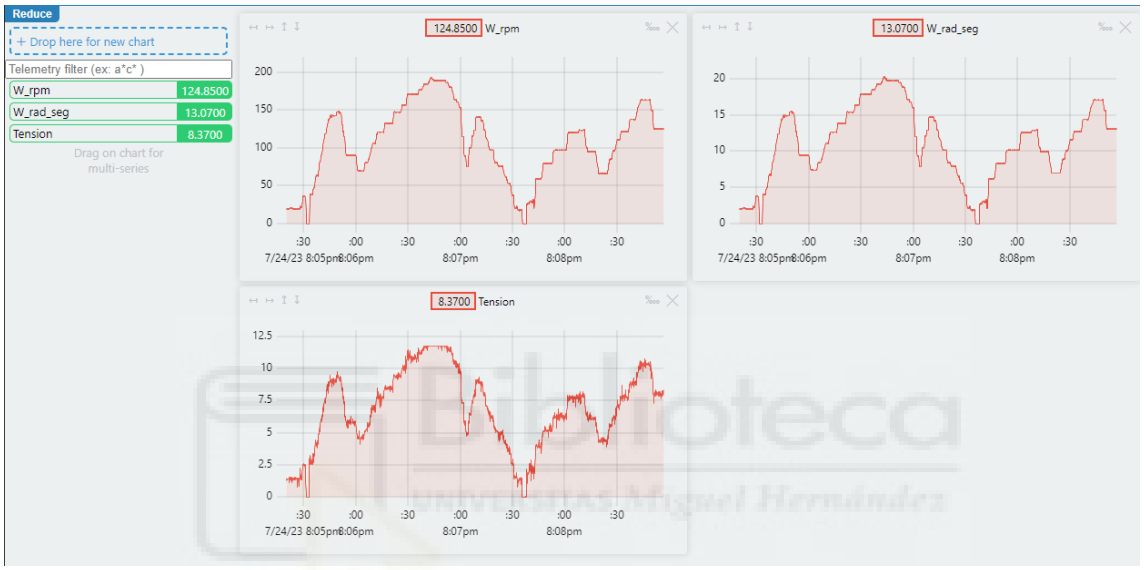


Figura 23. Interfaz complemento Teleplot

Una vez que tuvimos los datos en formato CSV los pudimos abrir directamente en Excel y graficar estos valores (Figura 24), enfrentando la tensión y la velocidad angular. Con ello obtuve la línea de tendencia y su ecuación correspondiente. Dividendo las variables de esta ecuación se obtiene el valor de Kv (ecuación (3)).

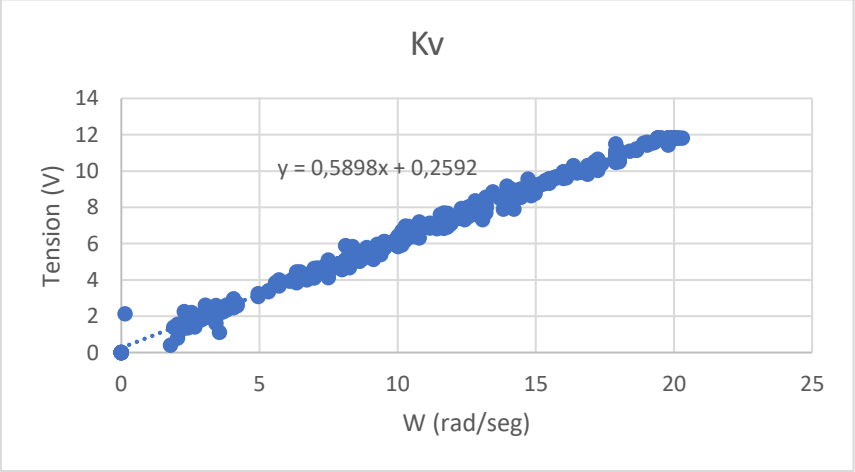


Figura 24. Relación tension-velocidad angular

El motor que voy a usar para este proyecto es un motor de 12VDC con reductora y enconder acoplado, en la Figura 25 podemos ver la distribución de estos componentes. Las características más importantes de nuestro motor en relación con el experimento son las siguientes:

Max. Velocidad = 180rpm Relación de reducción 1/45 IMP= 11 pulsos/vueltas

La Max.Velocidad es la velocidad máxima que podrá alcanzar nuestro motor a la salida del eje de carga. La relación de reducción, calculada en el punto 4.2.1, es la relación que existe entre las vueltas dadas en el eje del motor y el eje de carga, en nuestro caso para que el eje de carga de 1 revolución es necesario que el eje del motor de 45 revoluciones. IMP o impulsos es el número de pulsos o señales que manda el enconder al dar una vuelta.

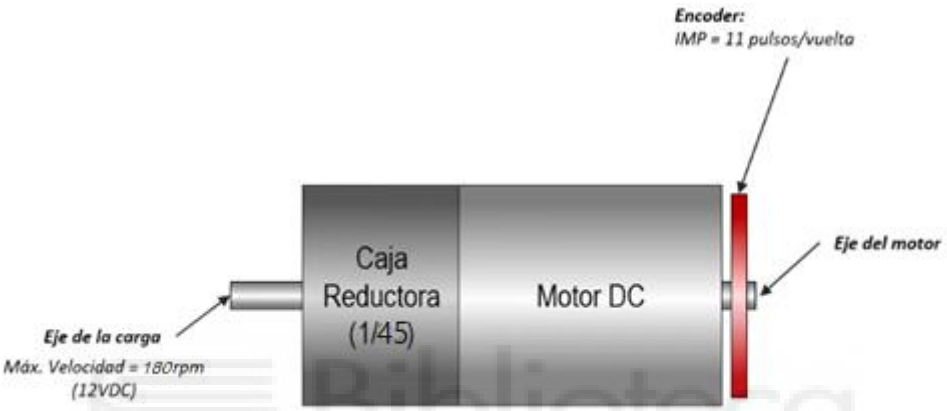


Figura 25. Representación Motor 12V200RPM

4.2.1 Velocidad angular (W)

El tener un encoder en el eje del motor facilita la obtención de datos de posición y velocidad del motor, para ello tengo en cuenta que la velocidad a la que gira el enconder no es la velocidad a la que gira el eje que va conectado a la rueda. El enconder va conectado directamente al motor DC en cambio el eje que se conecta a la rueda lleva entre el motor dc y el eje de salida una caja reductora 1/45, a la hora de realizar el código y sacar los datos tuve en cuenta esto.

El encoder de nuestro motor es de tipo incremental y magnético, se compone de un disco magnético con múltiples polos, el cual está conectado al eje de un motor DC y dos sensores de efecto Hall. Durante el funcionamiento del motor DC, el disco magnético gira, y en ese proceso, los polos magnéticos del disco pasan frente a los sensores Hall. Cada vez que un sensor detecta un polo magnético positivo, emite un pulso.

El disco magnético posee 22 polos alternados, lo que significa que por cada vuelta completa del motor, cada sensor Hall emitirá 11 pulsos, estos son los conocidos IMP del encoder. Los sensores Hall A y B (Figura 26) están desplazados entre sí 90º, y sus salidas generan ondas cuadradas, que también tienen un desfase de 90º. Este tipo de configuración se denomina "codificador de cuadratura".

El desfase de las ondas cuadradas es de gran utilidad para determinar el sentido de giro del motor. Al girar en una dirección, la onda A precederá a la onda B, mientras que, al girar en la dirección opuesta, la onda B precederá a la A (Figura 26).

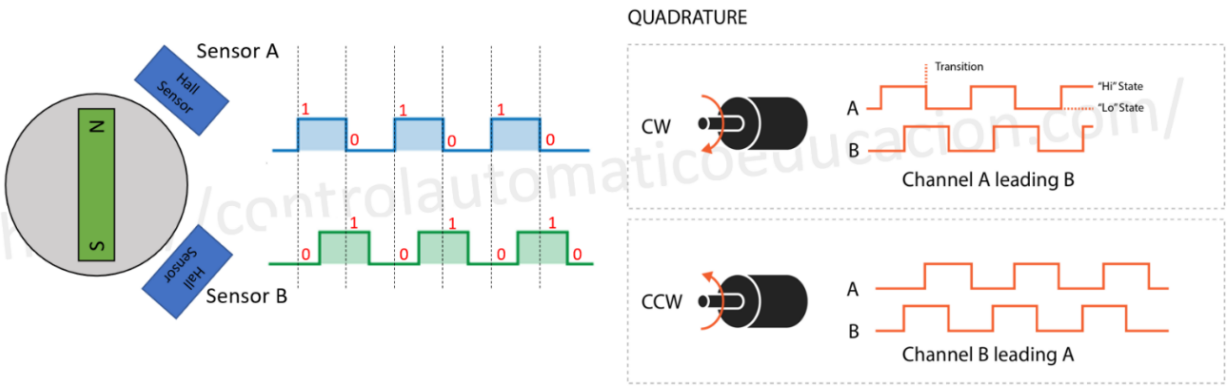


Figura 26. Encoder codificación de cuadratura

En el Código 5, donde se ve parte del código para el caculo de K_v, se puede apreciar que "contador" ira contando cada flanco de subido del "pinA", gracias la función "attachInterrupt()", el cual ejecuta la función "interrupción" cada vez que tenga un flanco de subida en el "pinA", este pin es la entrada de señal en forma de pulsos del encoder. Según la hoja de características, el encoder tiene un IMP de 11 con lo que la señal de "contador" la tengo que dividir entre 11 para que en cada 11 flancos de subida cuente 1 revolución del encoder, también tengo una caja reductora 1/45 conectada entre la rueda y el motor DC. Con lo que para que la rueda de una vuelta el encoder deberá dar 45, este valor también se lo dividimos a "contador", por último tuvimos que pasar el valor resultante de vueltas o revoluciones a radianes, para ello multiplicamos este valor por "2xπ".

El tiempo transcurrido lo obtengo mediante la función "millis()", la función "millis()" es una función que devuelve el tiempo en milisegundos desde que el programa comenzó a ejecutarse. El "if" del Código 5 actualiza los datos de velocidad angular "W_rad_seg", este "if" se ejecuta cuando el tiempo desde que se enciende el microcontrolador menos una variable que ha guardado el tiempo que lleva encendido el microcontrolador es mayor o igual a una variable que nosotros indiquemos en nuestro caso esta variable es de 100, con esto conseguimos que el "if" se ejecute cada 0.1 segundos, para pasarlos a segundo multiplicaremos la actualización de velocidad angular por 10.

Con todo esto podremos registrar el valor de velocidad angular que sale por nuestro eje y llega a la rueda, esta es la variable "W_rad_seg".

Código 5. Parte del código para el cálculo de K_V

```
int pinA=3;
```

```
     attachInterrupt(digitalPinToInterrupt(pinA),interrupcion,RISING);
//Flanco subida Pin3
```

```
void interrupcion() {
  contador++;
}
```

```
unsigned long previousMillis = 0;
long interval = 100; //0.1 segundo cada medida
```

```
  unsigned long currentMillis = millis();
```

```
  if ((currentMillis - previousMillis)>= interval)
  {
    previousMillis = currentMillis;
    W_rad_seg=10*contador*((1/(11*45.0))*2*3.14);
    contador=0;
  }
```

4.2.2 Tensión de entrada al motor (V_e)

Nuestro motor se alimenta entre 0 a 12V, las entradas analógicas del microcontrolador admiten una tensión de entrada de 0 a 5V como máximo. Para sacar la constante Kv tuve que registrar la tensión de entrada en el motor para ello usé un divisor de tensión como el que se ve en la Figura 27, con ellos pude bajar la tensión 12V a 5V. Si lo alimentamos directamente probablemente dañaríamos el microcontrolador.

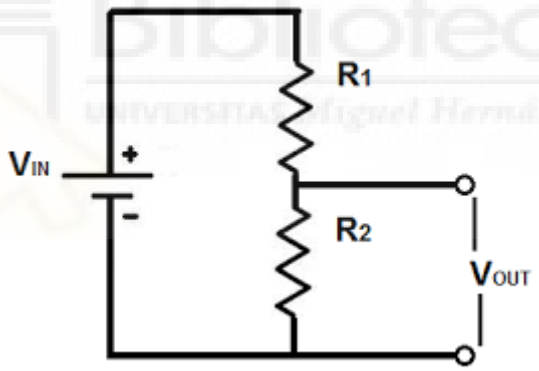


Figura 27. Divisor de tensión

Los cálculos para sacar R1 y R2 son sencillos, usando por ejemplo 100Ω para la resistencia R2, tendríamos que usar una resistencia de 140Ω según la ecuación (4), con esto tendríamos 12V en Vin y 5V en Vout.

$$V_{out} = \frac{R_1}{R_1 + R_2} * V_{in} \qquad (4)$$

4.2.3 Banco de pruebas, experimento velocidad angular-tensión

Debajo se puede apreciar una foto del experimento para el cálculo de K_V (Figura 28).

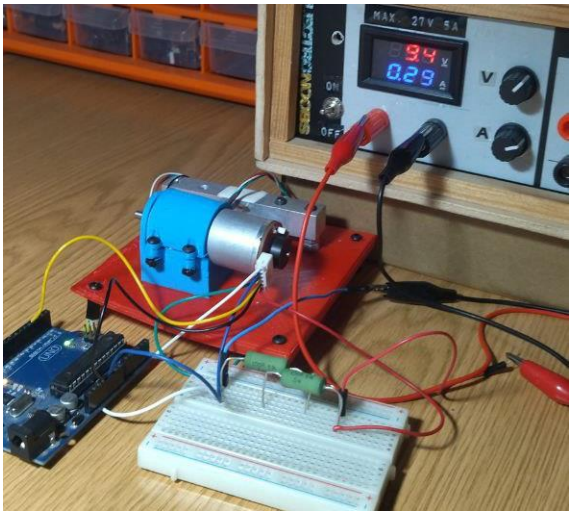


Figura 28. Banco de pruebas relación W-Ve

4.2.4 Cálculo de la relación de transformación de la caja reductora

El cambio de diseño del diseño 2 al diseño 3, que se explica en los próximos puntos, hizo que cambiáramos de motores y pasamos de unos motores paso a paso Nema17 a unos motores DC de 12V con enconder, estos nuevos motores incorporan una caja reductora. Uno de los problemas que me encontré a la hora de caracterizar el conjunto del motor fue que la relación entre la entrada y salida de la caja reductora no apareció en la documentación donde se compraron estos motores, tampoco el motor tenía ninguna referencia inscrita con la que buscar más información del motor. Por este motivo me vi en la obligación de desmotar esta caja reductora y calcular su relación de transformación. La disposición de la cadena de engranajes de la caja reductora es la mostrada en la Figura 29, estando acoplado el engranaje de la izquierda del todo al eje del motor y el engranaje de la derecha del todo al eje acoplado a la rueda.

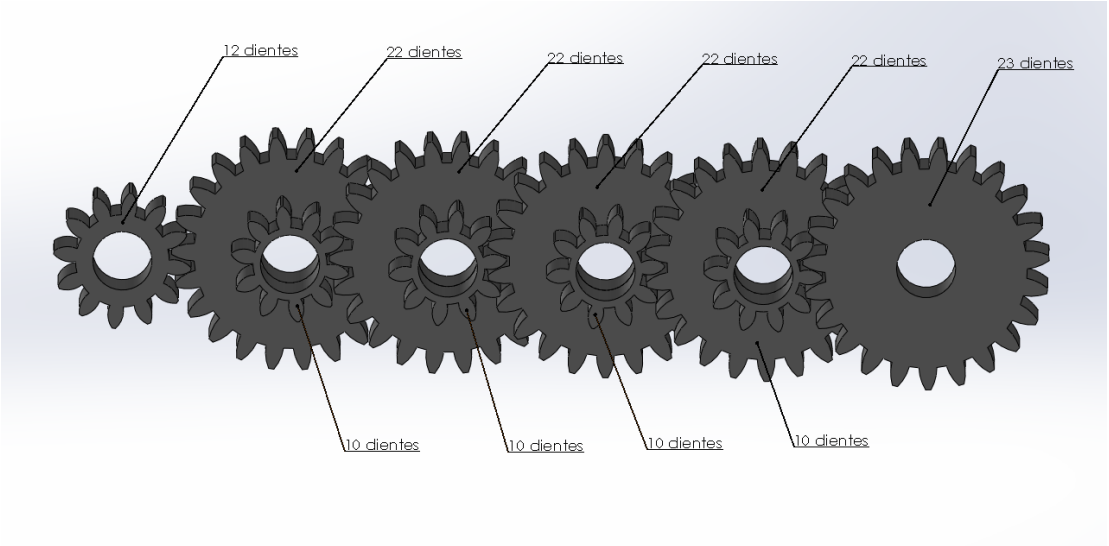


Figura 29. Disposición de los engranajes de la caja reductora

Una contado el número de dientes de cada engranaje y sabiendo su disposición dentro de la caja de engranajes pude calcular fácilmente su relación de transformación con la ecuación (5).

$$R_t = \frac{Nº\ dientes\ conducidos}{Nº\ de\ dientes\ conductores} = \frac{22 * 22 * 22 * 22 * 23}{12 * 10 * 10 * 10 * 10} = 44{,}899 \cong 45 \qquad (5)$$

4.3 Rueda. Cálculo del coeficiente de rozamiento

El coeficiente de rozamiento B_R que aparece en las ecuaciones (6) y (7), hace referencia al rozamiento de las ruedas del robot.

$$M_1 * \frac{dV_{X1}}{dt} = B_R * V_{X1} - F_{X2} \quad (6)$$

$$\Gamma_1 = \frac{1}{R_1} * (I_R * \frac{dW_1}{dt} + B_R * W_1) \quad (7)$$

Este coeficiente de rozamiento puede variar dependiendo del tipo de material del que este hecho la rueda y en función del suelo donde ruede, según esto la rueda tiene diferentes tipos de rozamientos los cuales vamos a ver a continuación.

4.3.1 Rozamiento por un fluido

Rozamiento por fluidos: Es el que se produce cuando hay un movimiento de un cuerpo a través de un fluido (líquido o gas). Este tipo de rozamiento depende de la forma del cuerpo, de la viscosidad del fluido y de la velocidad del cuerpo. Se puede calcular mediante la ecuación (8).

$$F_r = \frac{1}{2} * \rho * V^2 * C_d * A \quad (8)$$

Como la velocidad y el área del cuerpo van a ser muy pequeñas este rozamiento se pueden considerar directamente despreciables.

4.3.2 Rozamiento por deslizamiento

Es el que se produce cuando hay un movimiento relativo entre dos superficies que se deslizan una sobre el otra. Este tipo de rozamiento depende de la naturaleza de los materiales que están en contacto y de la fuerza normal que ejerce una superficie sobre la otra. Se puede calcular mediante la ecuación (9).

$$F_r = \mu * N \quad (9)$$

Donde F_r es la fuerza de rozamiento al deslizamiento, μ es el coeficiente de rozamiento al deslizamiento y N es la fuerza normal.

El valor del coeficiente de rozamiento al deslizamiento es característico de cada par de materiales en contacto; no es una propiedad intrínseca de un material. Depende además de muchos factores como la temperatura, el acabado de las superficies, la velocidad relativa entre las superficies, etc. La naturaleza de este tipo de fuerza está ligada a las interacciones de las partículas microscópicas de las dos superficies implicadas. Por ejemplo, hielo sobre una lámina de acero pulido tiene un coeficiente bajo; mientras que el caucho sobre el

pavimento tiene un coeficiente alto. El coeficiente de fricción puede tomar valores desde casi cero y normalmente no sobrepasa la unidad.

Para poder despreciar el rozamiento por deslizamiento, se tiene que eliminar el deslizamiento entre la rueda y el suelo. Para ello en la ecuación (10), V debe ser igual a 0, donde V es la velocidad lineal del centro de masa de la rueda, ω es la velocidad angular de la rueda y R es el radio de la rueda. Esta condición implica que la velocidad lineal del punto de contacto entre la rueda y el suelo sea cero, esto significa que el punto donde la rueda toca el suelo no se mueve respecto al suelo, sino que permanece fijo en ese lugar o lo que es lo mismo que no hay deslizamiento.

$$V = \omega * R \qquad (10)$$

Para poder eliminar el deslizamiento he usado un material con baja dureza y un diseño de la cubierta a rayas con el objetivo de tener máxima adherencia con el suelo, y así no deslizar con la misma. La llanta de la rueda esta echa de poliácido láctico (PLA), modelo de PLA 3D870 de la marca Smartfil, y la cubierta poliuretano termoplástico (TPU), modelo de TPU FLEX 93A también de la marca Smartfil (Figura 30. Diseño Rueda PLA-TPU, la mayor diferencia entre estos materiales y que más me interesa, para el proyecto es su dureza, según sus hojas de características y conforme a la ISO 7619-1 el PLA tiene un Shore D 86 y el TPU un Shore A 93. La dureza Shore se calcula mediante un ensaño de dureza que se hace a los materiales y mide la resistencia a la indentación. La identación mide la resistencia a la deformación cuando se aplica una fuerza de compresión muy concentrada y conocida a través de un identador, un instrumento de medida que tiene una forma muy concreta, su punta puede ser de tipo cono, pirámide… Tras aplicar una fuerza con un identador se mide la superficie o área que deja la huella del aparato en el material. Con la profundidad del área de la huella y sabiendo la fuerza aplicada se puede sacar el shore.

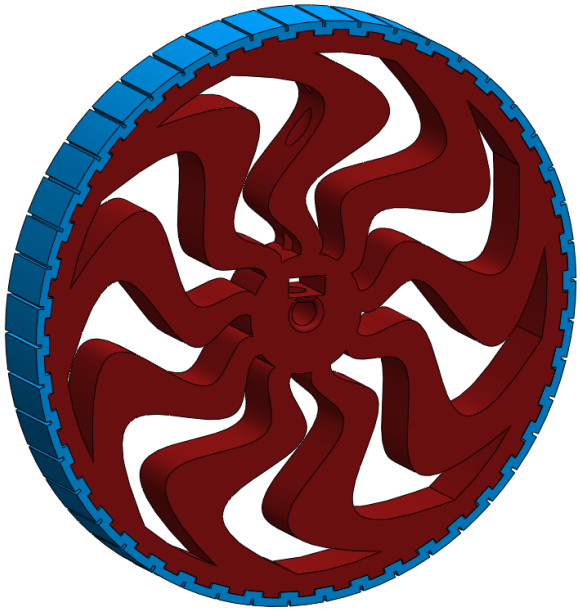


Figura 30. Diseño Rueda PLA-TPU

Existen varias escalas de dureza shore según el tipo de material, en mi caso tengo la escala A para el TPU y la escala D para el PLA. La escala A se usa para materiales blandos y flexibles y la escala D para materiales más duros y que resisten más a la flexión. En la Tabla 3 y en la Figura 31. Relación Shore sacados de la página de la empresa 3DALIA, empresa de desarrollo de productos, se puede ver una comparativa entre las diferentes escalas de Shore, se puede ver como la escala A correspondiente al TPU tiene una dureza de A 93, en cambio mi PLA tiene una dureza de D 86. La escala A va de 5 a 100 en el caso de la escala D un valor de 58 correspondería a un valor de 100 su máximo con lo que el PLA sería mucho más duro que TPU, saliéndose por mucho de la escala de medida del TPU.

He usado el TPU para la cubierta de la rueda ya que gracias a su baja dureza puedo conseguir que la rueda no deslice y así evitar el rozamiento por deslizamiento, el resto de rueda, la llanta, la parte más grande, he usado PLA ya que al crear las piezas de forma aditiva es más fácil de imprimir. El TPU por la propia propiedad que andamos buscando, la baja dureza, hace que a la hora de ser inyectada por el extrusor pueda crear atascos por su gran fricción con la boquilla y el barrel del extrusor.

Tabla 3. Características de materiales Shore

Shore 00 (ASTM D 2240)	Shore 0 (ASTM D 2240)	Shore A (DIN 53505)	Shore D (DIN 53505)	IRHD (ASTM 1415)
		100	58	100
		95	46	95
		90	39	90
		85	33	85
98	84	80	29	80
97	79	75	25	74
95	75	70	22	68
94	72	65	19	64
93	69	60	16	62
91	65	55	14	54
90	61	50	12	49
88	57	45	10	44
86	53	40	8	39
83	48	35	7	35
80	42	30	6	28
76	35	25		
70	28	20		
62	21	15		
55	14	10		
45	8	5		
Elastómeros celulares de media y baja densidad	Elastómeros compactos y celulares de media densidad	Elastómeros compactos y celulares	Poliuretanos, PTFE rígido, Termoplásticos y Elastómeros muy duros	Elastómeros compactos

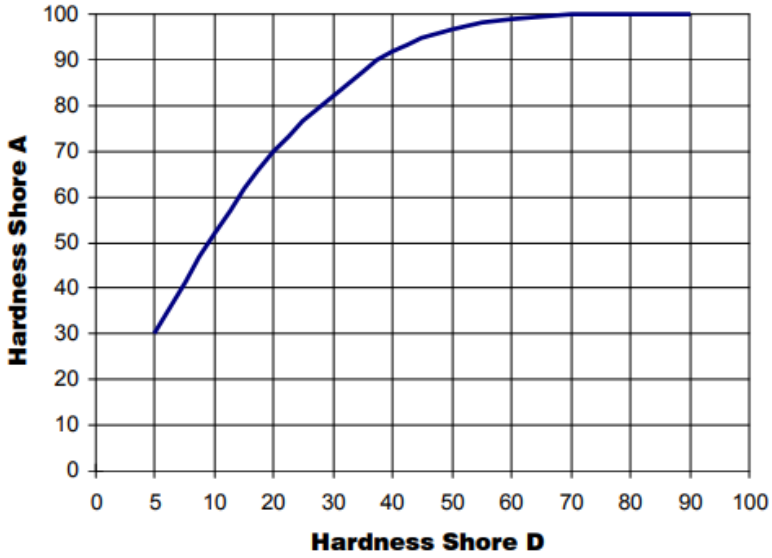


Figura 31. Relación Shore

4.3.3 Coeficiente de rozamiento por rodadura

Es el que mide la fuerza que se opone al movimiento de un cuerpo que rueda sobre una superficie. Depende principalmente de la deformación que sufren el cuerpo y la superficie al rodar uno sobre el otro. Esta deformación provoca una pérdida de energía por calor y una resistencia al movimiento. El coeficiente de rozamiento por rodadura se puede calcular mediante la ecuación (11).

$$\mu_r = \frac{F_r}{N * R} \qquad (11)$$

Donde μ_r es el coeficiente de rozamiento por rodadura, F_r es la fuerza de rozamiento por rodadura, N es la fuerza normal y R es el radio del cuerpo que rueda.

Como se puede observar, el coeficiente de rozamiento por rodadura depende también del radio del cuerpo que rueda, lo que implica que cuanto mayor sea el radio, menor será el coeficiente. Esto se debe a que un cuerpo con mayor radio sufre menos deformación al rodar sobre una superficie.

Para estimar el valor del coeficiente de rozamiento por rodadura, se pueden usar algunos valores típicos que se han obtenido experimentalmente para diferentes pares de materiales en contacto. Por ejemplo, para una rueda de caucho sobre asfalto seco, el coeficiente de rozamiento por rodadura es aproximadamente 0.01. Para una rueda de acero sobre acero, el coeficiente de rozamiento por rodadura es aproximadamente 0.001. Estos valores pueden variar según las condiciones del experimento, como la temperatura, la humedad, el estado de limpieza o el desgaste de la rueda y del suelo. En mi caso como la rueda gira y no se desliza este rozamiento lo despreciaremos.

5 DISEÑO DEL CHASIS

Una parte importante del proyecto es el diseño del chasis, a lo largo del proyecto me di cuenta de la importancia que tenía hacer un buen diseño para nuestro robot. He realizado varios diseños debidos a problemas técnicos relacionados con el posicionamiento y control del movimiento del robot, en cada rediseño el objetico principal ha sido la modificación del centro de masas y centro de gravedad del robot.

El centro de masas (C.M) es el promedio de la posición de todas las partículas de masa que forman el cuerpo, al aplicar una fuerza no equilibrada en este punto, el cuerpo realizara un movimiento de translación no de rotación. En la Figura 32 se pueden ver dos masas, M1 y M2, la masa de M2 es mayor que la de M1, estás masas están unidas por una barra sin peso y no deformable, podemos ver un punto que es el centro de masas, C.M, la distancia X2 es menor que X1 ya que la masa M2 es mayor que M1 esto hace que el centro de masas este mas cerca de M2, en el modelo de la izquierda de la figura se ve como se aplica una fuerza F sobre la barra esta fuerza es aplicada en un punto el cual este más cercano a la masa más pequeña esto provoca que el sistema gire en el sentido de las manecillas del reloj, en el modelo del centro de la figura se ve como se aplica una fuerza F sobre la barra esta fuerza es aplicada en un punto el cual este más cercano a la masa más grande esto provoca que el sistema gire en sentido opuesto al de las manecillas del reloj, por último ,en el modelo de la derecha de la figura se ve como se aplica una fuerza F sobre la barra esta fuerza es aplicada en el centro de masas C.M. del sistema esto hace que el sistema no tenga rotación y que su movimiento simplemente sea de traslación.

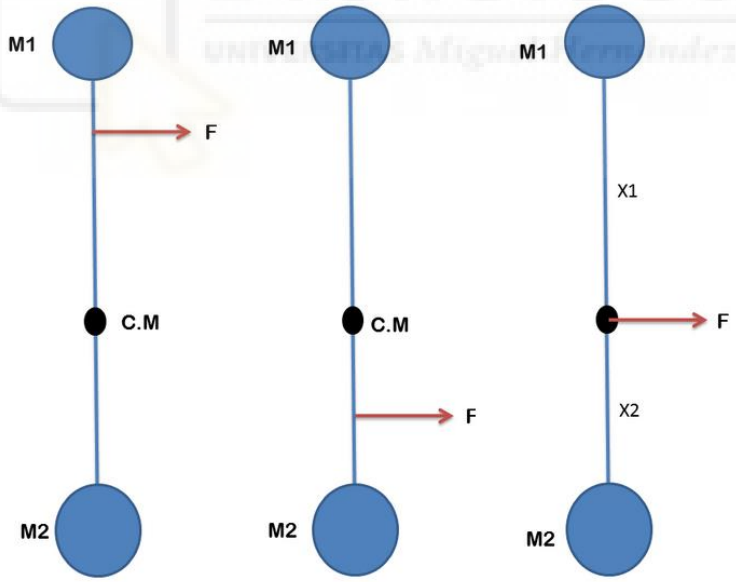


Figura 32. Movimiento de un cuerpo respecto al centro de masas

No debemos confundir el centro de masas con el centro de gravedad, el centro de gravedad es el punto imaginario de aplicación de la resultante de toda la gravedad que actúa sobre las distintas porciones materiales de un cuerpo, de tal forma que el momento de fuerza respecto a cualquier punto de esta resultante aplicada en el centro de gravedad es el mismo que el producido por los pesos de todas las masas materiales que constituyen dicho cuerpo. En un sistema sin gravedad no tiene sentido hablar de centra de gravedad, pero si podemos hablar de un centro de masas.

Es importante mencionar que el centro de masa coincide con el centro de gravedad cuando el cuerpo está en un campo gravitatorio uniforme. Esto significa que cuando el campo gravitatorio es de magnitud y dirección constante en toda la extensión del cuerpo, podemos considerar que el centro de masa y el centro de gravedad son el mismo punto.

Los diseños han sido realizados con el Software Solidworks en el cual diseñé cada parte del robot por separado, mediante la función ensamblaje, que tiene incorporado Solidworks pude relacionar lar diferentes partes del robot, mediante relaciones físicas y asi poder ver como interactúan entre sí. Aparte de estas relaciones indique las características físicas que tienen los diferentes materiales de los cuales se compone el robot como la densidad, dureza o rugosidad entre otros, esto me fue de gran utilidad para saber el centro de masas del ensamblaje en conjunto y usar sus relaciones para simular en Matlab.

5.1 Diseño 1

La primera versión del chasis del robot se puede ver en la Figura 33, al imprimir y probar esta versión me di cuenta de que el centro de masas no estaba en la perpendicular del eje del motor, respecto al suelo. Esto provoca que cuando el robot conseguía la perpendicularidad, el objetico del robot, este no conservaba el equilibrio y tenía la tendencia a caerse en dirección a la batera.

Este centro de masas y centro de gravedad se puede calcular mediante Solidworks, si ponemos propiedades físicas a los materiales que compone el robot, en la Figura 33 se puede ver un punto en blanco y negro el cual indica el centro de masas y también el centro de gravedad, como podemos ver cuando el robot mantiene la vertical este centro de gravedad no coincide con dicha vertical esto provocará que la gravedad tire de nuestro robot en dirección a la batería, nuestro punto de equilibrio, que es cuando la centro de gravedad está en la vertical con el horizonte no coindiza con la vertical de nuestro robot.

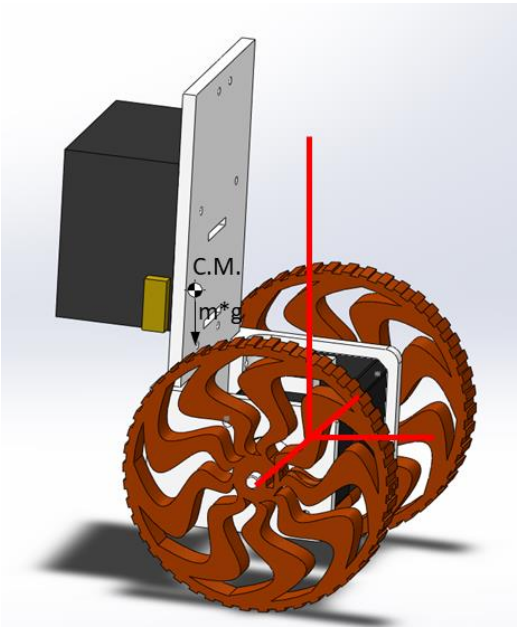


Figura 33. Diseño del Chasis. Diseño 1

En este primer diseño me di cuenta de la importancia que tenía tener una buena distribución de las masas en el robot si quería que la vertical con el horizonte coincidiera con la vertical de mi robot el centro de gravedad y el centro de masas debería coincidir con la línea que une la vertical con los ejes del motor, esto solo lo podía conseguir con un buen diseño en el robot.

5.2 Diseño 2

Con el segundo diseño, Figura 34, solucione el problema de que la recta que une el centro de masas con los ejes del motor no coincidiera con la vertical respecto al horizonte cuando el robot estaba en estado de equilibrio. Esto lo solucione haciendo una mejor distribución de los pesos, en la Figura 34 podemos ver como la batería, la parte negra cuadrada, la cual es la parte más pesada del robot en vez de ir en la parte trasera del robot ahora se ha colocado en la parte de arriba de este, haciendo coincidir con una recta el centro de masas de la batería con la línea imaginaria que une los ejes y el nuevo centro de masas, también se mejora la posición de la placa controladora para tener una mejor distribución de los pesos.

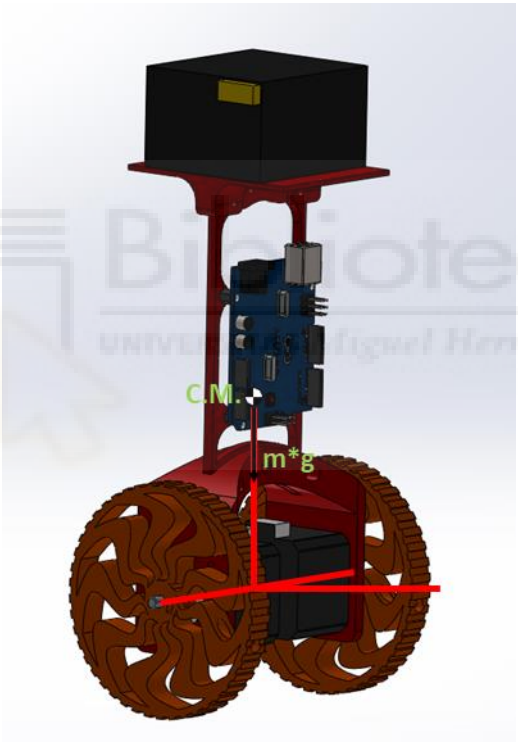


Figura 34. Diseño del Chasis Diseño 2

Una vez solucionado este problema me puse a simular en Matlab con este diseño, consigue controlar la posición de nuestro robot, pero era un sistema, bastante inestable y en cuánto el robot se separaba del punto de equilibrio tenía que corregir rápidamente la posición ya que si no llegaba a unos ciertos ángulos en los que ya no era capaz de volver al punto de equilibrio. Nuevamente este problema se puedo solucionar haciendo un nuevo diseño de chasis.

5.3 Diseño 3

Vamos a hacer un símil de nuestro robot con la Figura 35, en el que tenemos una barra sin peso que rota en su mitad sobre un apoyo que tiene la mesa azul, este apoyo donde rota la barra será

el eje del motor, esta barra tiene dos pesos una bola negra muy pesado y uno rosa poco pesada estos pesos representaran las diferentes partes del robot y sus pesos, estos pesos los podremos mover libremente a lo largo de la barra, para este ejemplo.

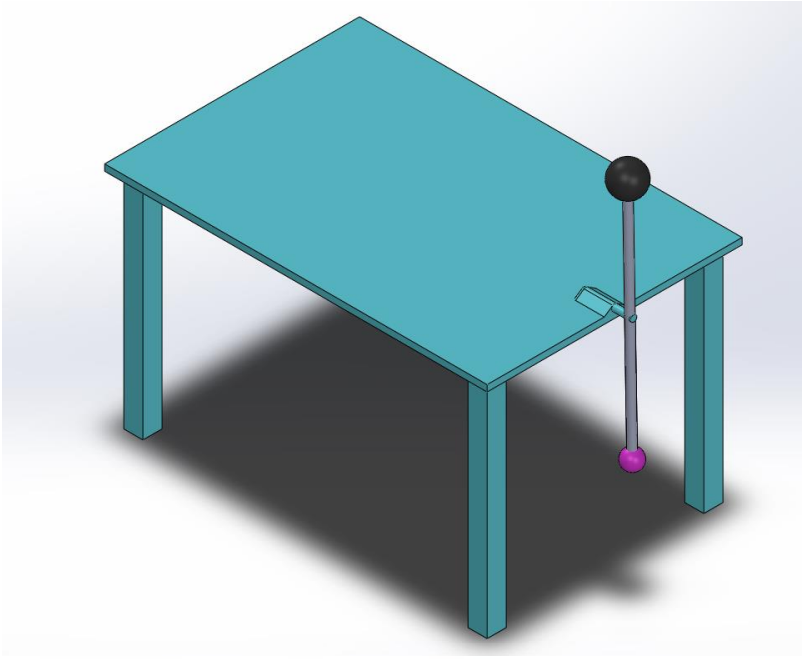


Figura 35. Simulación barras-pesos

En la Figura 36 se puede ver donde se sitúa el centro de masas C.M. Cuando tenemos el peso mas grande arriba del todo, imagen de la izquierda de la Figura 36, el sistema está en equilibrio ya que la dirección de la fuerza de la gravedad del cuerpo pasa por el punto de apoyo. En cuanto se produzca una pequeña perturbación la dirección de esta fuerza gravitaría ya no pasara por el apoyo y esta misma fuerza hará que la barra empiece a girar sobre el apoyo de la mesa, imagen de la derecha de la Figura 36.

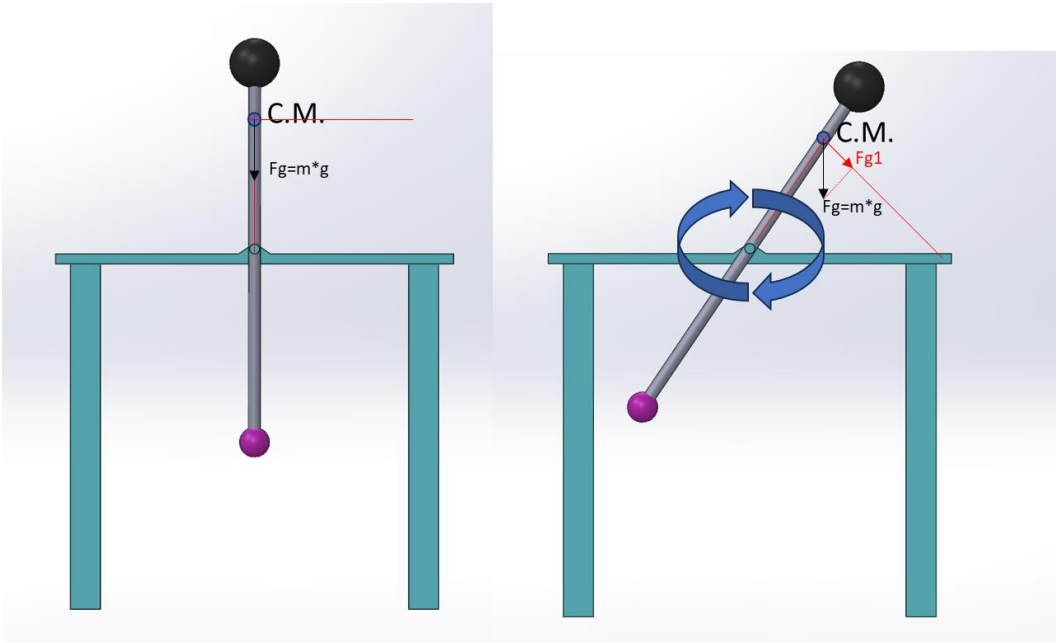


Figura 36. Pesos en los extremos

Pero qué pasa si ahora acercamos la masa mas pesada al punto de apoyo, imagen de la izquierda de la Figura 37, si hacemos esto su momento de inercia se reducirá.

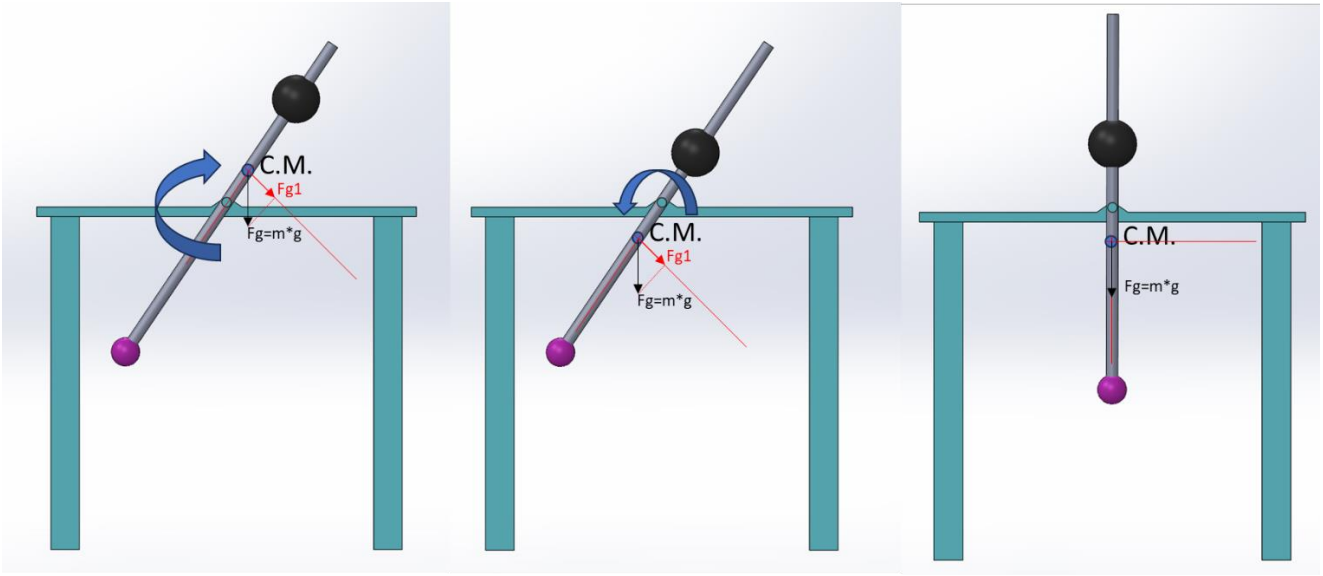


Figura 37. Movimiento del peso

El momento de inercia es la medida de la resistencia de un objeto a cambiar su estado de movimiento, su fórmula es la (12), siendo la m la masa del conjunto barra-pesos y la r la distancia del centro de masas al apoyo.

$$I = m * r^2 \qquad (12)$$

Cuando el peso se mueve más cerca del punto de apoyo, el centro de masa del sistema se acerca al punto de apoyo. Esto reduce el momento de inercia del sistema, lo que significa que el sistema es menos resistente al cambio en su estado de movimiento. Lo que nos facilita volver a la posición vertical el chasis del robot, si además como es el caso de mi robot tiene unas cubiertas en las rueda lo suficientemente gomosa para que la rueda no tenga un punto de apoyo con el suelo si no que la rueda se deforma aumentando la superficie de contacto con el suelo, esto provoca una resistencia al cambio de ángulo en ella. Esto nos permitirá que el punto de equilibrio ya no sea solo la vertical, si no que el par producido por la gravedad en el cuerpo debe ser lo suficientemente grande para vencer a el par que produce tener varios puntos de apoyo.

Al bajar el C.M. el par producido por el peso del robot es menor. Si pudiéramos bajar tanto el centro de masas como para que este se quedara por debajo del punto de apoyo, imagen del centro de la Figura 37, la propia fuerza de la gravedad haría que nuestro sistema se quedara en perpendicular con el horizonte, imagen de la derecha de la Figura 37.

Con la idea de bajar al máximo el centro de masas hicimos el nuevo diseño de nuestro chasis (Figura 38), para este diseño tuve que pensar en la parte mas pesada de mi robot y como podía reubicar.

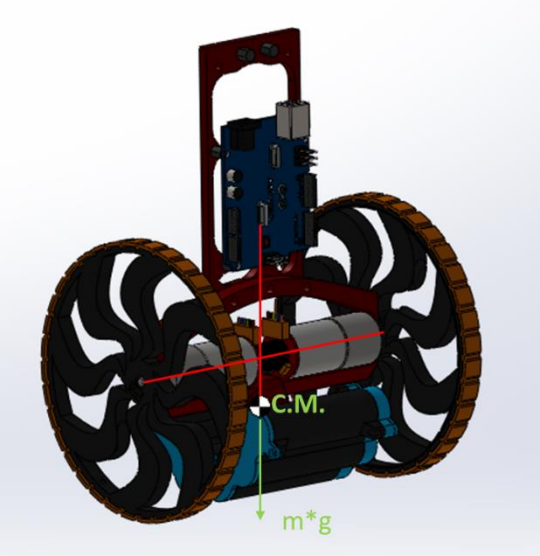


Figura 38. Diseño del chasis Diseño 3

Una de las partes más pesadas del robot es la batería, teniendo la posibilidad de realizar mi propia batería, diseñe y fabrique mi propia batería, Figura 39, para colocarla por debajo de los ejes del motor y con ello conseguir bajar el centro de masas.

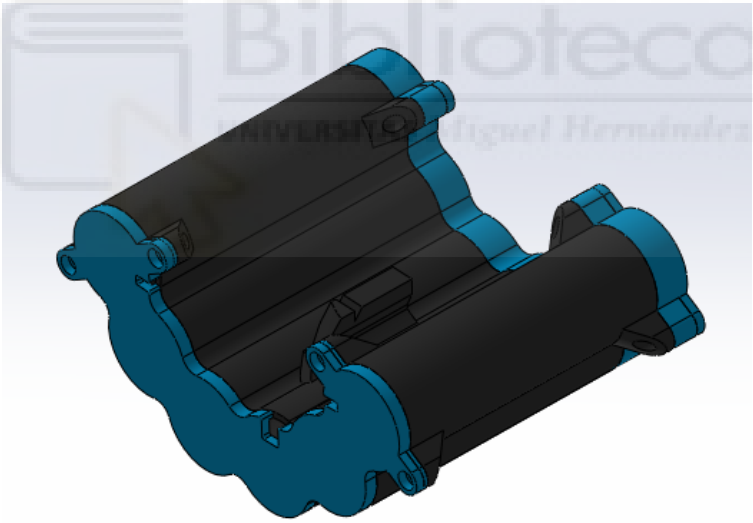


Figura 39. Batería Diseño 3

Conseguí bajar el centro de masas por debajo del eje de los motores y con ello conseguir que el propio peso del robot volviera a colocar al robot en su punto de equilibrio, la vertical con el horizonte.

5.4 Diseño 4

El diseño 3 carece de una cosa y que necesita nuestro robot para poder considerarse un AMR, no cuenta con sensores para recoger información de su entorno. Estos sensores por función y disposición van por encimo de los ejes de los motores, por ejemplo, el sensor lidar 360 grados necesita si o si, si quiero mapear los 360 alrededor del robot, no tener ninguna parte del robot obstaculizando su laser esto solo nos deja la opción de colocar este sensor en la parte de arriba del robot. Esto provoca que el centro de masa del robot vuelva a estar por encima de los ejes del robot, haciéndolo inestable, pero el hecho de haber colocado la batería por debajo de los ejes en el diseño 3 mejora y hace más sencillo el control del robot para que el robot consiga estar en su punto de equilibrio (Figura 40). En la Figura 41 se puede ver una foto real del AMR una vez montado.

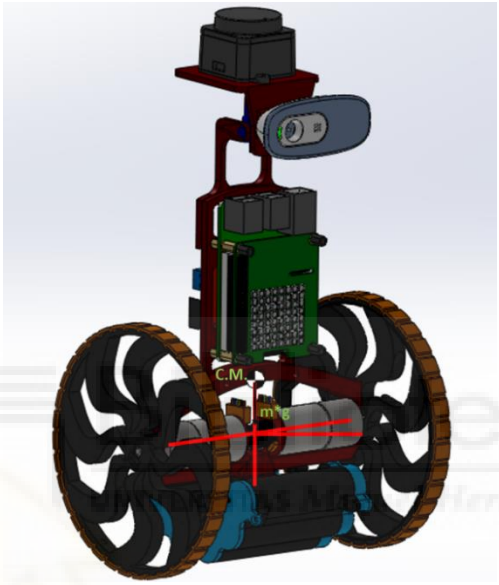


Figura 40. Diseño del chasis Diseño 4

Como podemos ver en la Figura 40 si comparamos el diseño 4 con los diseños 1 y 2 que corresponden a la Figura 33 y Figura 34 respectivamente, el diseño 4 es el que tiene el centro de más cerca de los ejes de los motores, que es el punto sobre el que rota el cuerpo del robot, esto hace que su momento de inercia, ecuación (12), sea el más bajo de todos, esto me ayudara cuando quiera controlar su posición.

Figura 41. Foto diseño 4

6 SIMULACIÓN

Como hemos visto en el punto 5 mi robot es un robot que debido a que tiene su centro de masas por encima de sus ejes si no se hace ninguna acción sobre él, el robot se vencerá hacia delante o atrás, ante cualquier perturbación. Haremos un símil de nuestro robot y la Figura 42, está figura está formada por un carro verde que se desliza sobre un rail azul, que está sujeto a la mesa, este carro tiene apoyada una barra que gira sobre él, la barra tiene dos pesos en sus entremos uno negro muy pesado y una rosa más ligera, los dos pesos están a la misma distancia del carro.

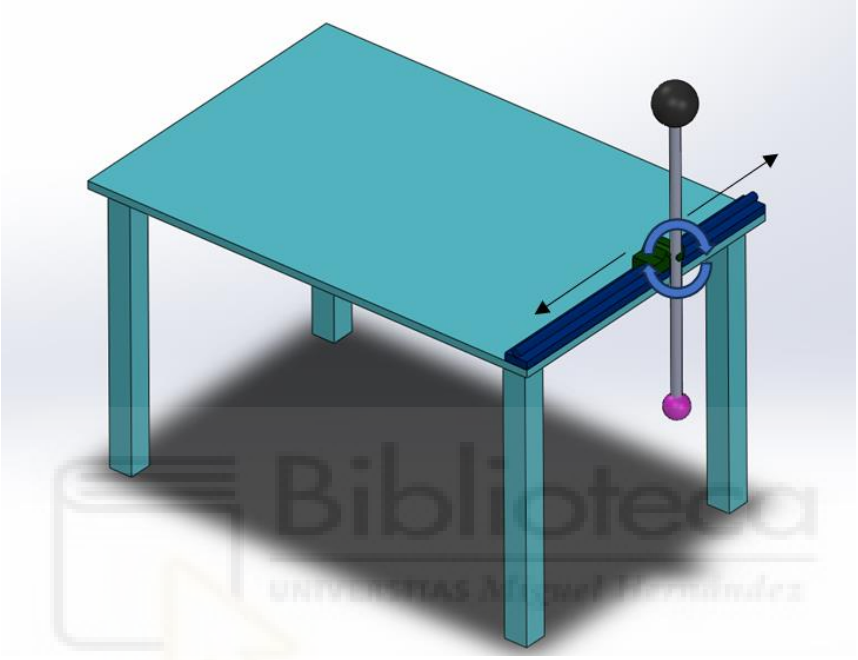


Figura 42. Simulación carro-barra-pesos

En esta figura tenemos el centro de masa, barra-pesas, por encima del eje donde rota, ante cualquier perturbación la pesa negra caerá hacia un lado u otro, pero que pasa si justo cuando va a caer movemos el carro verde en la dirección a la que cae la pesa, si lo hacemos lo suficientemente rápido podemos hacer que la pesa negra vuelva a estar en equilibrio (Figura 43). En nuestro robot sucede lo mismo, si movemos el cuerpo del robot, mediante los motores, en la dirección en la que está cayendo con una aceleración, velocidad y tiempo de respuesta correcta, el robot volverá a estar en equilibrio.

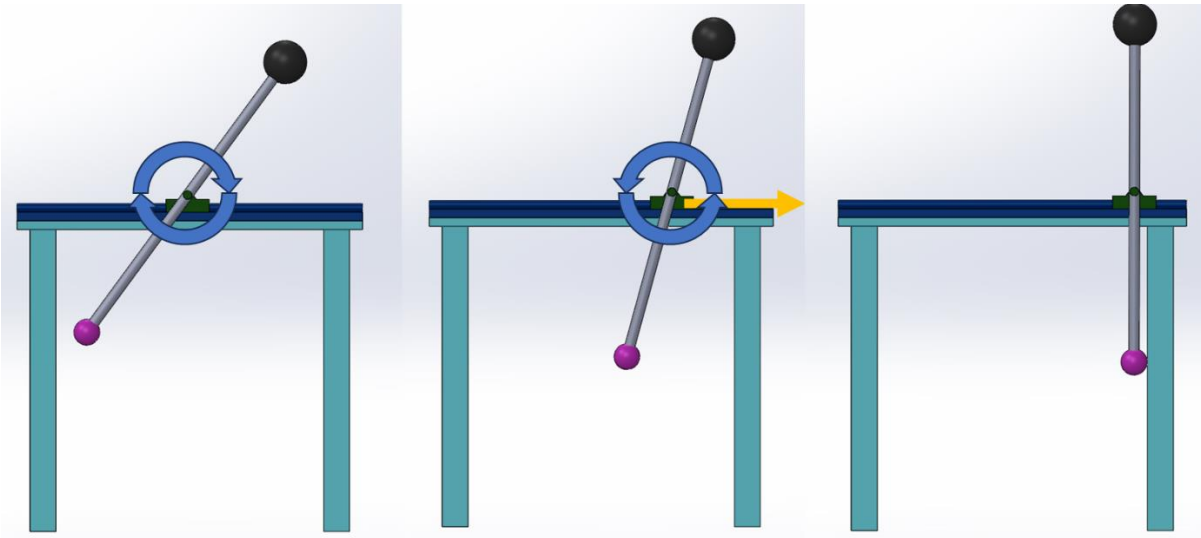


Figura 43. Movimiento carro-barra-pesos

Las variables por controlar para conseguir esto objetivo son, el tiempo que tiene que estar conectados los motores y la tensión que se le aplica para conseguir que el robot vuelva a estar en equilibrio. Saber el valor de estas variables no es una tarea sencilla, para ello necesito sensores, para conocer el ángulo al que esta el robot y la posición de los motores. También necesitamos algún tipo de regulador o controlador que en función de las señales de los sensores actuar sobre los motores de la forma correcta, para ello he implemento un regulador PID. Para conocer el tipo de PID y los valores de este, he realizado varias simulaciones del sistema mediante Matlab y SolidWorks.

6.1 GUIDE de Matlab, modelado del sistema

GUIDE (entorno de desarrollo de GUI) es una herramienta de MATLAB que proporciona un entorno de programación visual para diseñar interfaces de usuario gráficas (GUIs). He usado esta herramienta para poder calcular los valores PID de nuestro controlador y ver la respuesta del sistema ante diferentes entradas. Para llegar hacer esto primero debía tener un sistema el cual controlar, mediante dicho PID, y para conseguirlo modele mi robot, mediante ecuaciones que representaran el funcionamiento de este.

Lo primero que modele son las ecuaciones de la parte eléctrica de los motores, en la Figura 44 podemos ver un esquema eléctrico de las partes de un motor, en la que:

Rm: Representa la resistencia del motor. Esta resistencia es debido al alambre de cobre utilizado en el embobinado del motor.

Lm: Indica la inductancia del motor. La inductancia es una propiedad de los circuitos eléctricos que se opone a los cambios en la corriente eléctrica.

Vrm y Vlm: Son las caídas de voltaje a través de la resistencia y la inductancia, respectivamente. Estas caídas de voltaje son proporcionales a la corriente que fluye a través de la resistencia y la inductancia.

Vt: Es el voltaje total aplicado al circuito del motor. Este es el voltaje de entrada que alimenta el motor.

i: Corresponde a la corriente que fluye a través del circuito. Esta corriente es la que genera el campo magnético que hace girar el motor.

e: Es la fuerza electromotriz inducida en el motor. Esta es una tensión que se genera debido al movimiento del motor en el campo magnético y se opone al voltaje de entrada.

Tm: Representa el torque desarrollado por el motor. Este es el par de fuerzas que hace girar el motor.

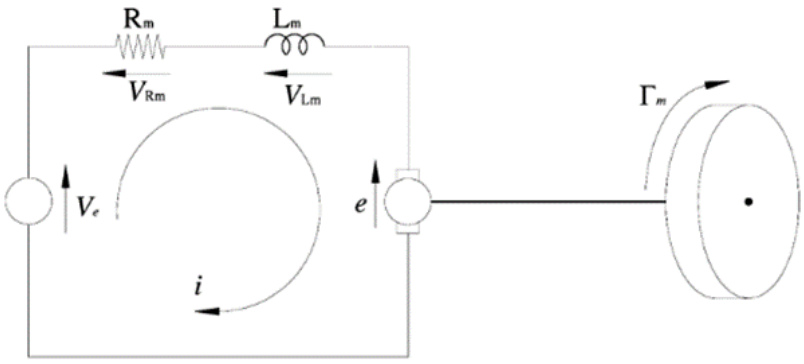


Figura 44. Esquema de un motor DC

Las ecuaciones que modelan el subsistema motor de corriente continua, parte eléctrica, son:

$$V_e = R_m * i + L_m * \left(\frac{di}{dt}\right) + K_V * W_1 \quad (13)$$

$$\Gamma_m = K_i * i \quad (14)$$

En la Figura 45 se ven las variables que entran en juego cuando queremos mover nuestro robot, las cuales son:

θ_1, θ_2: Son los ángulos de inclinación del robot y de las ruedas, respectivamente.

W_1, W_2: Representan las velocidades angulares del robot y de las ruedas, respectivamente.

V_{x1}, V_{x2}, V_{y2}: Son las velocidades lineales del robot y de las ruedas en las direcciones x e y.

X_1, X_2, Y_2: Representan las posiciones del robot y de las ruedas en las direcciones x e y.

F_{x2}, F_{y2}: Son las fuerzas aplicadas a las ruedas en las direcciones x e y.

F_r: Es la fuerza de resistencia que se opone al movimiento del robot.

M_2: Es la masa del robot.

g: Es la aceleración debido a la gravedad.

B_R: Es el coeficiente de resistencia.

R: Es el radio de las ruedas.

L: Distancia del eje de los motores al centro de masas

Tm: Representa el torque desarrollado por el motor. Este es el par de fuerzas que hace girar el motor.

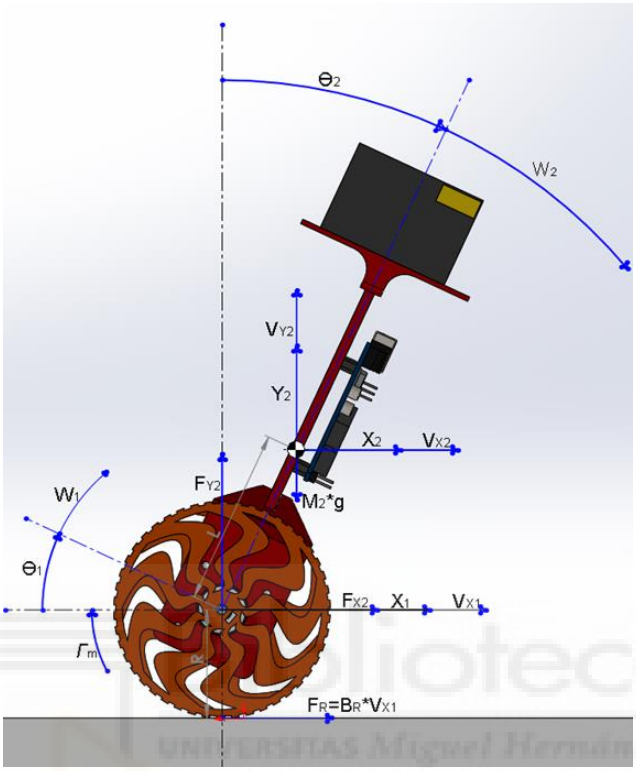


Figura 45. Variables del modelo

Las ecuaciones que modelan el movimiento la de la rueda, la suponemos sin deslizamiento, son:

$$M_1 * \left(\frac{dV_{X1}}{dt}\right) = B_R * V_{X1} - F_{X2} \quad (15)$$

$$V_{X1} = \frac{dX_1}{dt} \quad (16)$$

$$\Gamma_m = \frac{1}{R_1} * \left(I_R * \frac{dW_1}{dt} + B_R * W_1\right) \quad (17)$$

$$W_1 = \frac{d\theta_1}{dt} \quad (18)$$

Las ecuaciones que modelan el movimiento del cuerpo son:

$$F_{X2} = M_2 * \frac{dV_{X2}}{dt} \quad (19)$$

$$V_{X2} = \frac{dX_2}{dt} \quad (20)$$

$$V_{Y2} = \frac{dY_2}{dt} \quad (21)$$

$$I_C * \frac{dW_2}{dt} = L * (-F_{X2} * \cos(\theta_2) + F_{Y2} * \sin(\theta_2)) \quad (22)$$

$$W_2 = \frac{d\theta_2}{dt} \quad (23)$$

En la Figura 45 tenemos ligaduras entre variables debido a la unión mecánica entre elementos, estas ecuaciones relacionan el movimiento absoluto de algunas partes del robot cuando se mueven otras partes de este, debidas a la unión física de partes del robot, en nuestro caso son los ejes de los motores, las ecuaciones que relacionan estas variables son:

$$X_2 = X_1 + L * \sin(\theta_2) \quad (24)$$

$$Y_2 = L * \cos(\theta_2) \quad (25)$$

$$V_{X2} = V_{X1} + L * \cos(\theta_2) * W_2 \quad (26)$$

$$V_{Y2} = -L * \sin(\theta_2) * W_2 \quad (27)$$

$$\frac{dV_{X2}}{dt} = \frac{dV_{X1}}{dt} - L * \sin(\theta_2) * (W_2)^2 + L * \cos(\theta_2) * \frac{dW_2}{dt} \quad (28)$$

$$\frac{dV_{Y2}}{dt} = -L * \cos(\theta_2) * (W_2)^2 - L * \sin(\theta_2) * \frac{dW_2}{dt} \quad (29)$$

$$X_1 = R * \theta_1 \quad (30)$$

$$V_{X1} = R * W_1 \quad (31)$$

$$\frac{dV_{X1}}{dt} = R * \frac{dW_1}{dt} \quad (32)$$

Una vez que tenemos las ecuaciones que modelan el comportamiento de nuestro robot necesitamos una forma de relacionarlas entre las salidas y entradas, de tal forma que sea sencillo trabajar con ellas, para esto usaremos la representación interna o representación de estados, las ecuaciones (33) y (34) son las ecuaciones de la representación interna en la (33) es la ecuación de estado y la (34) es la ecuación de salida, las matrices A, B, C y D son constantes que define el comportamiento de nuestro robot, "ẋ" es la matriz de las variables de estado derivadas, "x" la matriz de las variables de estado, "u" la matriz de las variables de entrada y "y" la matriz de las variables de salida. La representación interna es útil porque simplifica el análisis y el diseño del controlador para el sistema

$$\dot{x} = A * x + B * u \quad (33)$$

$$y = C * x + D * u \quad (34)$$

La representación interna de nuestro robot es la mostrada en las ecuaciones (35) y (36) siendo cada matriz de la ecuación de estado (35) y la ecuación de salida (36) las explicadas en correspondiente ecuaciones (33) y (34). Para llegar a estas matrices a partir de las ecuaciones del sistema, necesitamos que nuestro sistema sea lineal e invariante en el tiempo (LTI) para ello previamente se han linealizado las ecuaciones del sistema respecto a un punto de equilibrio, en nuestro caso los 90º o posición vertical del robot.

$$\begin{bmatrix} \frac{d}{dt}\Delta\Theta_1 \\ \frac{d}{dt}\Delta W_1 \\ \frac{d}{dt}\Delta\Theta_2 \\ \frac{d}{dt}\Delta W_2 \\ \frac{d}{dt}\Delta i \end{bmatrix} := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{-B_R}{I_R} & 0 & 0 & \frac{K_i}{I_R} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{B_R \cdot R}{L \cdot M_2} + \frac{M_1 \cdot R \cdot B_R + B_R}{L \cdot M_2 \cdot I_R} & 0 & 0 & \frac{-M_1 \cdot R \cdot K_i - R \cdot K_i}{L \cdot M_2 \cdot I_R} \\ 0 & \frac{-K_V}{L_m} & 0 & 0 & \frac{-R_m}{L_m} \end{bmatrix} \cdot \begin{bmatrix} \Delta\Theta_1 \\ \Delta W_1 \\ \Delta\Theta_2 \\ \Delta W_2 \\ \Delta i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{L_m} \end{bmatrix} \cdot \Delta V_e \qquad (35)$$

$$[\Delta\Theta_2] := [0\ 0\ 1\ 0\ 0] \cdot \begin{bmatrix} \Delta\Theta_1 \\ \Delta W_1 \\ \Delta\Theta_2 \\ \Delta W_2 \\ \Delta i \end{bmatrix} + [0] \cdot \Delta V_e \qquad (36)$$

Muchas constantes de las matrices de las ecuaciones (35) y (36) han sido calculadas previamente en el punto 4, estas son la relación par intensidad K_i punto 4.1, la constante de velocidad del motor K_v punto 4.2 y el coeficiente de rozamiento de la rueda B_R punto 4.3. Otras como la inductancia L_m y resistencia del motor R_m se midieron directamente con un LCR. Las demás se obtuvieron mediante SolidWorks.

Para sacar el momento de inercia de la rueda I_R, pese la pieza una vez impresa y saque su volumen con SolidWorkds, con eso obtuve la densidad de la rueda la cual introduje en las propiedades de la rueda en SolidWorks. Una vez echo pude sacar el momento de inercia en el eje que va a girar, en nuestro caso y como se muestra en la Figura 46, es el eje Lzz.

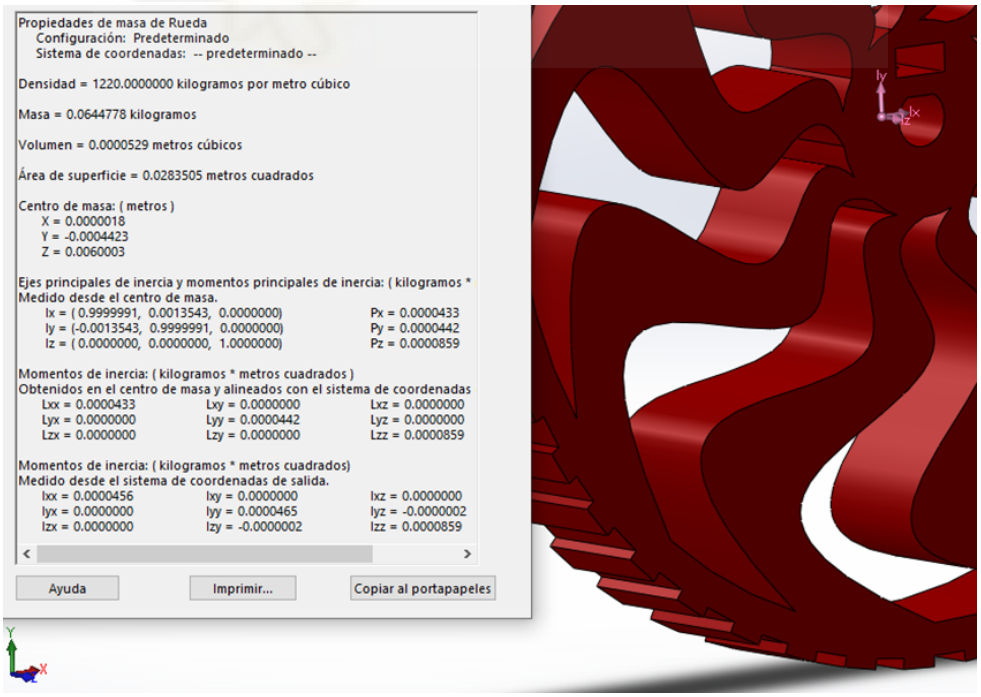


Figura 46. Momento de Inercia de la rueda SolidWorks

I_R=0.0000859Kg*m

Momento de inercia del cuerpo I_c, procederemos igual que con la rueda (Figura 47).

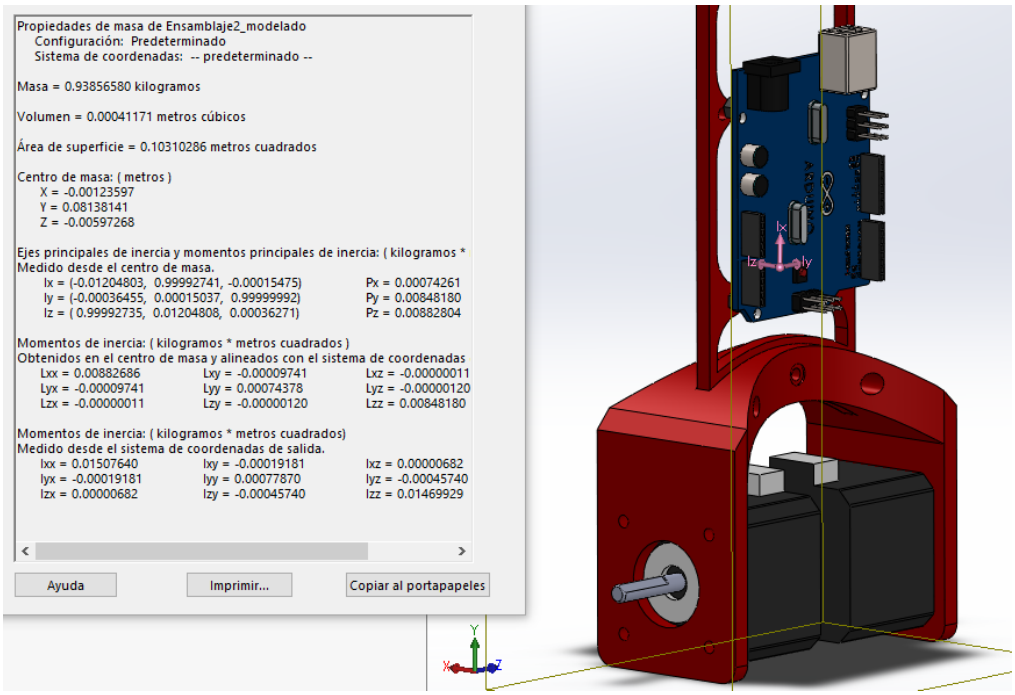


Figura 47. Momento de Inercia del cuerpo SolidWorks

I_C=0.0084818Kg*m2

El radio de la rueda R, y la distancia del eje de la rueda al centro de masas del cuerpo L lo sacamos mediante SolidWorks (Figura 48).

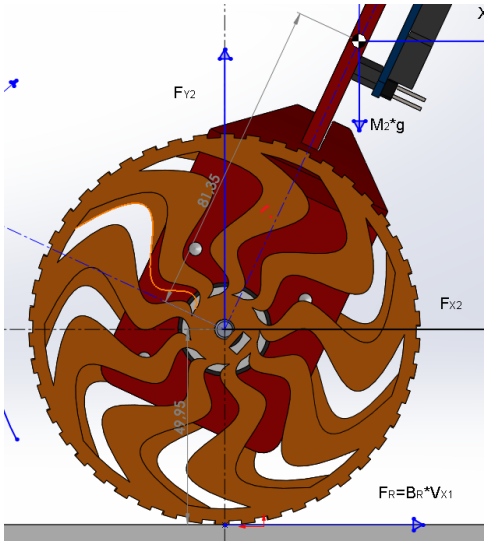


Figura 48. Radio de la rueda distancia centro de masa al eje SolidWorks

R=0.04995m y L=0.08135m

La masa de la rueda y la masa del cuerpo, lo podremos sacar mediante una báscula, que nos debería coincidir con la que nos muestra SolidWorks. La masa de la rueda se puede ver en la Figura 46 y la masa del cuerpo se puede ver en la Figura 47:

M1=0.0645kg y M2=0.9385kg

Una vez que sabes los valores de las constantes de las ecuaciones (35) y (36) y las variables de estas ecuaciones que modelan nuestro sistema podemos introducirlas en Matlab para poder calcular un PID que controlara nuestro robot. Mediante un interfaz de usuario GUIDE (Código 6) se puede ver como he creado una función para poder llamar al modelo en diferentes partes del código, esta función contiene las diferentes matrices de nuestras ecuaciones (35) y (36). Con la función "ss" creamos el sistema LTI y con "tf "creamos la función de transferencia que relaciona las entradas y salidas del sistema LTI.

Código 6. Modelo_1 para GUIDE

```
function [G] = modelo_1(varargin)
Br=0.7;Ir=0.0000859;Ki=0.5;M1=0.0645;M2=0.9358;R=0.04995;L=0.08135;Rm
=4.7;Lm=0.00072;Kv=0;
A = [0 1 0 0 0;
    0 -Br/Ir 0 0 Ki/Ir;
    0 0 0 1 0;
    0  (((Br*R)/(L*M2))+(((M1*R*Br)+Br)/(L*M2*Ir)))  0  0  (((-M1*R*Ki)-
(R*Ki))/(L*M2*Ir));
    0 -Kv/Lm 0 0 -Rm/Lm];
B = [0;
    0;
    0;
    0;
    1/Lm];
C = [0 0 1 0 0];
D = [0];
sys = ss(A,B,C,D);
G = tf(sys);
end
```

Mi interfaz de usuario para calcular el mejor PID es el mostrado en la Figura 49, en el podemos interactuar con los recuadros donde pone "P", "I" y "D" para poner los valores que nosotros queramos de PID. En estos recuadros de "P", "I" y "D" también se verá el PID calculado por el programa al pulsar el botón "AutoPID", con el botón Start graficaremos la respuesta de nuestro sistema, con los valores de PID que tengamos puestos en ese momento en los recuadros que acabo de mencionar. En ese momento se mostrará en la gráfica de arriba a la izquierda la respuesta del sistema a un impulso unitario, arriba a la derecha la respuesta del sistema a un escalón unitario, abajo a la izquierda la respuesta del sistema a una rampa creciente y abajo a la derecha la respuesta del sistema a una entrada sinusoidal.

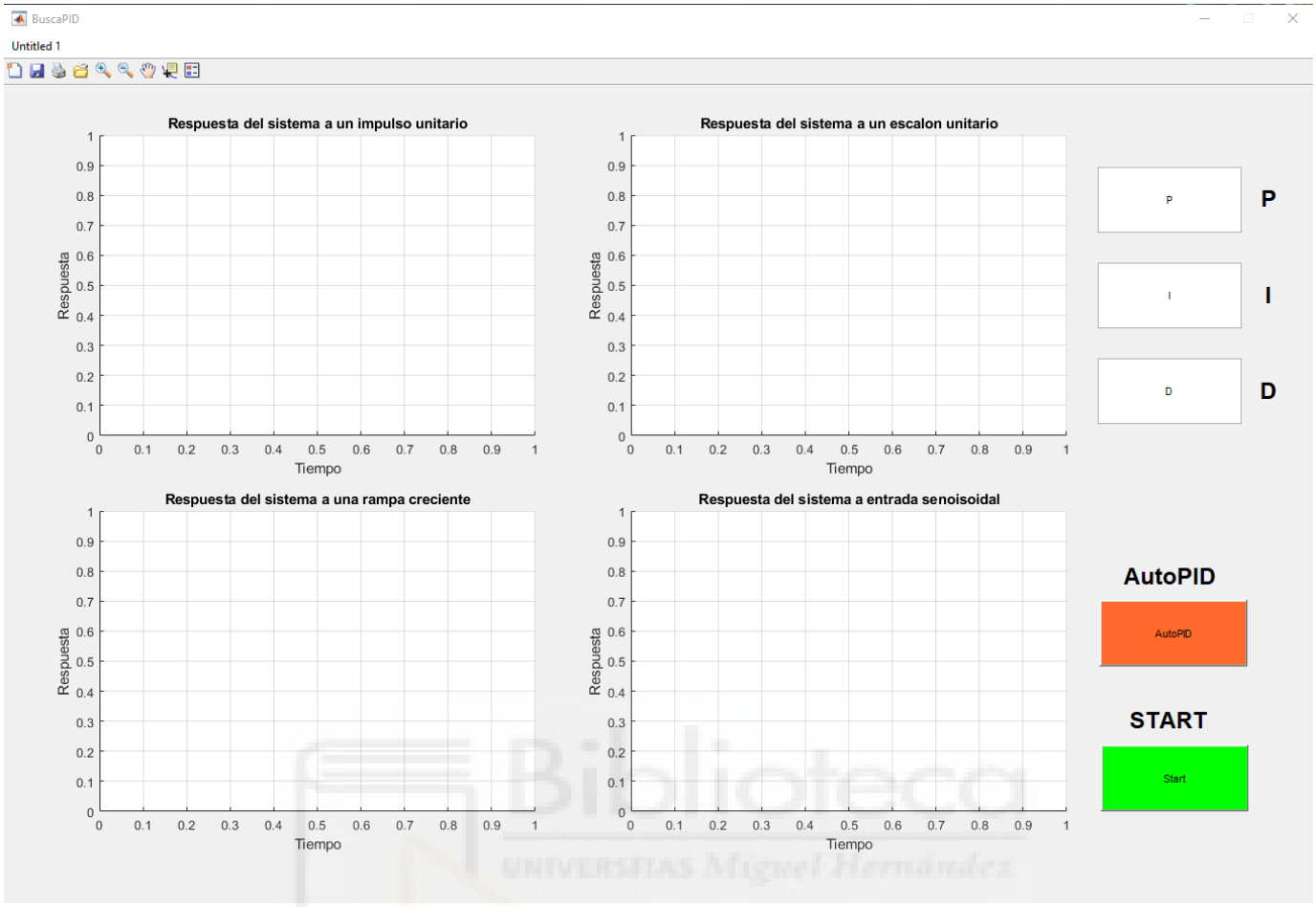


Figura 49. Interfaz de usuario mediante GUIDE

El Código 7, es la parte de código que se ejecuta al pulsar el botón “AutoPID” en ella se ejecuta la función “objetive”, como función anónima, esta se ejecutara 10000 debido a la condición de la función “optimset”, esta función se ejecutara ese número de veces cambiando los valores de “kp”,”ki” y “kd” buscando que devuelva el numero más pequeño en “error”, esto lo hacemos con la función “fminsearch”, siendo “objetive” la función donde queremos encontrar su mínimo y “initialParamas” los parámetros iniciales que se irán modificando para buscar el mínimo, “initialParams” serán los @(params) de la función anónima “objective”.

El valor de “error” lo podremos calcular para diferentes tipos de entradas, modificando el código. En este caso tiene una señal escalón (Código 7), también podemos modificar por código la manera en que tiene de calcular el error yo planteo dos modos en el Código 7 una como error absoluto medio MAE “error = sum(abs(y - r), 'all')” o mediante el error cuadrático MSE “error = sum((y - r).^2, 'all')”. El MAE es la diferencia absoluta promedio entre los valores observados (y) y los valores reales (r). El MSE mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima.

Según si elegimos uno u otro obtendremos un error u otro.

Si queremos que trate todos los errores de la misma manera independientemente de su tamaño, entonces el MAE sería la mejor opción.

Si queremos penalizar más los errores grandes, la mejor elección es el MSE porque al elevar al cuadrado los errores, los errores más grandes tienen un impacto desproporcionadamente mayor en el MSE que en el MAE.

En nuestro caso elegimos el error MSE ya que el robot tiene que ser más rápido cuando se aleja del ángulo objetivo porque cuanto más se aleje de este punto más le costara llegar a él.

Código 7. BuscaAutoPID para GUIDE

```
function [Kp Ki Kd]=BuscaAutoPID()

%Funcion de transferencia del sistema LTI
[G] = modelo_1();

% Función objetivo para minimizar
objective = @(params) calculateError(params, G);
% Valores iniciales para los parámetros del controlador PID
initialParams = [0.1, 0.1, 0.1];
options = optimset('MaxFunEvals',10000);

% Realizar la búsqueda de los valores óptimos
optimalParams = fminsearch(objective, initialParams,options);

% Extraer los valores óptimos de los parámetros
Kp = optimalParams(1);
Ki = optimalParams(2);
Kd = optimalParams(3);


% Función auxiliar para calcular el error
function error = calculateError(params, G)

    Kp = params(1)
    Ki = params(2)
    Kd = params(3)
    C = pid(Kp, Ki, Kd);

    % Sistema multiplicado por el PID en lazo cerrado
    % unitaria
    T = feedback(C*G, 1);
    t = 0:0.01:10;
    %% Señal escalon
    r = ones(size(t));
    [y, ~] = step(T*r, t);
    y = squeeze(y);
    %% Señal rampa
    %r= t;
    %[y, t] = lsim(T, r, t);

    %% Señal de entrada senoidal
    %r= sin(t);
    %[y, t] = lsim(T, r, t);

    %% Calcular el error (desviación de la respuesta respecto a la
referencia)
    error = sum((y - r).^2, 'all');
    %error = sum(abs(y - r), 'all');
end
end
```

6.2 Simscape Multibody para Simulink de Matlab

Otro tipo de simulación que he utilizado para encontrar el mejor regulador para mi robot, lo he realizado mediante Simscape Multibody para Simulink Matlab y SolidWorks. Simscape Multibody es un complemento que se utiliza en aplicaciones CAD, como SolidWorks, para exportar modelos de ensamblaje a Simscape Multibody. Este complemento exporta un modelo de ensamblaje CAD como archivos XML y archivos de geometría del cuerpo los cuales se pueden convertir en modelos Simscape Multibody, utilizando la función smimport.

Estos archivos contienen las propiedades físicas que hemos definido en SolidWorks, al indicar el tipo de material del que están hechos. Estas propiedades incluyen la masa, el momento de inercia, las dimensiones, la elasticidad, etc. Con estos archivos que generaremos en SolidWorks y mediante la herramienta Simscape Multibody en Simulink Matlab, podremos generar relaciones de movimiento entre diferentes partes del ensamblaje. El comportamiento de cada parte será el que se definió en SolidWorks, lo que nos permitirá simular dinámicas multicuerpo.

Además, Simscape Multibody permite la simulación de sistemas mecánicos en un entorno 3D, lo que facilita la visualización y el análisis de los resultados de la simulación. También es posible definir componentes mecánicos modulares que se pueden reutilizar fácilmente en otros sistemas.

Esta simulación se ha hecho con los diseños 2, 3 y 4. La única diferencia, importante, entre un diseño y otro es la distribución del peso en el robot, esto modifica el centro de masas del robot. Como veremos solo con esto, el control que podremos hacer en nuestro robot cambiara de un diseño a otro.

6.2.1 Simscape. Diseño 2

En la Figura 50 se pueden ver las 3 subpartes de nuestro robot, las cuales se mueven independientemente entre sí, estas subpartes son “Ensamblaje Cuerpo”, “Rueda izquierda” y “Rueda derecha”. Estas 3 subpartes contienen todas las partes de nuestro robot, batería, controladora, motores etc. y las propiedades físicas de estas. Cada parte de las subpartes tiene su propiedad física las cuales son independientes y diferentes entre ellas, se tendrán en cuenta asi en la simulación en Simulink.

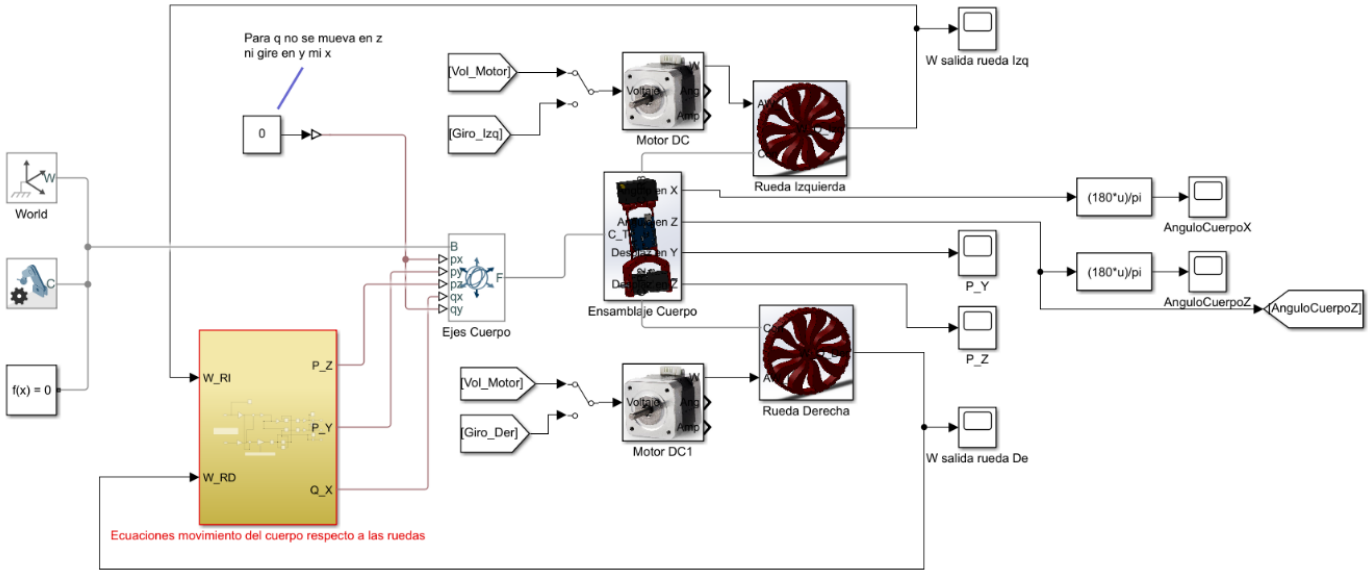


Figura 50. Relaciones entre partes Simulink. Diseño 2

En el bloque de las ruedas tenemos una relación de movimiento de rotación (Figura 51) con la subparte “Ensamblaje Cuerpo” el movimiento de esta rueda es transmitido por el motor bloque “Motor DC”(Figura 50), mediante “AW_I”.

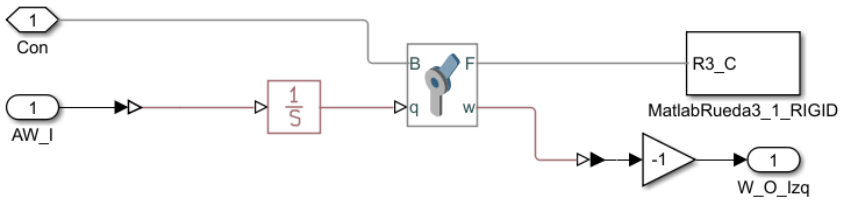


Figura 51. Bloque “Rueda” Simulink

El interior del bloque “Motor DC” mostrado en la Figura 52, es la única parte de la simulación en la cual no podemos heredar las propiedades físicas de SolidWorks, ya que son propiedades eléctricas y mecánicas, asi que las hemos tenido que modelar. El circuito de color azul (Figura 52) corresponde a la parte eléctrica del motor en la que tenemos una fuente de alimentación, en la cual simularemos el voltaje que le entra a los motores, es nuestra señal por controlar, tenemos un amperímetro para conocer los amperios que pasan por el motor, una resistencia y bobina representando la resistencia e inductancia del bobinado del motor. La unión entre la parte eléctrica y mecánica del motor es mediante un conversor de rotación electromecánica. En la parte verde tenemos la parte mecánica del motor en la cual tenemos un volante de inercia para simular la inercia en el eje del motor y también la representación del rozamiento del motor, por último, tenemos la salida de aceleración y velocidad del motor. Los valores de las constantes de rozamiento, inercia, resistencia e inductancia del motor han sido calculados en el punto 4 y han sido introducidos en esta parte de la simulación.

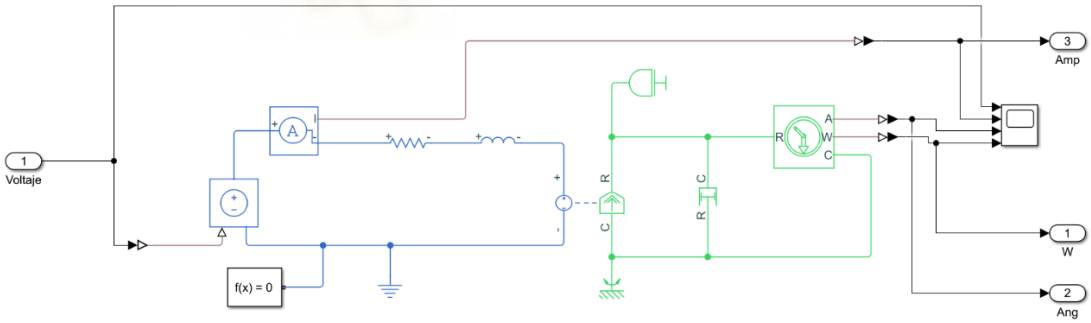


Figura 52. Bloque "Motor DC" Simulink

El movimiento de la subparte “Ensamblaje Cuerpo”, Figura 50, se moverá respecto a “World” mediante “Eje Cuerpo” este bloque le dará sus propiedades de movimiento y este movimiento es controlado mediante el bloque “Ecuaciones movimiento del cuerpo respecto a las ruedas” el interior de este bloque se puede ver en la Figura 53. En este bloque tenemos la relación de movimiento de los bloques “Rueda Izquierda” y “Rueda Derecha”, que giran en un suelo sin deslizamiento, con el bloque “Ensamblaje Cuerpo” estas relaciones las dan las ecuaciones de la Figura 54. En la Figura 53 tenemos estas ecuaciones en forma de bloques de Simulink.

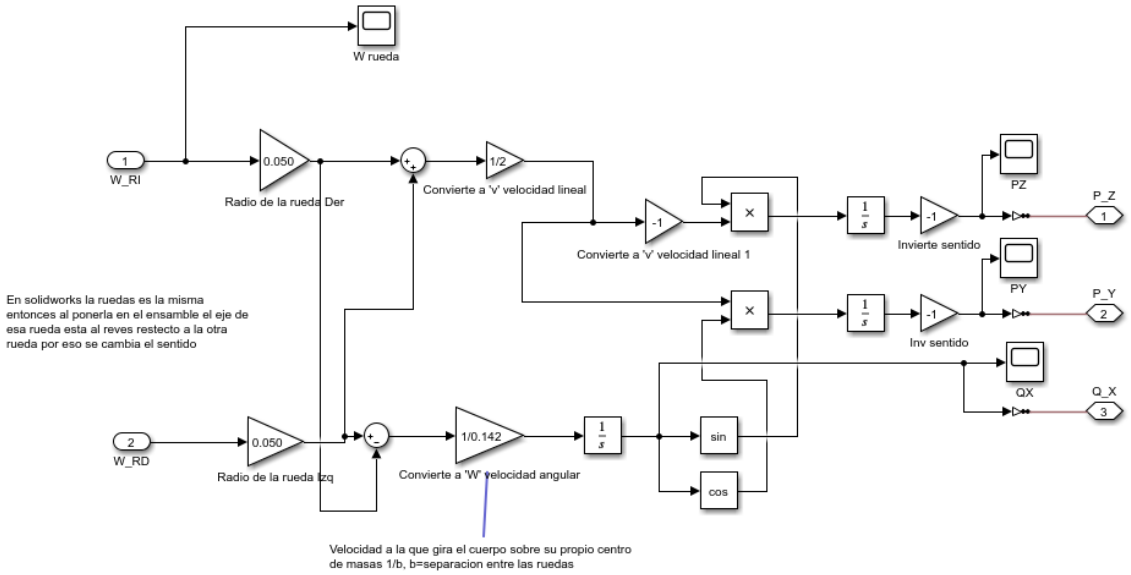


Figura 53. Bloque "Ecuaciones movimiento del cuerpo respecto a las ruedas" Simulink

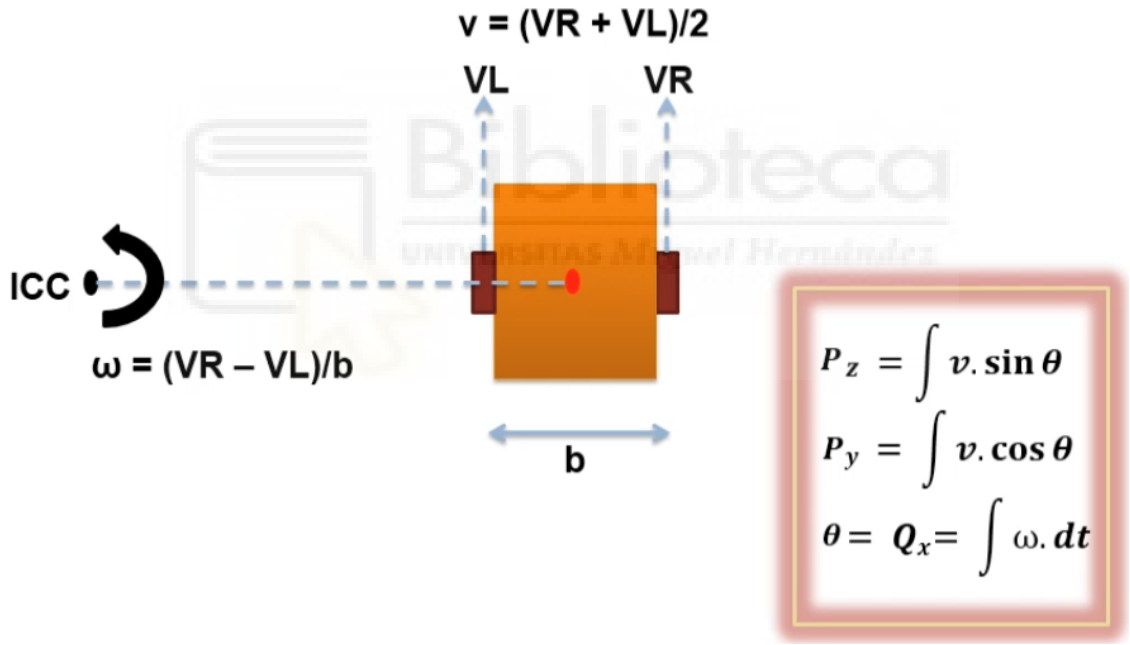


Figura 54. Ecuaciones que relacionan el movimiento de las ruedas y el cuerpo del robot

Una vez que tenemos nuestras partes del diseño bien exportadas con todas sus propiedades físicas, el movimiento relativo entre ellas es correcto y tenemos modelado los motores, nos toca crear un regulador que controle la entrada de voltaje al motor "Vol_Motor", para que el alguno "AnguloCuerpoZ" sea el que nosotros indiquemos en "Angulo Obj", la vertical del cuerpo del robot, para ello he usado el bloque "PID Controller", Figura 55.

El bloque "PDI Controller" nos da la facilidad de usar el PID Tuner, el cual nos ayudara a encortrar los valores óptimos de PID para nuestro controlador, como vemos en la Figura 55 con el bloque "ActivadorPID" retardaremos la acción del PID esto lo hacemos para no partir del punto de equilibrio a la hora de controlar los motores.

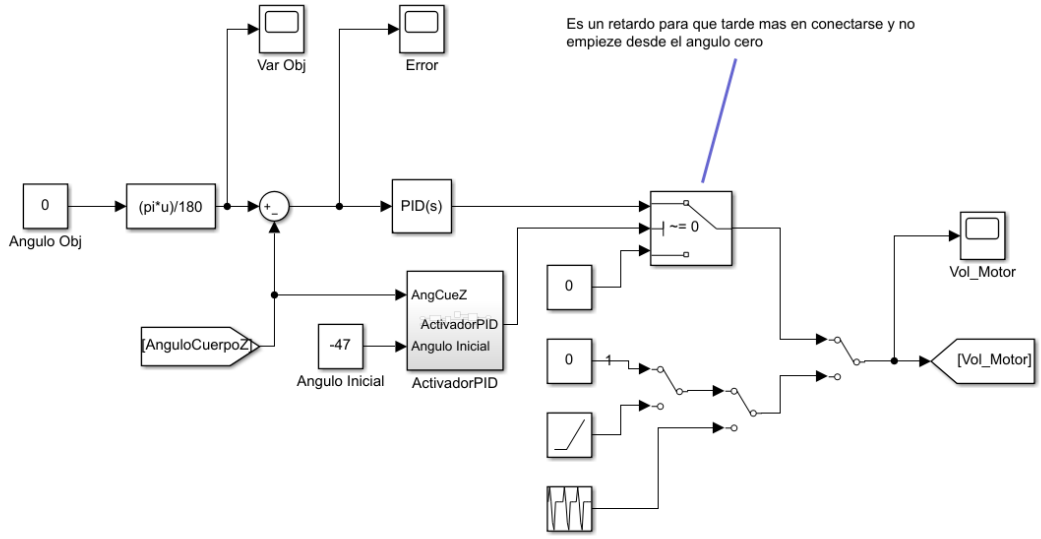


Figura 55. PID Simulink

Una vez se realizaron varias pruebas modificando los valores de PID y el coeficiente de filtro "N", este filtro se usa para suavizar la acción derivativa, se llegaron a los valores óptimos mostrados en la Figura 56.

Figura 56. Valor PID Simulink. Diseño 2

En la Figura 57, tenemos la representación gráfica del ángulo deseado, línea azul, y del ángulo del cuerpo de nuestro robot, en amarillo, con el controlador PID mostrado en la Figura 56. En la gráfica tenemos en el eje de las ordenadas, ángulos en radianes, y en el eje de abscisas el tiempo, en segundos. Como podemos ver cuando el cuerpo llega aproximadamente a -0.82 radianes, la acción del PID empieza, vemos que tenemos una respuesta rápida del sistema con una pequeña sobre oscilación, después de esto vemos como el cuerpo se estabiliza sobre el setpoint.

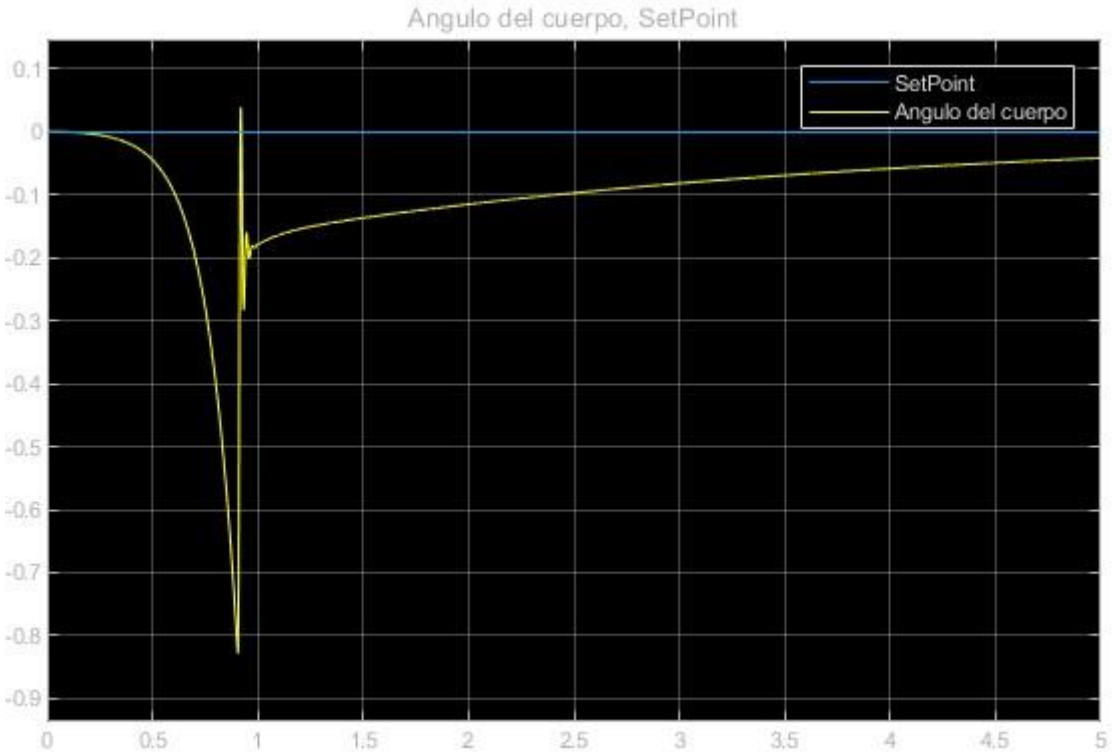


Figura 57. SetPoint vs salida del sistema. Diseño 2

Una particularidad que tiene la herramienta Simscape Multibody para Simulink es que podemos visualizar el comportamiento de nuestro modelo mediante un renderizado 3D de los archivos CAD, que hemos creado en Solidworks, y asi poder ver visualmente cual sería el comportamiento de nuestro sistema antes diferentes estímulos o PIDs. En la Figura 58 podemos ver como se estabiliza nuestro robot con el PID de la Figura 56, partiendo de una posición de equilibrio, no empezara la hacían de control del controlador hasta que el cuerpo del robot no llega por primera vez a los -47 grados o -0.82 radianes, una vez llegados el controlador mandara la señal de control a los motores estabilizando el cuerpo del robot.

Figura 58. Simulación Simulink. Diseño 2

Como podemos ver en la Figura 57 y en la Figura 58 el comportamiento de nuestro robot tanto en la gráfica como en el renderizado es el mismo con lo que la simulación es correcta.

6.2.2 Simscape. Diseño 3

La simulación que hicimos en el punto 6.2.1 la repetí, pero ahora con el diseño chasis 3, Figura 59, la parte del diseño en simulink que relaciona el comportamiento de las diferentes partes del robot es igual que la del punto anterior, solo se ha cambiado las partes del nuevo diseño y las propiedades físicas asociados a estas partes. Como explicamos en el punto 5.3 este diseño mejora el reparto de pesos del robot, como veremos a continuación esto hace que el robot no necesite un controlador para estabilizarse. El bloque "Señal de entrada a los motores" representa el valor de voltaje que llegara a nuestros motores durante la simulación es una señal directa sin pasar por ningún regulador.

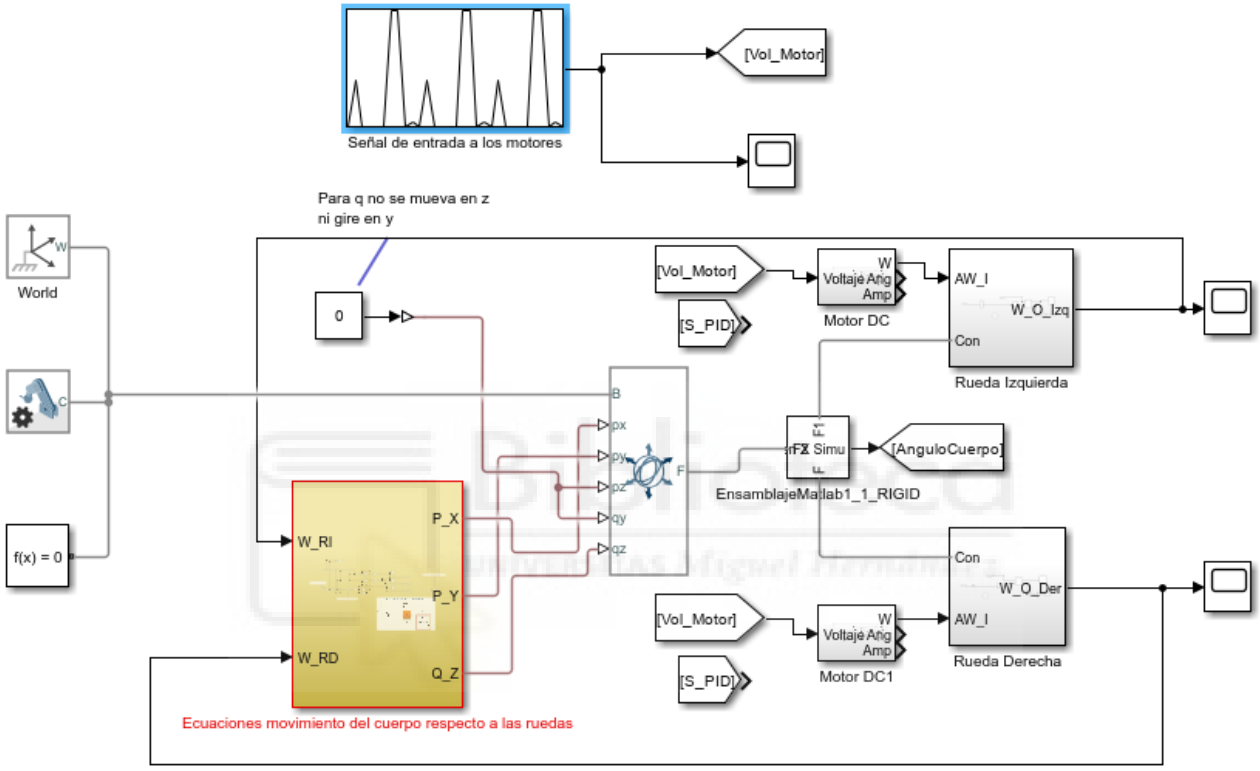


Figura 59.Relaciones entre partes Simulink. Diseño 3

En la Figura 60 y Figura 61 podemos ver el resultado de la simulación en el diseño 3, ante la entrada directa, sin pasar por ningún regulador (Figura 59) como podemos ver la señal de entrada a los motores es totalmente aleatoria. Se puede apreciar como tanto en la gráfica como en el renderizado 3 el cuerpo del robot siempre tiende a volver a su vertical, esto es debido, como ya he explicado en el punto 5.3 a que el centro de masa del robot está por debajo de su eje de rotación, el eje de los motores. Esto hace que no necesitemos ningún regulador para hacer que el robot recupere su vertical ya que por el reparto de pesos en el robot hace que el robot tienda a estar siempre en vertical respecto del suelo.

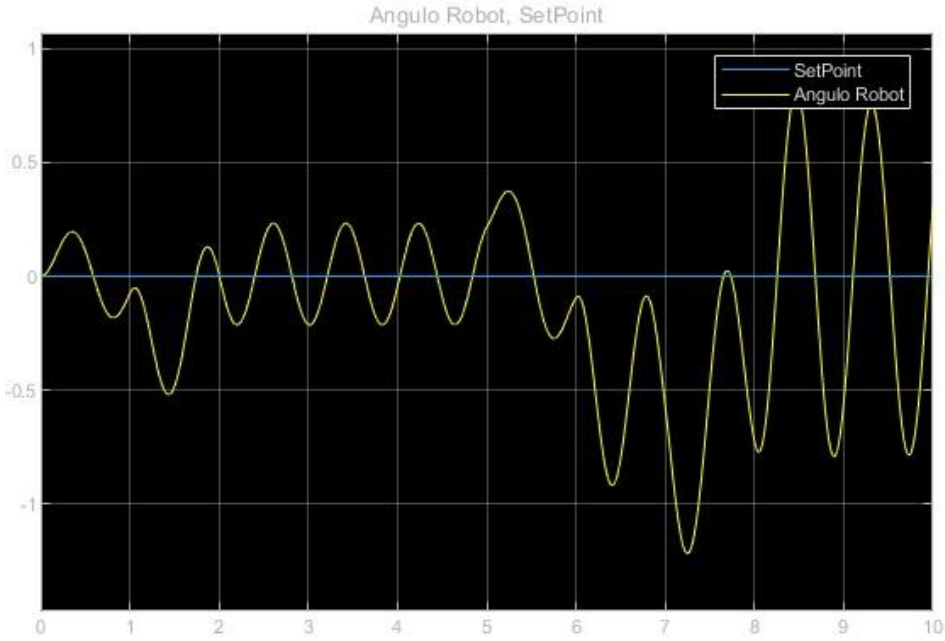


Figura 60. SetPoint vs salida del sistema. Diseño 3

Figura 61. Simulación Simulink. Diseño 3

6.2.3 Simscape. Diseño 4

En el diseño 4 el centro de masas no está por debajo del eje de los motores como en el diseño 3, esto hace que tengamos que colocar un controlador PID para poder controlar la posición vertical del robot. En comparación con el diseño 2, la batería está colocada por debajo de los ejes de los motores, esto hace que el eje de masas del diseño 4 este mucho mas cerca del eje de los motores, esto hace que nuestro robot sea mucho más fácil de controlar.

Debido a que este robot tiene más fácil volver a su vertical, los valores óptimos para el regulador PID pueden ser diferentes obteniendo en todos los casos el objetico de que el robot recupere la vertical. En la Figura 62, podemos ver unos de los valores de PID con los que el robot consiguió su objetivo de llegar a colocarse en vertical.

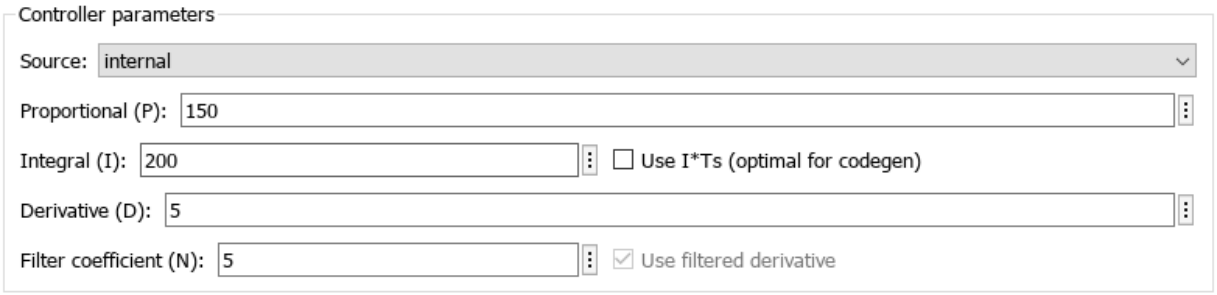


Figura 62. Valor PID Simulink. Diseño 4

En la Figura 63 y Figura 64 podemos ver el resultado de la simulación del diseño 4, con cun regulador con los valores mostrados en la Figura 62. El controlador PID no empezara a funcionar hasta que el cuerpo de nuestro robot no llegue a los 47 grados o 0,82 radianes una vez llegado los motores empezaran a funcionar rápidamente intentando buscar la vertical del robot, vemos que en esta acción no tenemos sobre oscilaciones y como el ángulo de nuestro robot se va estabilizando en un ángulo óptimo para nosotros.

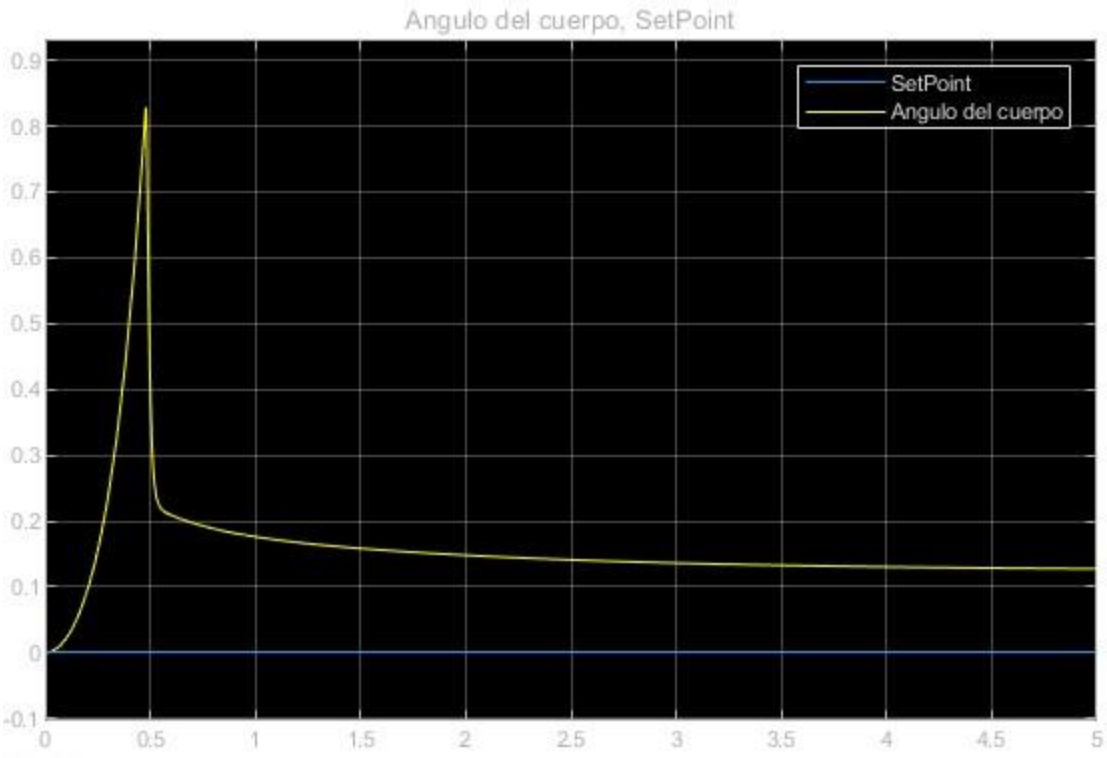


Figura 63. SetPoint vs salida del sistema. Diseño 4

Figura 64. Simulación Simulink. Diseño 4

7 PROGRAMACIÓN DEL ROBOT

Para controlar los motores y poder leer los diferentes sensores he usado la controladora Rasberry Pi 3b+. La Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito, se puede considerar un ordenador ya que tiene procesador, chip gráfico y memoria Ram. El código de programación usado para controlar nuestro robot es Python, he elegido Python por ser un lenguaje de alto nivel lo que hace que su sintaxis sea clara y concisa, Python es uno de los lenguajes más usado en todo el mundo lo que hace que tenga una gran cantidad de recursos y bibliotecas de terceros, muchas de las cuales están orientadas a la robótica.

La estructura de mi programa esta dividida en diferentes hilos, los hilos también conocidos como thread comparten memoria y recursos, lo que permite que los datos se comuniquen y se sincronicen de manera eficiente entre ellos. Los hilos son útiles cuando quieres simular la ejecución en paralelo. En mi caso es necesario para poder ejecutar procesos muy rápidos como la lectura de los enconder y otros que no necesitan ser tan rápidos como el control de los motores. A continuación, voy a explicar lo que hace cada hilo.

7.1 Hilo encoder

Para saber la posición del robot los motores incorporan unos enconder los cuales mandan 22 pulsos por cada vuelta del enconder, 11 pulsos por un cable y otros 11 pulsos por otro cable estas señales están desfasadas para indicar el sentido de giro del motor, como tiene una reductora de relación 1/45 necesitarnos 990 pulsos para dar una vuelta de la rueda, si esto lo pasamos a grados para una vuelta, 360 grados, tenemos por cada pulso 0.363636 grados o lo que es lo mismo 0,0063466 radianes en la rueda. En el Código 8 podemos ver como por cada motor se va aumentando o restando esos radianes a la cuenta de radianes de cada motor, angulo_ejeD y angulo_ejeI, se sumara cuando en la señal A tengo un flanco positivo y en la señal B tengo un flanco negativo, se restara cuando teniendo un flanco positivo en A en la señal B tengamos un flanco positivo tambien.

Sabiendo el diámetro de la rueda y los radianes que ha girado la rueda se puede calcular lo que se ha desplazado linealmente cada rueda, en el Código 8 podemos ver este cálculo en delta_s_I y delta_s_D. Conocido esto podemos saber lo que se desplaza nuestro robot como la suma de estos dos, dividido entre dos, delta_s. Delta_theta es el ángulo de giro sobre sí mismo respecto al suelo que será la diferencia de lo que gira cada rueda dividido entre la separación de las ruedas.

Para poder compartir estos datos entre diferentes hilos no podemos usar variables normales ya que podría ocurrir una condición de carrera, que ocurre cuando el resultado de un programa depende del orden específico en que se ejecutan los hilos. Por ejemplo, si dos hilos están incrementando simultáneamente el valor de una variable compartida, uno de los incrementos podría perderse, llevando a resultados incorrectos. Otro problema es la inconsistencia de datos, que puede ocurrir cuando varios hilos leen y escriben en una estructura de datos compartida y uno de los hilos ve una versión intermedia de los datos mientras otro hilo está actualizando los datos. Para compartir información entre diferentes hilos usamos las colas, en el caso del Código 8 nuestra cola se llama pos, las colas son estructuras de datos en Python que siguen el principio de FIFO (First In First Out o Primero en Entrar, Primero en Salir). Esto significa que el elemento que entra primero sale primero. Las colas proporcionan una forma segura de manejar los datos

producidos por un hilo y consumidos por otro hilo. Sin una cola, los hilos tendrían que acceder a la misma estructura de datos, lo que podría llevar a problemas de concurrencia.

Código 8. Hilo Encoder

```
while True:
    currentStateD = GPIO.input(encoderDPinA)
    if currentStateD != lastStateD:
        if GPIO.input(encoderDPinB) != currentStateD:
            angulo_ejeD +=0.0063466
        else:
            angulo_ejeD -=0.0063466
    lastStateD = currentStateD
    currentStateI = GPIO.input(encoderIPinA)
    if currentStateI != lastStateI:
        if GPIO.input(encoderIPinB) != currentStateI:
            angulo_ejeI +=0.0063466 def acc_gyro_hilo():
        else:
            angulo_ejeI -=0.0063466
    lastStateI = currentStateI
    # Convertir los ángulos de rotación a la distancia recorrida por cada rueda
    delta_s_I= -(angulo_ejeI * DIAMETRO_RUEDA/2)
    delta_s_D = angulo_ejeD * DIAMETRO_RUEDA/2
   # Calcular el desplazamiento del robot
    delta_s = (delta_s_I + delta_s_D) / 2
    # Calcular el cambio en el ángulo de orientación del robot
    delta_theta = (delta_s_D - delta_s_I) / SEPARACION_RUEDAS)
    # Poner los valores en la cola
    pos.put((delta_s, delta_theta))
```

7.2 Hilo giro

Para poder mantener el robot en vertical necesito conocer el ángulo que tiene el cuerpo del robot respecto al suelo. Para ello he usado una controladora de vuelo Pixhawk con Ardupilot, he elegido esta opción ya que ArduPilot realiza internamente un filtrado de los datos del giroscopio y el acelerómetro para obtener una estimación más precisa y estable del ángulo.

En un principio use un giroscopio y acelerómetro externo pero los datos recogidos daban errores y eran inestables. Por ello decidí usar una controladora externa la cual se comunica con mi rasberry pi mediante el protocolo de comunicación Mavlink.

En el Código 9, podemos ver que gracias a la librería pymavlink, cuando el mensaje que llega por el puerto serial a nuestra rasberry observa una cadena de caracteres tipo ATTITUDE se guarda en una variable el mensaje roll. Posteriormente este variable se almacena en la cola y, para ser leída en el hilo de control.

Código 9. Hilo giro

```
def acc_gyro_hilo():
    while True:
        try:
            #Lee el mensaje
            msg = master.recv_match() #Intercepta mensajes conforme llegan
            if msg is not None:
                #Si el mensaje es de tipo ATTITUDE (actitud)
                if msg.get_type() == 'ATTITUDE':
                    # Si el mensaje es roll lo guardas
                     roll = math.degrees(msg.roll)
                     # Poner los valores en la cola
                    y.put(roll)
        except Exception as e:
            print('Error:', e)
```

7.3 Hilo lidar

Mara poder considerarse mi robot, un robot autónomo AMR necesita recoger información de su alrededor y modificar su comportamiento en función de lo que se encuentre. Para ello mi robot esta dotado con un lidar 360, FHL-LD19.

La conexión entre este sensor y la rasberry es mediante serial, gracias a la documentación del sensor se puede ver que el LD19 adopta una comunicación unidireccional. Después de funcionar de forma estable, empieza a enviar paquetes de datos de medición sin enviar ningún comando. El formato del paquete de medición se muestra en la Tabla 4.

Tabla 4. Protocolo del sensor LD19

Header	VerLen	Speed		Start angle		Data	End angle		Timestamp		CRC check
54H	1 Byte	LSB	MSB	LSB	MSB	.....	LSB	MSB	LSB	MSB	1 Byte

- Encabezado: La longitud es de 1 Byte, y el valor es fijo en 0x54, indicando el comienzo del paquete de datos;
- VerLen: La longitud es de 1 Byte, los tres bits superiores indican el tipo de paquete, que actualmente es fijo en 1, y los cinco bits inferiores indican el número de puntos de medición en un paquete, que actualmente es fijo en 12, por lo que el valor del byte es fijo en 0x2C.
- Velocidad: La longitud es de 2 Bytes, la unidad es grados por segundo, indicando la velocidad del lidar.
- Ángulo inicial: La longitud es de 2 Bytes, y la unidad es 0.01 grados, indicando el ángulo inicial del punto de datos del paquete.
- Datos: Indica los datos de medición, una longitud de datos de medición es de 3 bytes.
- Ángulo final: La longitud es de 2 Bytes, y la unidad es 0.01 grados, indicando el ángulo final del punto de datos del paquete.
- Marca de tiempo: La longitud es de 2 Bytes, la unidad es milisegundos, y el máximo es 30000. Cuando llega a 30000, se vuelve a contar, indicando el valor de la marca de tiempo del paquete de datos.
- Verificación CRC: La longitud es de 1 Byte, obtenida de la verificación de todos los datos anteriores excepto ella misma.

Como vemos en el Código 10 estamos leyendo todo el rato los bytes que nos llegan, cuando llegan dos bytes seguidos en el que el primero es x54 y el segundo x2c, entonces guarda el mensaje que nos llega. Guardando el ángulo inicial, start_angle, el angulo final, end_angle y unas medidas de distancia relacionadas con este rango de ángulos, si en nuestro diccionario, dici, ya tenemos en este rango start_angle-end_angle algún valor lo sobrescribimos con estos nuevos valores. Sabemos que en cada mensaje nos llegaran 11 distancias estas distancias las interpolamos al rango de grados en el que estamos, esto lo hacemos mediante le bucle for, una vez hecho esto, guardando los nuevos puntos en la librería si estos nuevos puntos están entre 285 a 360 y de 0 a 75 grados.

Código 10. Hilo lidar

```
while True:
    # Leer un byte
    byte = ser.read(1)
    # Comprobar si el byte es 0x54, Leer el siguiente byte, Comprobar si el siguiente byte es 0x2C
    if byte == b'\x54':
        next_byte = ser.read(1)
        if next_byte == b'\x2C':
            # Hemos encontrado el inicio del paquete, leer los bytes restantes
            data = ser.read(45)  # Asumiendo que el tamaño total del paquete es 47 bytes
            start_angle = struct.unpack("<H", data[2:4])[0]
            end_angle = struct.unpack("<H", data[40:42])[0]
            # Crear una lista de las claves que están en el rango especificado
            claves_a_eliminar = [k for k in dici if (start_angle - 100) <= k <= (end_angle + 100)]
            # Eliminar las claves
            for k in claves_a_eliminar:
                dici.pop(k, None)
            # Iterar sobre los puntos de medición
            for i in range(12):
                # Extraer la distancia y la intensidad de cada punto
                distance = struct.unpack("<H", data[4 + i*3 : 6 + i*3])[0] / 1000  # Convertir a metros
                # Calcular el ángulo de cada punto por interpolación lineal
                if end_angle > start_angle:
                    step = (end_angle - start_angle) / 11
                    angle = (start_angle + step * i)
                elif end_angle < start_angle:
                    step = (end_angle + 36000 - start_angle) / 11
                    angle = (start_angle + step * i)
                if angle >= 36000:
                    angle = angle - 36000
                #Guardamos los nuevos puntos, entre 285- 360 grados y 0-75 grados, en la cola
                if 36000 >= angle >=28500 or 0<= angle <= 7500:
                    dici[angle] = distance
```

Hicimos un hilo para poder graficar los valores que estamos guardado en nuestro diccionario, Código 11. El resultado se muestra en la Figura 65, esta grafica se va actualizando en tiempo real cada 1.1 segundos. Debemos parar entre muestra y muestra en la gráfica poque sino el buffer aumenta más rápido de lo que es capaz de actualizarse la gráfica y esto provocaría un retardo que sería cada vez mayor entre muestra y muestra.

Código 11. Hilo grafica lidar

```
def dibu():
    global dici
    while True:
        #Sepramos las keys y los valores del diccionario en dos listas
        key_dici = list(dici.keys()) #los angulos
        valor_dici = list(dici.values()) # la distancia
        # Limpiar el eje antes de dibujar los nuevos puntos
        ax.clear()
        ax.plot(-((360.0 - np.array(key_dici)/100) / 180 * np.pi), valor_dici, 'ro', markersize=2)  # Convertir a radianes
        # Configurar la gráfica
        ax.set_rticks([0.5, 1, 1.5, 2])  # Anillos a 0.5, 1, 1.5 y 2 metros
        ax.set_rmax(2)  # Máximo a 2 metros
        ax.set_theta_zero_location('N')  # El ángulo cero (0 radianes) en la parte superior
        ax.set_theta_direction(-1)  # Los ángulos aumentan en sentido horario
        # Actualizar
        plt.draw()
        plt.pause(1.1) #Tenemos que pararlo pq sino el buffer aumenta más rápido de lo que es capaz de mostrar
```

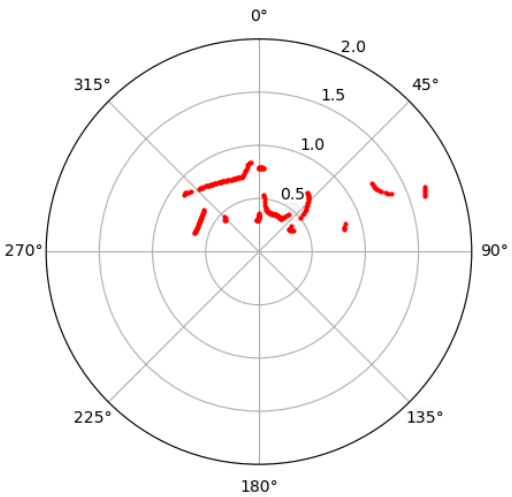


Figura 65. Grafica lidar

7.4 Hilo de control

He creado una clase para poder definir y controlar los motores del robot esta clase aparece en el Código 12.

Código 12. Clase Motor

```
class Motor:
    def __init__(self, pi, en, in1, in2):
        self.pi = pi
        self.en = en
        self.in1 = in1
        self.in2 = in2
        # Configura los pines EN, IN1 e IN2 como salida
        self.pi.set_mode(self.en, pigpio.OUTPUT)
        self.pi.set_mode(self.in1, pigpio.OUTPUT)
        self.pi.set_mode(self.in2, pigpio.OUTPUT)
        # Establece el estado de los pines IN1 e IN2
        self.pi.write(self.in1, 0)  # GPIO.LOW
        self.pi.write(self.in2, 0)  # GPIO.LOW
    def run(self, speed, direction):
        if direction == 'adelante':
            self.pi.write(self.in1, 1)
            self.pi.write(self.in2, 0)
        elif direction == 'atras':
            self.pi.write(self.in1, 0)
            self.pi.write(self.in2, 1)
        # Configura PWM en los pines EN
        # Parámetros: (GPIO, frecuencia, ciclo de trabajo)
        self.pi.hardware_PWM(self.en, 50, (speed*10000))  # 50% de ciclo de trabajo
    def stop(self):
        self.pi.hardware_PWM(self.en, 50, 0)  # 50% de ciclo de trabajo
```

En el Código 13, tenemos el hilo de control del motor en este se observa que podemos parar los motores el cualquier momento cuando pulsemos un mando externo, además obtendremos las variables de posición y rotación del cuerpo del robot, las cuales obtuvimos en los hilos explicados anteriormente y los cuales están almacenados en las colas pos y roll. El control de los motores se hace mediante un controlador PID, pid_ang, el cual parte con los valores calculados en el punto 6 y el cual fue modificado a la hora de probar el robot. Este controlador tiene como retroalimentación el valor de la cola roll, que obtiene del Hilo giro, su setpoint se ira modificando

en función de si el robot ha llegado la poción objetivo o no. Si el robot está en la posición objetivo el setpoint del ángulo será 0 grados, pero si no está en esta posición el setpoint será controlado por otro PID, pid_pos. El setpoint de este nuevo PID sera la posición deseada y su retroalimentación el valor de la cola pos. Todo el control de los motores se hace con la clase Motor la cual se muestra en el Código 12.

Código 13. Hilo de control

```
def control_hilo():
    # Configuración de los pines de los motores
    motor1 = Motor(pi, 18, 4, 17)
    motor2 = Motor(pi, 13, 27, 22)
    while True:
        #Obtienes la posicion del joystick
        pygame.event.pump()
        x = joystick.get_axis(4)
        #Para el buqle While
        if joystick.get_button(1):
            break
        # Obtener los valores de la cola de la posición, giro sobre si mismo y giro del cuerpo
        while not pos.empty():
            posix, aux_ang_cuerpo = pos.get()
            try:
                pos.queue.clear()  # Vacía la cola después de obtener el último valor
            except:
                pass
        while not q.empty():
            roll = q.get()
            try:
                q.queue.clear()  # Vacía la cola después de obtener el último valor
            except:
                pass
        control_pos = pid_pos(posiX)
        pid_ang.setpoint = control_pos
        control_ang = pid_ang(roll)
        if control_ang > 0.01:
                motor1.run(abs(int(control_ang)), 'atras')
                motor2.run(abs(int(control_ang)), 'atras')
        elif r control_ang < -0.01:
                motor1.run(abs(int(control_ang)), 'adelante')
                motor2.run(abs(int(control_ang)), 'adelante')
    motor1.stop()
    motor2.stop()
```

8 PRESUPUESTO. MATERIAL Y EQUIPOS NECESARIOS

A lo largo de este proyecto el robot ha ido sufriendo modificaciones, estas modificaciones han sido para solucionar problemas que he encontrado mientras desarrolla el proyecto y también han sido modificaciones para añadir nuevas funcionalidades a mi robot. En la Tabla 5 se han incluido todos los componentes que se han utilizado para el desarrollo del robot, también se han incluido los aparatos de medida utilizados para las diferentes calibraciones en la Tabla 6 y maquinas utilizadas para el desarrollo del chasis en la Tabla 7. Finalmente, en la Tabla 8 se describe brevemente el coste total del proyecto.

-Gastos en material para el robot

Tabla 5. Componentes eléctricos y electrónicos del robot

COMPONENTES ELECTRICOS Y ELECTRONICOS DEL ROBOT			
Cantidad	Descripción	Precio/Unid	Subtotal
Diseño 1 y 2			
1	Arduino UNO Rev3	20,40€	20,40€
1	Modulo sensor acelerómetro-giroscopio MPU-6050	4,48€	4,48€
1	CNC Shield V3	6,99€	6,99€
2	Driver de motor A4988	6,49€	12,98€
2	Motores paso a paso Nema 17	33,28€	66,56€
	Consumibles y materias primas: filamento, tornillos, cable…	35€	35€
		TOTAL	146,41€
Diseño 3			
1	Raspberry pi 3B +	54,99€	54,99€
1	Raspberry pi Sense Hat V2	39,95€	39,95€
1	Controlador de motores doble puente H – L298N	18,76€	18,76€
2	SKU-GS20205-03 JGA25-317 DC12V200RPM	15,25€	30,50€
1	Lidar 360º FHL-LD19	65,50€	65,50€
1	MPU-6050	1,40€	1,40€
1	Cámara Pi OV9281	24,24€	24,24€
1	Cámara Logitech C270	24,38€	24,38€
	Consumibles y materias primas: filamento, tornillos, cable…	35€	35€
		TOTAL	294,72€
		TOTAL	441,13€

- Activos fijos

Tabla 6. Aparatos de medida y prueba

APARATOS DE MEDIDA Y PRUEBA			
Cantidad	Descripción	Precio/Unid	Subtotal
1	Galga extensiométrica y modulo ADC HX711	6,99€	6,99€
1	Sensor de corriente ACS712	5,99€	5,99€
1	Multímetro UNI-T UT61E	79,99€	79,99
1	Multímetro Pinza UNI-T UT210E	48,99€	48,99€
1	Generador de Funciones JUNTEK JDS-2900	114,61€	114,61€
1	Osciloscopio HANTEK K6022BL	75,99€	75,99€
1	Fuente de Alimentación KIPRIM DC310S	109,99€	109,99€
		TOTAL	442,55€

Tabla 7. Maquinaria usada

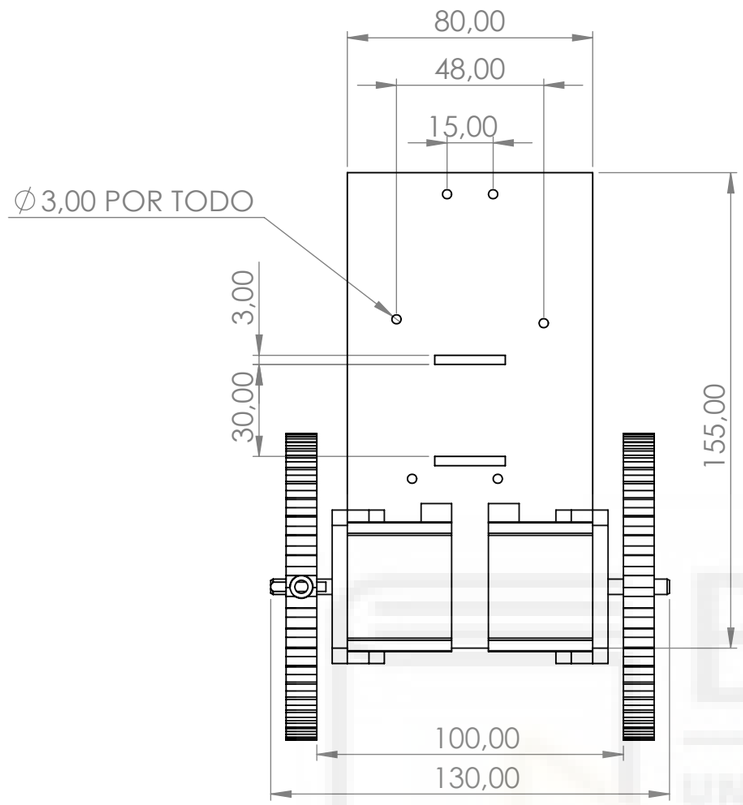
MAQUINARIA USADA			
Cantidad	Descripción	Precio/Unid	Subtotal
1	Impresora 3D Anycubic Kossel Lineal Plus	176,99€	176,99€
1	CNC TWOTREES TTC450	379,30€	379,30€
1	Soldador TS101	94,99€	94,99€
		TOTAL	711,28

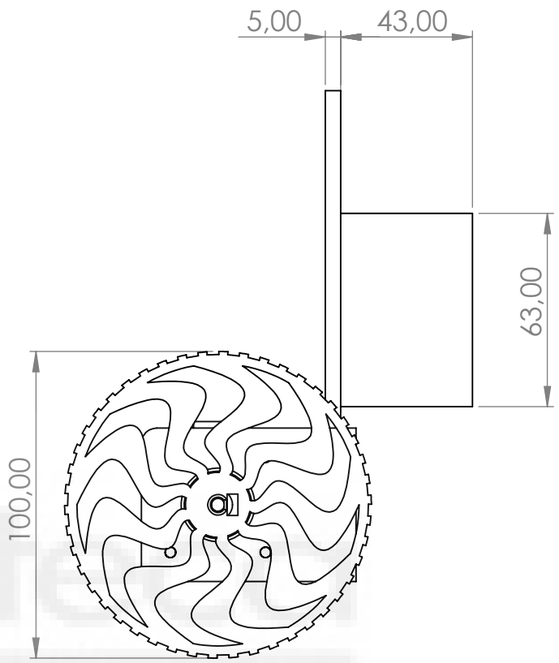
- Coste total

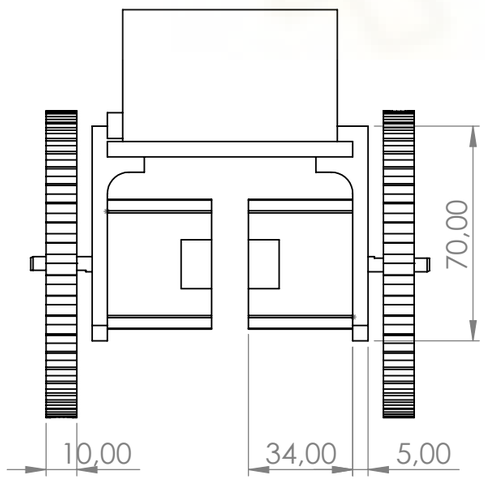
Tabla 8. Coste total

Gastos en material para el robot	441,13€
Activos fijos	1153,83€
TOTAL	1594,96

9 PLANOS







UNIVERSITAS Miguel Hernández

SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM

GRADO: Ingeniería Electronica y Automática Industrial

AUTOR: Juan Miguel Oñate

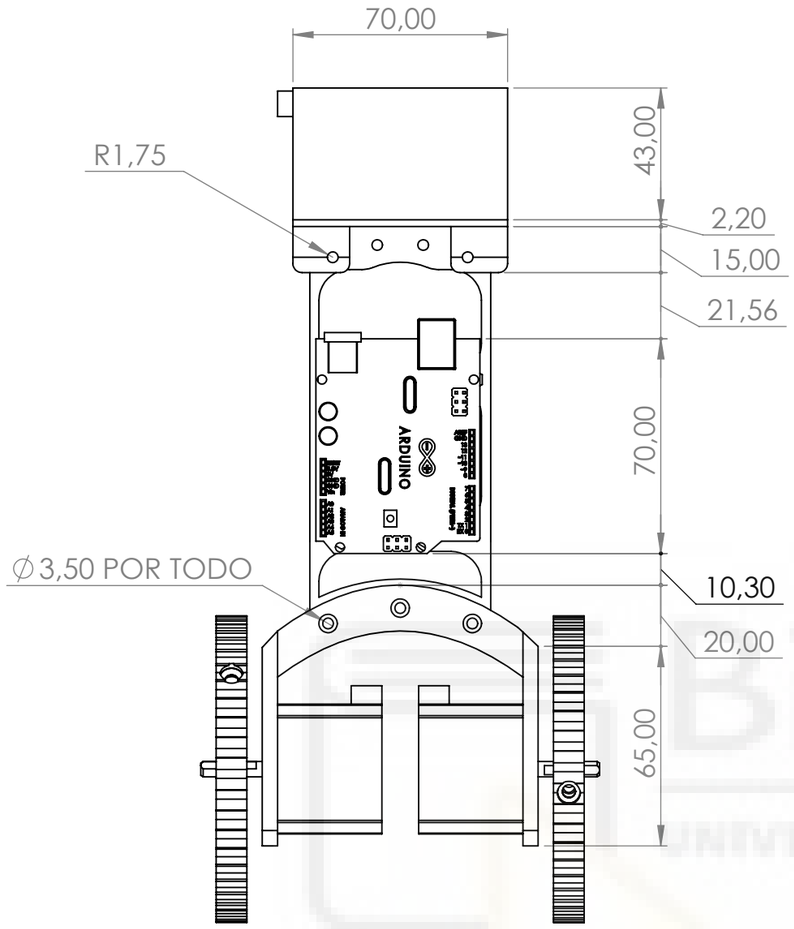
TÍTULO: Vehículo De Guiado Autonomo (AGV)

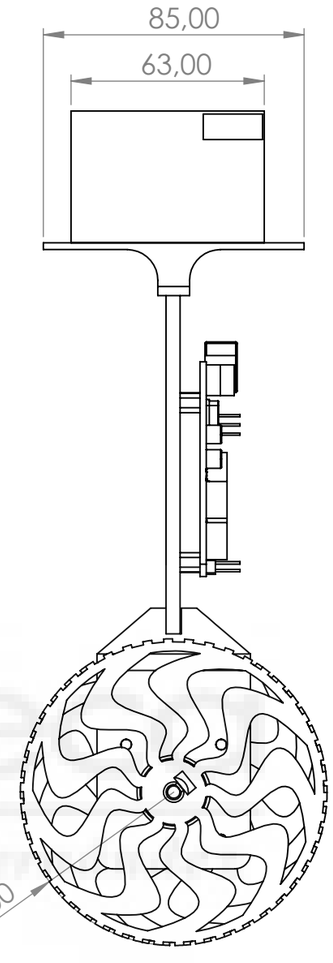
N.º DE DIBUJO: Plano Diseño 1

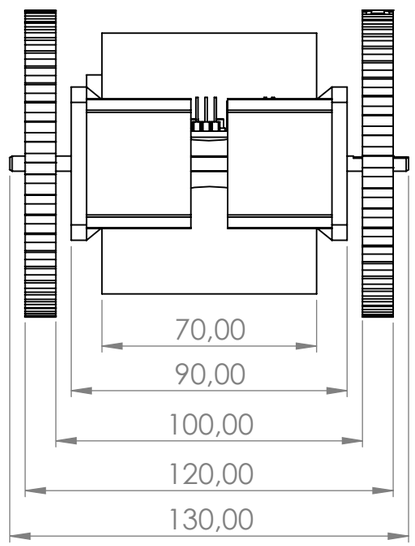
A3

ESCALA: 1/2

HOJA 1 DE 4







UNIVERSITAS Miguel Hernández

SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM

GRADO: Ingeniería Electronica y Automática Industrial

AUTOR: Juan Miguel Oñate

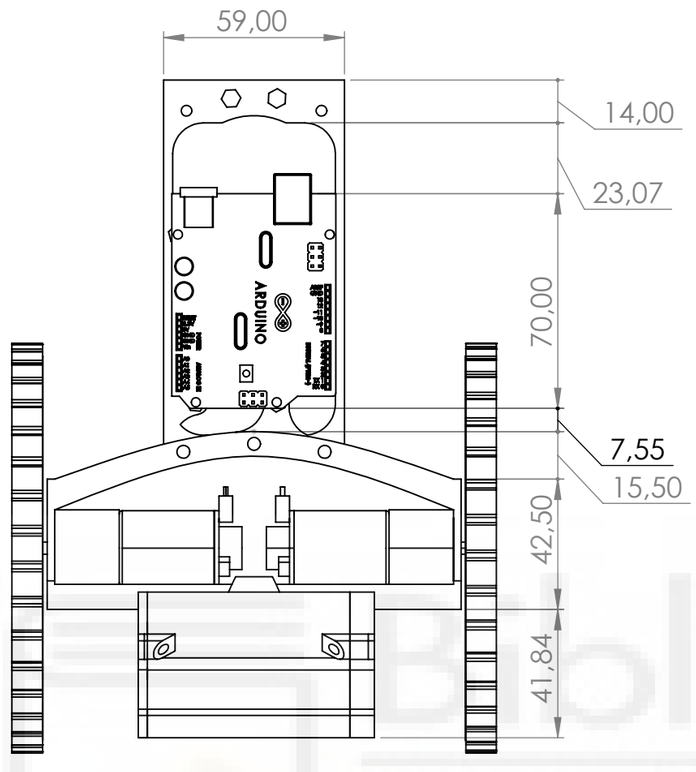
TÍTULO: Vehículo De Guiado Autonomo (AGV)

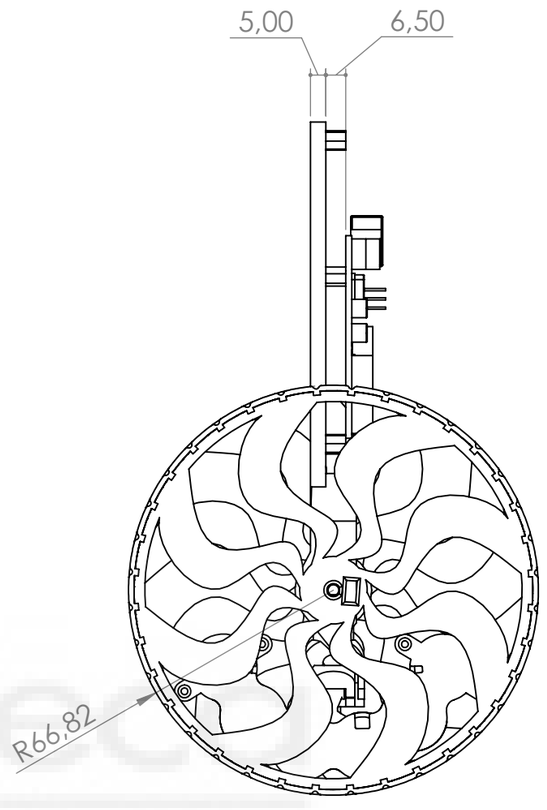
N.º DE DIBUJO: Plano Diseño 2

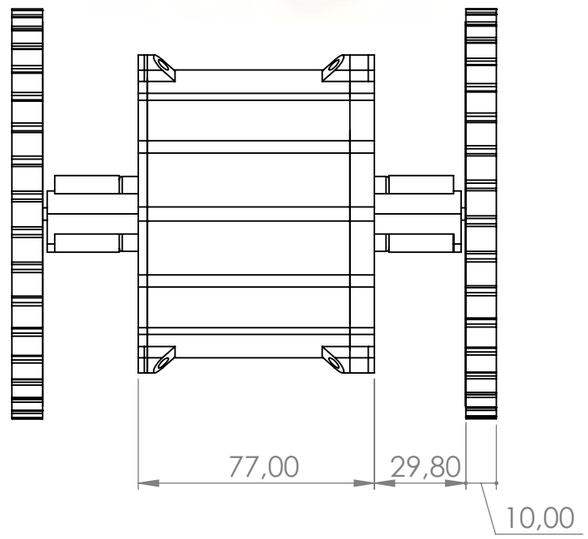
A3

ESCALA: 1/2

HOJA 2 DE 4







UNIVERSITAS Miguel Hernández

SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM

GRADO: Ingeniería Electronica y Automática Industrial

AUTOR: Juan Miguel Oñate

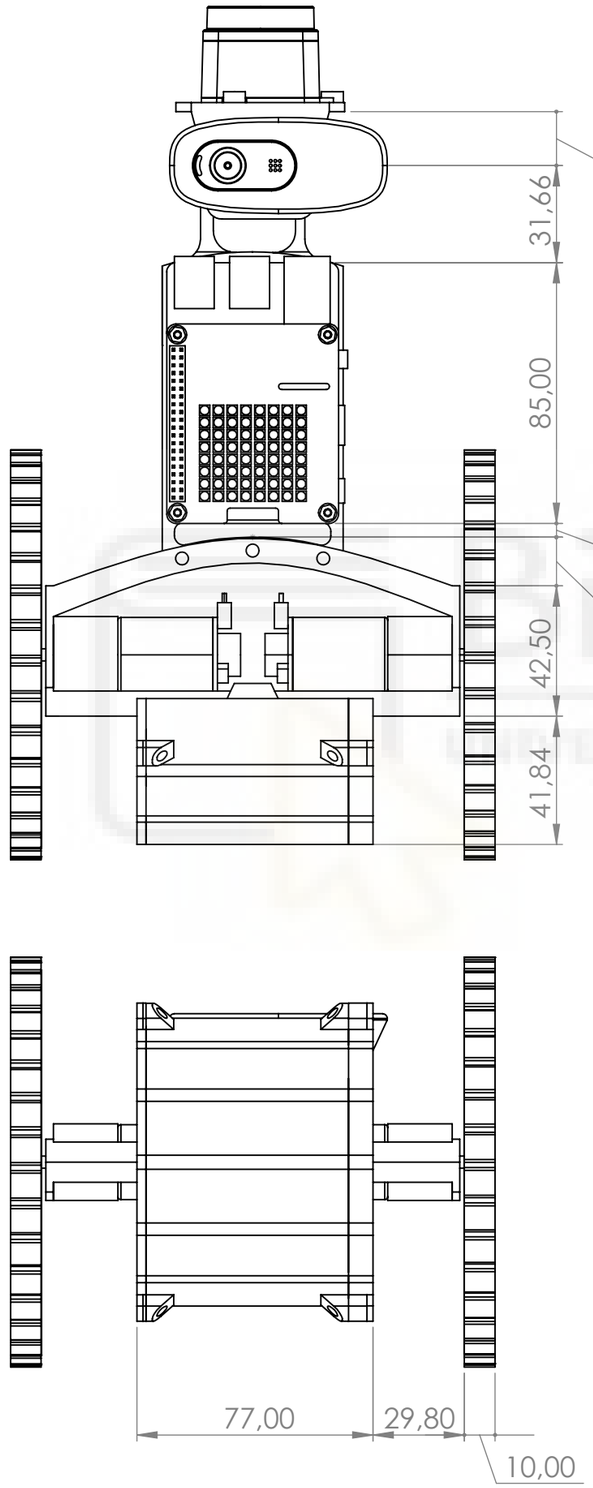
TÍTULO: Vehículo De Guiado Autonomo (AGV)

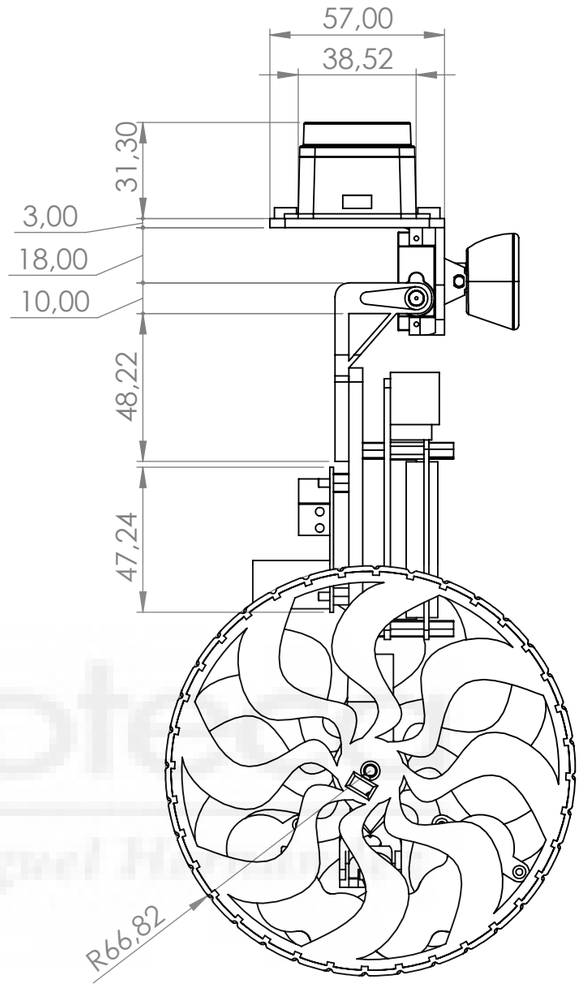
N.º DE DIBUJO: Plano Diseño 3

A3

ESCALA: 1/2

HOJA 3 DE 4





UNIVERSITAS Miguel Hernández SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM	GRADO: Ingeniería Electronica y Automática Industrial	TÍTULO: Vehículo De Guiado Autonomo (AGV)	
	AUTOR: Juan Miguel Oñate	N.º DE DIBUJO Plano Diseño 4	A3
		ESCALA: 1/2	HOJA 4 DE 4