UNIVERSIDAD INTERNACIONAL DEL ECUADOR

FACULTAD DE CIENCIAS TÉCNICAS

ESCUELA DE INGENIERÍA MECATRÓNICA

PLATAFORMA MÓVIL CON TRACCIÓN DIFERENCIAL Y CONTROL DE VELOCIDAD BASADO EN PID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

ANDRE SEBASTIAN ESCALANTE ZABALA

DIRECTOR: ING. VERÓNICA GREFA, MSc

D. M. Quito,

2020

DECLARACIÓN

Yo, Andre Sebastian Escalante Zabala, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que se han investigado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Internacional del Ecuador, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por normativa institucional vigente.

Andre Sebastian Escalante Zabala

C.I. 171801902-7

CERTIFICACIÓN

El proyecto de investigación "PLATAFORMA MÓVIL CON TRACCIÓN DIFERENCIAL Y CONTROL DE VELOCIDAD BASADO EN PID". fue desarrollado por el Sr. ANDRE SEBASTIAN ESCALANTE ZABALA ha sido debidamente revisado y está en condiciones de ser entregado para que siga lo dispuesto por la Facultad de Ciencias Técnicas, correspondiente a la sustentación y defensa del mismo.

ING. VERÓNICA GREFA, MSc

DIRECTOR DE PROYECTO

Agradecimientos

Agradezco a mis padres quienes con su amor, paciencia y esfuerzo me han permitido llegar a cumplir hoy un sueno mas, gracias por inculcar en mi el ejemplo de esfuerzo y valentia, de no temer las adversidades porque Dios esta conmigo siempre.

A mi hermana y mi tía por su carino y apoyo incondicional, durante todo este proceso, por estar conmigo en todo momento gracias. A toda mi familia porque con sus oraciones, consejos y palabras de aliento hicieron de mi una mejor persona y de una u otra forma me acompanan en todos mis suenos y metas.

Finalmente quiero dedicar esta tesis a todos mis amigos, por apoyarme cuando más los necesito, por extender su mano en momentos difíciles.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

PLATAFORMA MÓVIL CON TRACCIÓN DIFERENCIAL Y CONTROL DE VELOCIDAD BASADO EN PID

1. Tema

El tema del proyecto planteado es el diseño y la implementacion de una plataforma móvil con tracción diferencial y control de velocidad basado en PID.

2. Objetivos

2.1. General

Diseñar e implementar una plataforma móvil con tracción diferencial y control de velocidad basado en PID.

2.2. Específicos

- Investigar y documentar el uso de control PID para la velocidad en motores DC.
- Diseñar y mejorar los elementos mecánicos de la plataforma móvil con tracción diferencial.
- Seleccionar los mecanismos, partes y componentes de la plataforma móvil.
- Implementar el sistema operativo ROS para la navegación remota.
- Ejecutar un protocolo de pruebas de funcionamiento de la plataforma móvil.

3. Problema

El control eficaz de una plataforma móvil depende del control y de la comunicación a distancia. Para que una plataforma móvil pueda explorar satisfactoriamente un entorno debe incluir un control adecuado para su correcto movimiento y el acoplamiento de tecnología para la comunicación inalámbrica; así también, los dispositivos de potencia para el control de los motores.

4. Hipótesis

La plataforma móvil con tracción diferencial cuenta con un sistema PID para el control de la velocidad en sus cuatro motores. Se utiliza el sistema operativo ROS para la navegación remota lo que permite una navegación en diferentes tipos de terrenos.

5. Estudio teórico de plataformas móviles

5.1. Plataformas móviles

Las plataformas móviles son capaces de trasladarse en ambientes con obstáculos mediante diferentes métodos de control, se relacionan con los robots móviles debido a que son capaces de procesar la información de su comportamiento de su entorno como velocidad, posición, evasión de obstáculos mediante la interacción de sensores y patrones. Para el desplazamiento de las plataformas móviles se pueden emplear diferentes sistemas dependiendo del medio que exploran, pueden clasificarse de diversas maneras según se observa en la Figura 1.

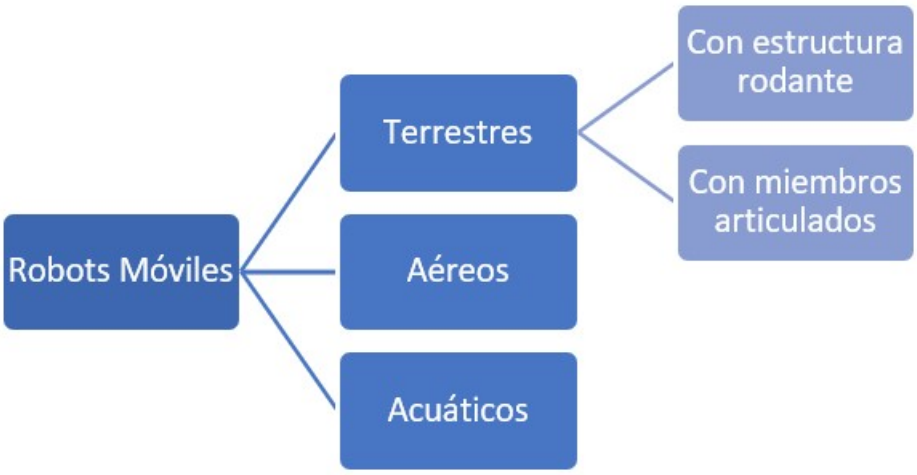

Figura 1. Clasificación de robots móviles

5.2. Tipos de desplazamiento

Dentro de los robots móviles terrestres con estructura rodante se puede encontrar la división por mecanismos de traslación como:

- Desplazamiento por una rueda motora única.

- Desplazamiento diferencial.
- Desplazamiento por ruedas sincrónicas.
- Desplazamiento Ackermann.

Desplazamiento por una rueda motora única

El desplazamiento por una rueda motora única es un concepto simple que se caracteriza por el control de la trayectoria en una sola rueda. Se configura el ángulo de dirección en una rueda para conseguir la trayectoria deseada. Se puede observar el funcionamiento en la Figura 2.

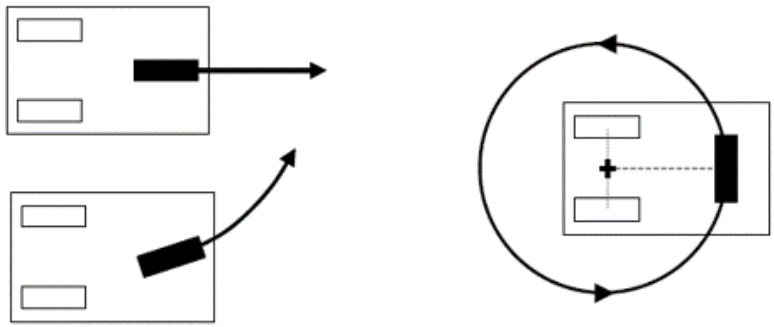

Figura 2. Funcionamiento de una rueda motora única [1]

Desplazamiento diferencial

El desplazamiento diferencial consta de dos motores para el control independiente de dos ruedas, se puede realizar con tres ruedas dado que esos tres puntos de contacto son los requerimientos mínimos para que se estabilice, esta tercera rueda puede ser una rueda pivote o giratoria. Para lograr el desplazamiento de trayectorias, especialmente curvas, se puede controlar el giro sobre su propio eje, como se indica en la parte derecha de la Figura 3, o aumentando la velocidad de una rueda y disminuyendo la velocidad de la otra según el giro que se desee como se puede observar en la parte izquierda de la Figura 3.

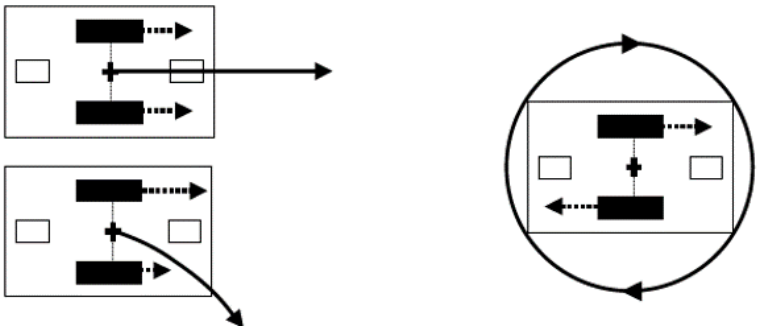

Figura 3. Funcionamiento del desplazamiento diferencial [1]

Desplazamiento por ruedas sincrónicas

Esta forma de desplazamiento tiene semejanza entre los dos desplazamientos anteriores dado que se van a controlar todas las ruedas a la vez, tanto en su velocidad como en el sentido de giro. Esto quiere decir que en el caso de que se tengan tres ruedas, las tres girarán juntas un mismo ángulo y de esta manera se puede realizar su trayectoria. Este desplazamiento tiene la ventaja de que se puede conducir en cualquier dirección deseada casi inmediatamente, cuando se quiera cambiar el giro del robot este deberá detenerse y realinear sus ruedas. En la Figura 4 se puede encontrar el funcionamiento de este tipo de desplazamiento.

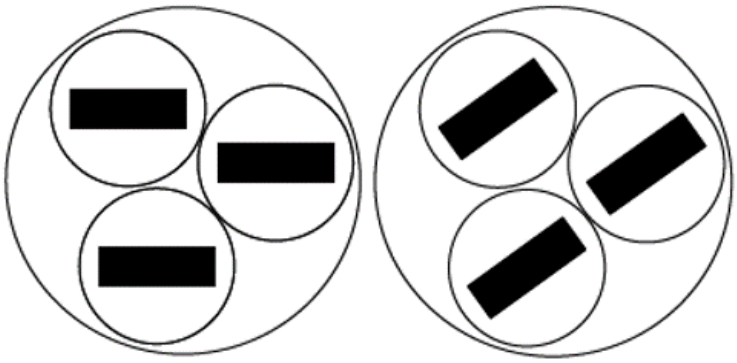

Figura 4. Funcionamiento por ruedas sincrónicas [1]

Desplazamiento Ackermann

Este desplazamiento es similar al que se encuentra en un vehículo, teniendo de esta manera dos ruedas traseras combinadas y dos ruedas delanteras combinadas. Así no se genera un problema como en el desplazamiento diferencial o un desplazamiento por ruedas sincrónicas dado que al estar combinadas las ruedas delanteras o las traseras entre si no

se presenta una diferencia de velocidad porque son accionadas por un único motor en comparación con las anteriormente mencionadas. El control de velocidad se aplica sobre el conjunto delantero y trasero, mientras que la dirección se controla únicamente sobre las ruedas delanteras como se presenta en la Figura 5.

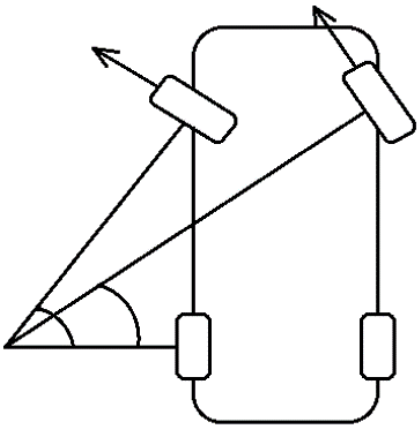

Figura 5. Funcionamiento Ackerman

5.3. Motores DC y su funcionamiento

Los motores de corriente continua son máquinas que convierten energía eléctrica en energía mecánica mediante un movimiento rotatorio. Estos motores se encuentran compuestos por el rotor y el estator principalmente, donde el rotor es la parte móvil y el estator es la parte fija del motor. En la Figura 6 se puede observar las dos partes principales del motor, donde el estator contiene un electroimán que produce un campo magnético que induce una fuerza sobre el rotor.

Figura 6. Motor DC [2]

Modelo matemático de un motor DC

El siguiente modelo matemático del motor DC posee características eléctricas como el voltaje de alimentación del rotor V_i, la corriente del rotor I_i, la resistencia del bobinado del rotor R_i, la inductancia del rotor L_i, la fuerza contra-electromotriz ϵ, el voltaje de alimentación del estator V_f, la corriente del estator V_f, la resistencia del bobinado del estator R_f y la inductancia del estator L_f. A esto se le suma el comportamiento dinámico como lo es la velocidad angular de giro ω, el momento de inercia en el eje J y el coeficiente de rozamiento viscoso B [3].

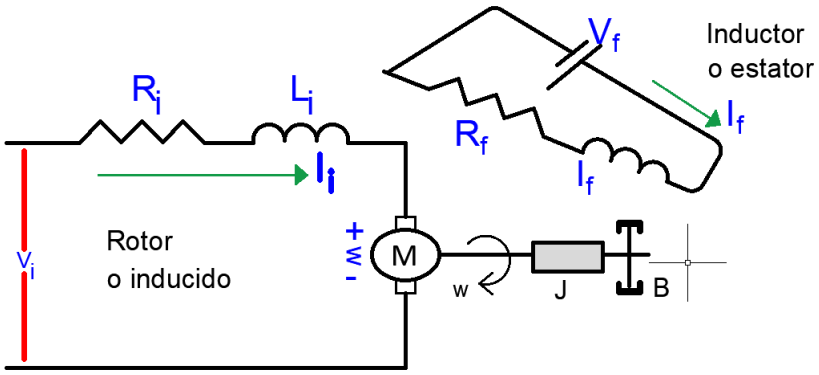

Figura 7. Esquema de un motor DC [3]

Se puede observar la parte mecánica y la parte eléctrica por lo que se debe utilizar una ecuación para cada una de estas partes; la primera, basada en la ley de la dinámica y la ley de Kirchoff.

En (1), se modela el movimiento del rotor que interviene en la parte mecánica y en (2) se modela el circuito eléctrico. Estas ecuaciones se relacionan en (3) debido a que la corriente que circula por el rotor induce una fuerza contra electromotriz, la misma que se opone a la causa que la produce, como se menciona en la ley de Lenz.

$$\epsilon = K_b \cdot \omega \tag{1}$$

Donde

ϵ fuerza electromotriz, en V;

K_b constante de la fuerza ccontra electromotriz, en V/rad/s;

ω velocidad angular a la que trabaja el motor, en RPM.

$$V_i - \epsilon = R_i \cdot I_i + L_i \cdot \frac{dI_t}{dt} \tag{2}$$

Donde

V_i tensión de alimentación del motor, en V;

R_i resistencia del bobinado del estator, en Ω;

I_i corriente de armadura, en A;

L_i inductancia del bobinado del rotor, en H.

Reemplazando (1) en (2):

$$V_i - K_b \cdot \omega = R_i \cdot I_i + L_i \cdot \frac{dI_t}{dt} \tag{3}$$

El rotor realiza el movimiento mediante el torque electromagnético τ_e, el mismo que es generado por el campo magnético que produce el estator, por lo que se relaciona con la corriente que circula en el motor, de esta manera (4) representa el torque electromagnético.

$$\tau_e = K_p \cdot I_i \tag{4}$$

Donde

τ_e torque electromagnético, en N·m;

K_p constante de torque electromagnético, en N·m/A.

El movimiento del motor en función de los torques que genera se representa en la Figura 8, donde se puede observar τ_e, τ_c y τ_f.

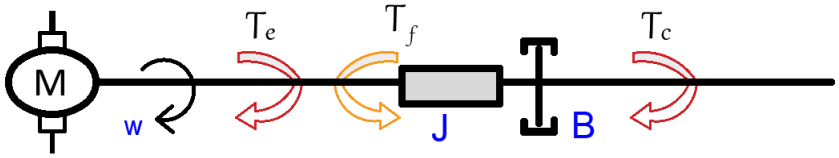

Figura 8. Diagrama de torques del motor DC [3]

La ecuación (5) describe τ_c:

$$\tau_c = J \cdot \frac{d\omega(t)}{dt} \tag{5}$$

Donde

τ_c torque de carga, en N·m;

J momento de inercia equivalente del eje rotor con la carga a colocar, en kg·m^2.

La (6) describe τ_f:

$$\tau_f = B \cdot \omega(t) \tag{6}$$

Donde

τ_f torque de fricción, en N·m;

B coeficiente de rozamiento viscoso, en kg·m^2·s.

Mediante (7) se obtiene la sumatoria de torque en (8).

$$\sum \tau = J \cdot \omega(t) \tag{7}$$

$$\tau_e - \tau_f = \tau_c \tag{8}$$

Reemplazando (4), (5), (6), (7), (8) se despeja I_i para obtener (9). Esta se deriva con respecto al tiempo y se obtiene (10).

$$I_i = \frac{B \cdot \omega(t) + J \cdot \frac{d\omega(t)}{dt}}{K_p} \tag{9}$$

$$\frac{dI_i}{dt} = \frac{B \cdot \frac{d\omega(t)}{dt} + J \cdot \frac{d\omega^2(t)}{dt}}{K_p} \tag{10}$$

Las ecuación (9) y (10) se reemplazan en (3) para obtener el modelo matemático de un motor DC. Esta es una ecuación diferencial de segundo orden no homogénea, lineal y de coeficientes constantes como se puede observar en (11).

$$V_i - K_b \cdot \omega(t) = R_i \cdot \frac{B \cdot \omega(t) + J \cdot \frac{d\omega(t)}{dt}}{K_p} + L_i \cdot \frac{B \cdot \frac{d\omega(t)}{dt} + J \cdot \frac{d\omega^2(t)}{dt}}{K_p} \tag{11}$$

En (11) se puede observar el modelo matemático de un motor de corriente continua, esta ecuación se la utiliza para anticipar los comportamientos de la máquina eléctrica.

5.4. Tipos de plataformas móviles

En el mercado se encuentran diferentes tipos de plataformas móviles, las mismas que son utilizadas de diversas maneras. En la Tabla 1 se encuentran las características de las plataformas móviles presentes en el mercado.

Tabla 1. Tipos de plataformas móviles

Robot	Longitud [mm]	Ancho [mm]	Altura [mm]	Masa [kg]	Velocidad [m/s]
Seekur Jr [14]	425	663	494	77	1,2
MMP-40 [15]	686	529	184	18	0,73
LT2 [16]	686	431	178	20	1,52
MegaBot Mobile [17]	787	660	355	80	3,35
JaguarLite [5]	640	538	176	15	2
Packbot	686	406	178	11	1,8
Talon [6]	864	572	279	39	2
Summit-XL Steel [18]	663	847	509	105	3
Summit-XL [4]	722	613	220	45	3
Promedio	**684,33**	**584,33**	**285,89**	**45,39**	**2,067**

5.5. Aplicaciones de las plataformas móviles

Las plataformas móviles tienen diversas aplicaciones, entre las más comunes se encuentran las de operación de rescate, mapeos de terrenos, navegación autónoma, transporte de carga, video vigilancia, etc. Son diseñadas para que sean lo más livianas y compactas posibles potenciando su movilidad sin limitarse a terrenos como se pueden observar en las Figuras 9, 10, 11, 12, 13, 14, 15.

Figura 9. Prototipo Summit [4]

Figura 10. Summit [4]

Figura 11. Summit [4]

Figura 12. Summit [4]

Figura 13. Jaguar [5]

Figura 14. Jaguar [5]

Figura 15. Talon [6]

5.6. Sistemas de Control

Los sistemas de control han asumido un papel importante en el desarrollo y avance de la tecnología, se puede decir que todas las actividades se encuentran influenciadas por algún tipo de sistema de control. Estos se encuentran en todos los sectores de la industria, se pueden clasificar por dos tipos: los de lazo abierto y de lazo cerrado.

Por su parte, los sistemas de control de lazo abierto son mejores en términos económicos; sin embargo, carecen de exactitud en comparación a los sistemas de lazo cerrado. La principal característica de los sistemas de lazo cerrado es que poseen una realimentación que compensa la señal de salida, debido a que esta es comparada con la señal de entrada. La realimentación puede reducir los efectos de las perturbaciones, puede lograr que un sistema sea insensible a las variaciones del proceso y por último conseguir que un sistema siga fielmente sus señales de entrada.

En la Figura 16 se presenta un sistema de realimentación sencillo con un diagrama de bloques. El sistema se basa en dos grandes componentes, el proceso y el controlador. El proceso tiene una entrada, la cual se denota por u. La variable de control influye sobre el proceso a través de un actuador, que puede ser una válvula o un motor. La salida o mejor conocida como, ”variable de proceso” y la función que cumple la realimentación es aumentar la variable manipulada cuando el error es positivo y disminuirla cuando el error es negativo, a esto se lo conoce como realimentación negativa.

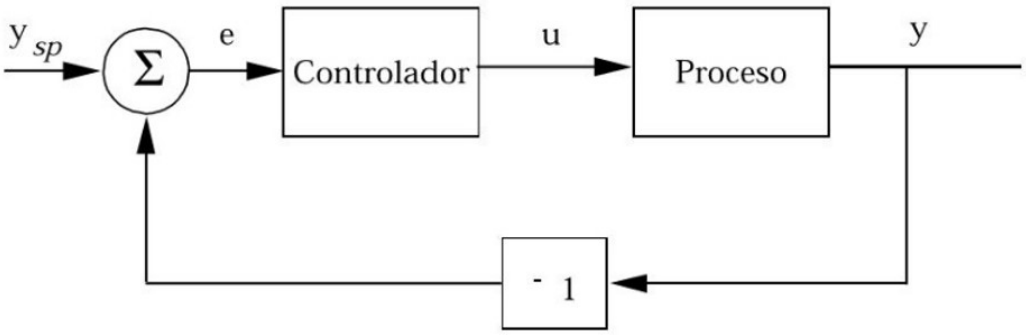

Figura 16. Sistema de control con realimentación [7]

Si la realimentación funciona bien, el error será pequeño e idealmente será cero. Cuando el error es pequeño la variable de proceso está también próxima a la referencia independientemente de las propiedades del proceso; por último, para conseguir la realimentación es necesario tener sensores y actuadores apropiados que efectúen las acciones de control.

5.7. Control PID

El controlador PID es un controlador realimentado cuyo objetivo es conseguir un error mínimo (próximo a cero), esto lo logra mediante sus tres fases a través de las acciones

proporcional que representa el tiempo presente, integral y derivativa [7], estas representan el tiempo presente, pasado y futuro por extrapolación lineal del error, respectivamente.

Se ha comprobado que el controlador PID es capaz de resolver un amplio espectro de problemas de control por el trayecto que ha tenido en la historia, su funcionamiento se ha desarrollado a lo largo de un período que se extiende por lo menos 250 años. Los primeros controladores fueron dispositivos mecánicos empleados para controlar molinos de viento y máquinas de vapor [7].

La acción proporcional es usada para generar una salida que es proporcional al error, obteniendo como resultado la multiplicación de la señal de error por una constante, la función de transferencia se puede observar en (12).

$$u(t) = K_p \cdot e(t) \tag{12}$$

Donde

$u(t)$ función de transferencia

K_p ganancia proporcional

$e(t)$ señal de error

De esta manera, mientras mayor sea la ganancia proporcional menor será el error y así se reduce el error hasta un valor o rango aceptable.

La acción integral ofrece una salida del controlador proporcional al error acumulado, esto reduce el fallo mediante la suma de pequeños errores para intentar corregirla logrando un porcentaje de inestabilidad. Se puede observar en (13) la función de transferencia, considerando K_i como ganancia integral.

$$u(t) = K_i \cdot \int_0^t e(t)dx \tag{13}$$

La acción derivativa es proporcional a la derivada de la señal de error. La acción derivativa del controlador reacciona a que tan rápido cambia la entrada con respecto al tiempo, alterando la señal de salida en proporción con la tasa de cambio de entrada, teniendo como resultado una función de transferencia que se puede observar en (14), se considera K_d como ganancia derivativa.

$$u(t) = K_i \cdot \frac{de(t)}{dt} \tag{14}$$

La función de esta acción es disminuir el fallo, este hecho corrige el error en el mismo tiempo que se produce con el fin de evitar su incremento. Es utilizada cuando el tiempo de retardo del elemento a controlar es considerable porque, si se utiliza en un elemento a controlar con demasiadas oscilaciones, generará una sensibilidad y complicaciones al momento del control.

Por lo tanto, en todo tipo de diseño es importante conocer las limitaciones fundamentales, usualmente estas son: dinámica del proceso, no linealidades, perturbaciones e incertidumbre del proceso.

La robustez es un aspecto clave frente a variaciones del proceso en el diseño de un sistema de control. Los parámetros del proceso pueden variar por muchas razones, por lo general dependen de las condiciones de operación. Los retardos de tiempo y las constantes de tiempo usualmente cambian con los niveles de producción.

Para esta plataforma se usará un controlador PID debido que permite un control en la velocidad de los motores, el control de velocidad actúa sobre los motores suministrando más energía si no se ha alcanzado la velocidad de referencia, o por el contrario, dejando de suministrar energía en el caso de haber superado esta velocidad.

De esta manera se logra mejorar fallos que se pueden encontrar fuera del sistema como pequeños obstáculos y finalmente la plataforma móvil tiene una velocidad que se realimenta y se compara con la velocidad de referencia logrando un error mínimo en la variación de velocidad.

6. Diseño Mecatrónico

6.1. Diseño de la plataforma móvil

A continuación se adjunta la información correspondiente a los diferentes componentes de la plataforma móvil.

El proyecto comprende el desarrollo de una plataforma cuya velocidad sea controlada mediante PID, este proyecto tiene diferentes aplicaciones debido a esto se adapta según las

necesidades del usuario. Mediante el análisis se obtiene las características necesarias para proceder con el diseño y construcción de la plataforma que se pueden observar mediante en el Anexo A.

Concluyendo con el análisis de la casa de la calidad se obtiene que la plataforma debe contar con las característcas técnicas que se detalla a continuación.

- Peso máximo 25kg
- Dimensiones máximas 700 mm x 600 mm x 300 mm
- Capacidad batería: 12 Vdc a 12 Ah
- Estructura modular
- Pendiente máxima: 25 grados
- Tiempo de funcionamiento 40 min
- Sistemas de suspensión independientes

Dimensionamiento mecánico

El dimensionamiento mecánico se lo realizó con la característica principal de que sea una plataforma capaz de desplazarse en superficies difíciles, como tierra y césped. Por lo que se dispuso que esta plataforma tenga una arquitectura fácil de modificar para que se puedan adecuar diferentes sensores como buscaminas, cámaras, etc.

Se consideró un diseño modular para poderlo reemplazar, modificar o reparar de forma ágil y rápida. La estructura consta de dos conjuntos que se dividen en el chasis y los sistemas de suspensión.

El material del chasis es perfil de aluminio 6065 tipo v de sección de 20×20 mm. Los sistemas de suspensión se dividen en dos partes, suspensión delantera y trasera, estas se constituyen en uniones en forma de U, las mismas que unen los amortiguadores con los perfiles de aluminio y bases que sostienen los motores. Todas estas piezas son de acero.

La plataforma cuenta con una caja plástica cubierta de $21 \times 15 \times 10$ cm para todos los sistemas eléctricos. La batería se encuentra en la parte exterior como se puede ver en la Figura 17, esta muestra la vista delantera, lateral y trasera.

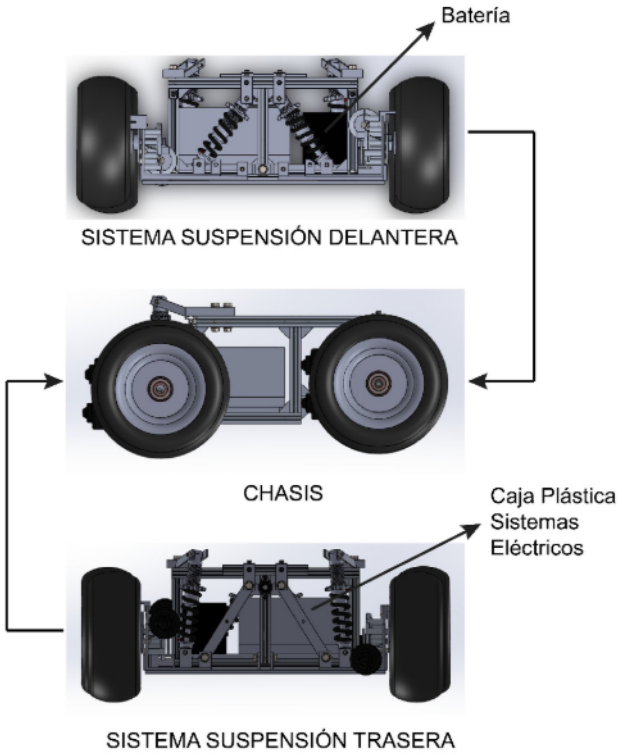

Figura 17. Suspensión delantera, chasis y suspensión trasera en la plataforma móvil

Sistemas de suspensión

El sistema de suspensión fue una de las principales mejoras en cuanto al diseño anterior, esta mejora se encarga de mantener las 4 ruedas en contacto con el suelo el mayor tiempo posible. En el coche se tienen 4 amortiguadores que se clasifican como sistema de suspensión delantera y sistema de suspensión trasera. Esto afecta principalmente el comportamiento de la plataforma en socavones leves, saltos, movimientos acelerados y desacelerados.

Suspensión delantera

La suspensión delantera es una suspensión independiente, similar a la que utilizan los autos de carreras que tienen control remoto. Este sistema forma un triángulo, en el cual el extremo superior del amortiguador está anclado al chasis de la plataforma y el otro extremo está sujeto a la base que soporta el motor. Para completar el triángulo, se junta una pletina a un perfil de aluminio; estas pletinas son las que guían en los diferentes movimientos para mejorar la estabilidad de la plataforma como se puede observar en la Figura 18.

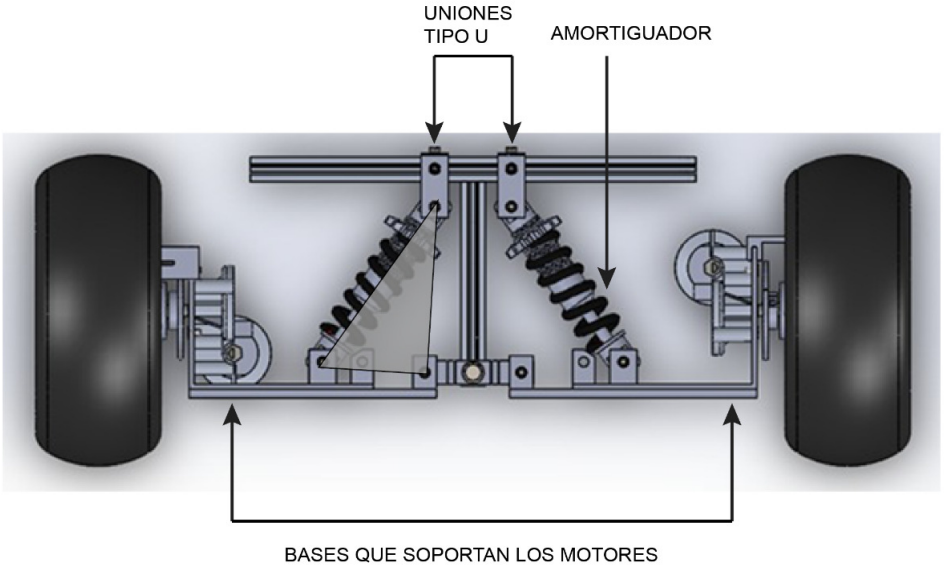

Figura 18. Suspensión delantera en la plataforma mòvil

Suspensión trasera

La suspensión trasera es una suspensión independiente, de tipo puntal, similar a la que utiliza el vehículo Grand Vitara. El extremo superior de un puntal está anclado en el chasis de la plataforma mediante un soporte de puntal. El extremo inferior del puntal está conectado a la base que soporta el motor y esta, está incorporada en una unidad con una platina a la unión tipo T; por consiguiente, mejora la estabilidad de la plataforma. Se puede visualizar la suspensión trasera en la Figura 19.

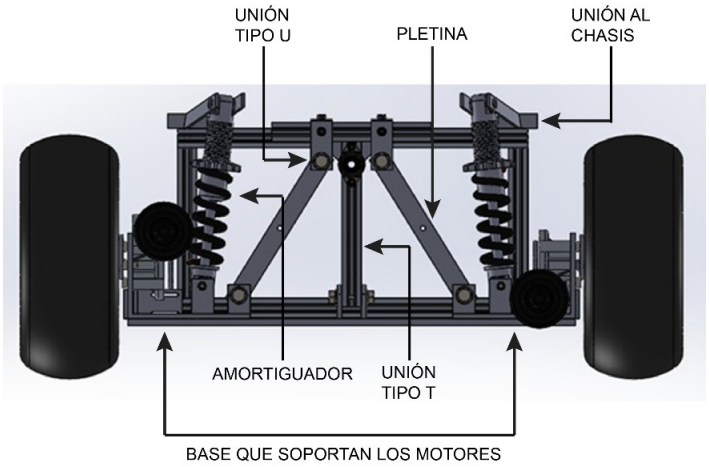

Figura 19. Suspensión trasera de la plataforma móvil

Análisis de deformación para la plataforma móvil

Para el análisis de deformación se toma en consideración el peso de 3,67 kg que se incorpora a través de las baterías y otros componentes extras, ubicados en la parte inferior resaltada en la Figura 20. Las baterías son el principal factor para el análisis.

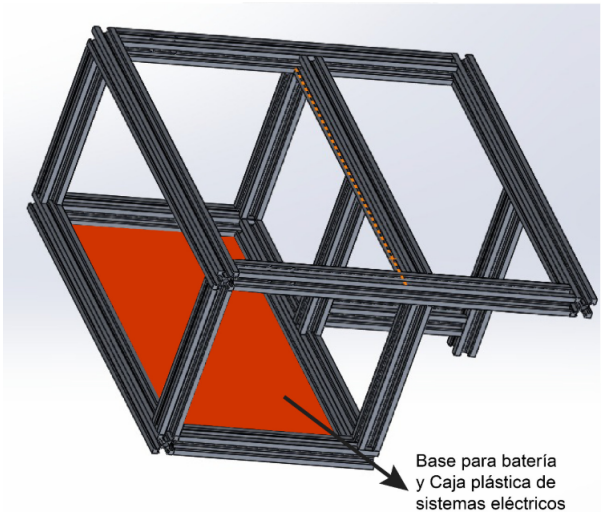

Figura 20. Chasis de la plataforma móvil y zona de control

Se analiza la deformación en el perfil para comparar y verificar dentro de qué grupo se encuentra según la deformación obtenida. En la Figura 21 se puede observar cómo el peso de las baterías y los componentes electronicos afecta con una fuerza F de 300 N en el perfil de aluminio, también se pueden observar los momentos y las reacciones.

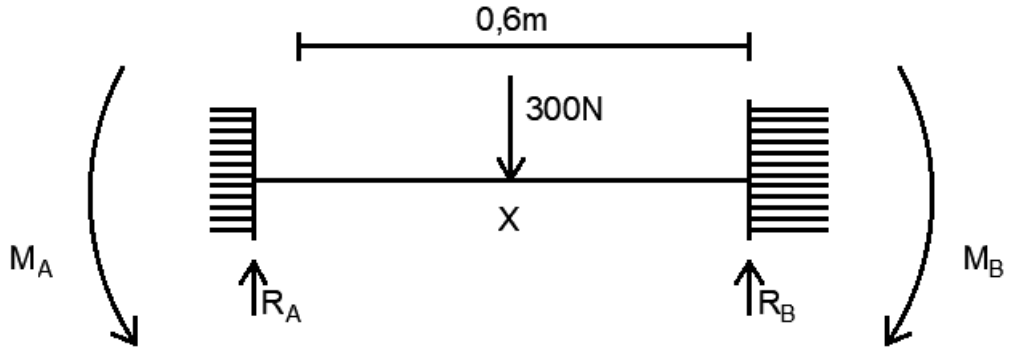

Figura 21. Diagrama de cuerpo libre en el perfil de aluminio

Para la resolución se realiza la sumatoria de fuerzas y de momentos como se observa en (15), (16), (17), (18), (19).

$$\sum F_y = 0 \tag{15}$$

$$R_A + R_B - 300\,[\mathrm{N}] = 0 \tag{16}$$

$$R_A + R_B = 300\,[\mathrm{N}] \tag{17}$$

Donde

$\sum F_y$ sumatoria de fuerzas en el eje Y;

R_A reacción en el soporte A, en N;

R_B reacción en el soporte B, en N.

$$\sum M = 0 \tag{18}$$

$$-M_A + M_B - 0,6\,[\mathrm{m}] \cdot R_B + 300\,[\mathrm{N}] \cdot 0,3\,[\mathrm{m}] = 0 \tag{19}$$

Donde

$\sum M$ sumatoria de momentos;

R_A momento en el soporte A, en N·m;

R_B momento en el soporte B, en N·m.

De manera general, se obtiene (20), la misma que integrando una vez se puede conseguir:

$$M = R_A x - M_A - 300 \cdot (x - 0{,}3) \tag{20}$$

A (20) se le integra una vez, obteniendo (21):

$$E_I\sigma = \frac{R_A x^2}{2} - M_A x - 300 \cdot \frac{(x-0{,}3)^2}{2} + C_1 \tag{21}$$

A (20) se le integra una segunda vez, obteniendo (22):

$$E_{Iy} = \frac{R_A x^3}{6} - \frac{M_A x^2}{2} - 300 \cdot \frac{(x-0,3)^3}{6} + C_1 x + C_2 \tag{22}$$

Para el valor de las constantes se realiza con operaciones iniciales o contables teniendo:

$$x_1 = 0 \Rightarrow \sigma = 0, y_1 = 0, C_1 = 0, C_2 = 0 \tag{23}$$

$$x_2 = 0{,}6 \Rightarrow \sigma = 0, y_2 = 0, C_1 = 0, C_2 = 0 \tag{24}$$

Donde

x_1 distancia en x en la reacción R_A, en m;

x_2 distancia en x en la reacción R_B, en m;

y_1 distancia en y en la reacción R_A, en m;

y_2 distancia en y en la reacción R_B, en m;

Con estos valores, se reemplaza en (22) y se procede a calcular los momentos y reacciones en las siguientes ecuaciones (25), (26) partiendo de que el punto en la mitad del perfil se considera el más crítico teniendo de (20) se obtiene (26) en términos del momento en A.

$$0 = \frac{0{,}6^2 \cdot R_A}{2} - 0{,}6 \cdot M_A - 300 \cdot \frac{(0{,}6-0{,}3)^2}{2} + 0 \tag{25}$$

$$R_A = (0{,}6 \cdot M_A + \frac{300 \cdot 0{,}3^2}{2}) \cdot \frac{2}{0{,}6^2} \tag{26}$$

De igual manera en (22) se sustituyen los valores y se ingresan los cálculos de (26) en (28) para encontrar (30) que será el recurso del momento M_A y luego se encuentra el valor

de las reacciones R_A.

$$0 = \frac{0{,}6^3 \cdot R_A}{6} - \frac{0{,}6^2 \cdot M_A}{2} - 300 \cdot \frac{(0{,}3 - 0{,}3)^3}{6} + 0 + 0 \tag{27}$$

$$0 = \frac{0{,}6^3}{6} \cdot \frac{2}{0{,}6^2} \cdot (0{,}6 \cdot M_A + \frac{300 \cdot 0{,}3^2}{2}) - \frac{0{,}6^2 \cdot M_A}{2} - 300 \cdot \frac{(0{,}3 - 0{,}3)^3}{6} \tag{28}$$

$$0 = 0{,}12 \cdot M_A + 2{,}7 - 0{,}18 \cdot M_A - 1{,}35 \tag{29}$$

$$M_A = \frac{2{,}7 - 1{,}35}{0{,}18 - 0{,}12} = 22{,}5 \tag{30}$$

$$R_A = \frac{27 \cdot 1}{0{,}6^2} = 150 \tag{31}$$

De esta manera se despejan los valores del momento en B y la reacción en B.

$$R_B = 300 - R_A \tag{32}$$

$$R_B = 150 \ [\mathrm{N}] \tag{33}$$

$$M_B = M_A + 0{,}6 \cdot R_B - 300 \cdot 0{,}3 \tag{34}$$

$$M_B = 22{,}5 \ [\mathrm{N \cdot m}] \tag{35}$$

Con estos valores se calcula la deformación mediante carga uniforme con (36).

$$Y_{max} = \frac{w \cdot l^4}{384 \cdot E \cdot I} \tag{36}$$

Donde

E módulo de elasticidad del material, en N/m^2;

l longitud, en m;

I inercia, en m^4.

$$Y_{max} = \frac{\frac{300}{0{,}6} \cdot 0{,}6^4}{384 \cdot 6{,}9 \times 10^{10} \cdot 1{,}33 \times 10^{-8}} \tag{37}$$

$$Y_{max} = 0{,}000184 \tag{38}$$

La comprobación de los cálculos se realiza mediante elementos finitos en el programa SolidWorks. Para la simulación por elementos finitos de la estructura de la plataforma móvil,

en primera instancia, se bosqueja el chasis, como se puede observar en la Figura 20.

Se seleccionaron las uniones de la plataforma como puntos fijos para colocar las cargas en la parte inferior del chasis, estimando una carga de 300 N como se puede observar en la Figura 22, para determinar la deformación en la viga y observar los puntos críticos. Se realiza un mallado y el análisis por elementos finitos para observar la deformación.

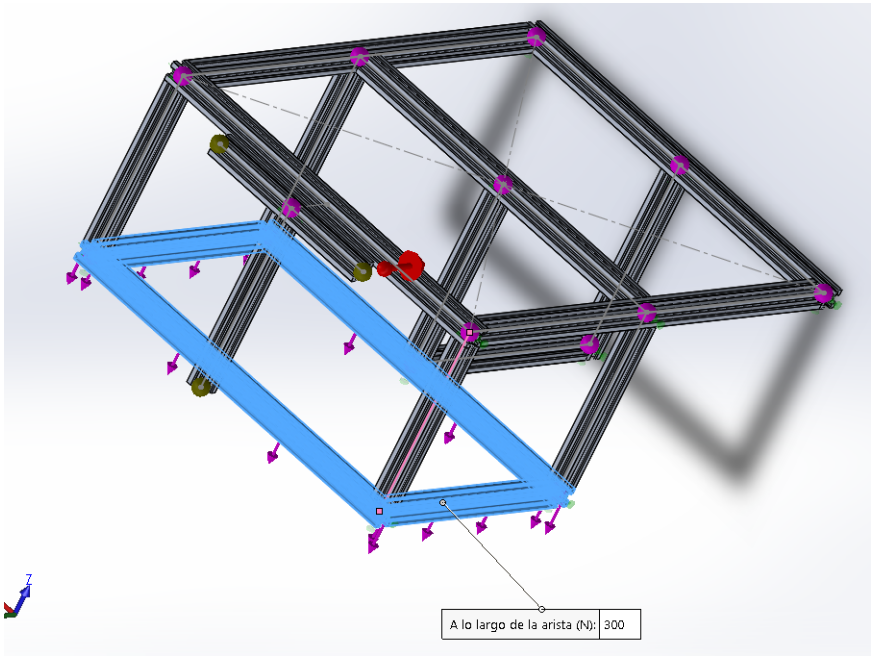

Figura 22. Cargas y sujeciones en la plataforma móvil

En la Figura 23 se puede observar el resultado del análisis de la simulación, se puede interpretar que el centro del perfil de aluminio tiene una deformación de 0,121 mm con una fuerza de 300 N.

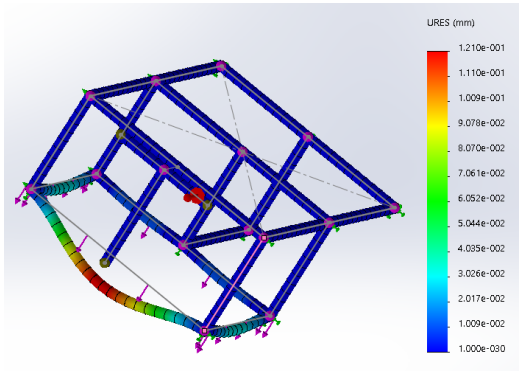

Figura 23. Análisis elementos finitos

Se puede concluir que la relación de la flexión máxima a la longitud de la viga se encuentra en el rango de precisión moderada tanto en el método analítico y por simulación

como se puede observar en la Tabla 2.

Tabla 2. Comparación metodo analítico y simulación FEA

	Método Analítico	Simulación FEA
Deformación	$1,84 \times 10^{-4}[\mathrm{m}]$	$1,21 \times 10^{-4}[\mathrm{m}]$

Es conveniente que se encuentren los valores de deformación en el rango de precisión moderada por el uso que la plataforma móvil va a tener, esta estructura se va a desplazar y va a tener perturbaciones, de esta manera se puede comprobar que la estructura de la plataforma no resulta afectada.

Dimensionamiento de motor

Para el dimensionamiento de los motores se utiliza el peso como principal factor ya que los motores deben ser capaces de desplazar su masa en terrenos que no necesariamente tengan que ser blandos. Al estar las ruedas en contacto directamente con el suelo se genera una resistencia a la rodadura, en la Tabla 3 se encuentran los coeficientes para algunos terrenos.

Tabla 3. Coeficientes de resistencia a la rodadura [12]

Superficie de Contacto	Coeficiente de resistencia a la rodadura
Concreto o Asfalto	0,013
Grava Apisonada	0,02
Tarmac	0,025
Camino sin Pavimentar	0,05
Césped, tierra y arena	0,1-0,35

Previamente, en la Tabla 1 se analizaron las diferentes plataformas disponibles y se obtuvieron las características más importantes, después del análisis mediante la casa de la calidad que se puede observar en el Anexo A se obtuvieron las siguientes características:

- Longitud = 648 mm
- Ancho = 584 mm
- Altura = 285 mm
- Masa = 45,93 kg

- Velocidad = 2 m/s

Con la velocidad estimada se puede obtener la velocidad angular dado ya que se tiene el diámetro de la rueda que se va a usar de 25 cm.

$$v = \omega \cdot R \tag{39}$$

Donde

v velocidad, en m/s;

ω velocidad angular, en Rad/s;

R radio de la rueda, en m.

De la ecuación anterior se despeja ω y se obtiene la velocidad angular

$$\omega = 16 \ \left[\frac{\text{Rad}}{\text{s}}\right] \tag{40}$$

$$\omega = 152,8 \ [\text{RPM}] \tag{41}$$

La potencia del motor se obtiene mediante (42).

$$P = \tau \cdot \omega \tag{42}$$

Donde

P potencia del motor, en W;

τ torque, en N·m.

La masa de la plataforma se encuentra repartida en diferentes partes, las mismas que se pueden observar de manera detallada en la Tabla 4.

Tabla 4. Masa de la plataforma móvil

Descripción	Cantidad	Peso Unitario [kg]	Peso total [kg]
Chasis	1	2,86	2,86
Llantas	4	2	8
Soporte con motores	4	1,75	7
Amortiguadores	4	0,4	1,6
Batería	1	3,37	3,37
Componentes electrónicos	1	0,3	0,3
Carga deseada	1	30	30
Total			**53,13**

En total se tiene una masa de 23,13 kg solo de la plataforma y a eso se le suma la carga deseada de 30 kg (se le desea agregar esta carga para transportar componentes o debido a su estructura modular acoplarle elementos o sensores y que estos no influyan en el rendimiento de la plataforma móvil) y al multiplicar por la gravedad se obtiene:

$$w = 521,2 \ [\mathrm{N}] \tag{43}$$

Con (44) se obtiene la fuerza de la resistencia de la rodadura.

$$F_r = w \cdot c \tag{44}$$

Donde

F_r fuerza de la resistencia a la rodadura, en N;

c coeficiente de resistencia a la rodadura.

En este caso se seleccionó el coeficiente de rodadura para terrenos de césped, tierra o arena que es de 0,35 de la Tabla 3 y con el peso de la plataforma se obtiene:

$$F_r = 182,42 \ [\mathrm{N}] \tag{45}$$

Con esta fuerza se podrá desplazar la plataforma, para el torque del motor y se calcula mediante el peso de la misma con (46).

$$\tau = m \cdot g \tag{46}$$

Donde

m masa, en kg;

g gravedad, en m/s^2.

Se utiliza esta fuerza y se multiplica por el radio de la palanca donde ejercerá ese torque como se observa en (47).

$$\tau = F_r \cdot R \tag{47}$$

De la cual conociendo que el radio es 0,125 m y la fuerza F_r es 182,42 N se obtiene:

$$\tau = 22 \tag{48}$$

Como se utilizan cuatro motores para el desplazamiento de la plataforma se divide el torque para 4 obteniendo un torque de 5,5 N·m y para encontrar la potencia, se usa (27) que la velocidad angular es 16 rad/s, esto se reemplaza en (28) y se obtiene que:

$$P = 80 \ [\mathrm{W}] \tag{49}$$

Obteniendo como resultado la Tabla 5 que indica las características del motor a utilizar.

Tabla 5. Características del Motor

Potencia	80 W
Torque	5,5 N·m
Velocidad	152,8 RPM
Cantidad	4

6.2. Diseño Electrónico

El diseño electrónico comprende todos los elementos que se utilizan para la parte de control, comunicación y potencia para el funcionamiento de la plataforma móvil. Se puede observar el diagrama de bloques en la Figura 24.

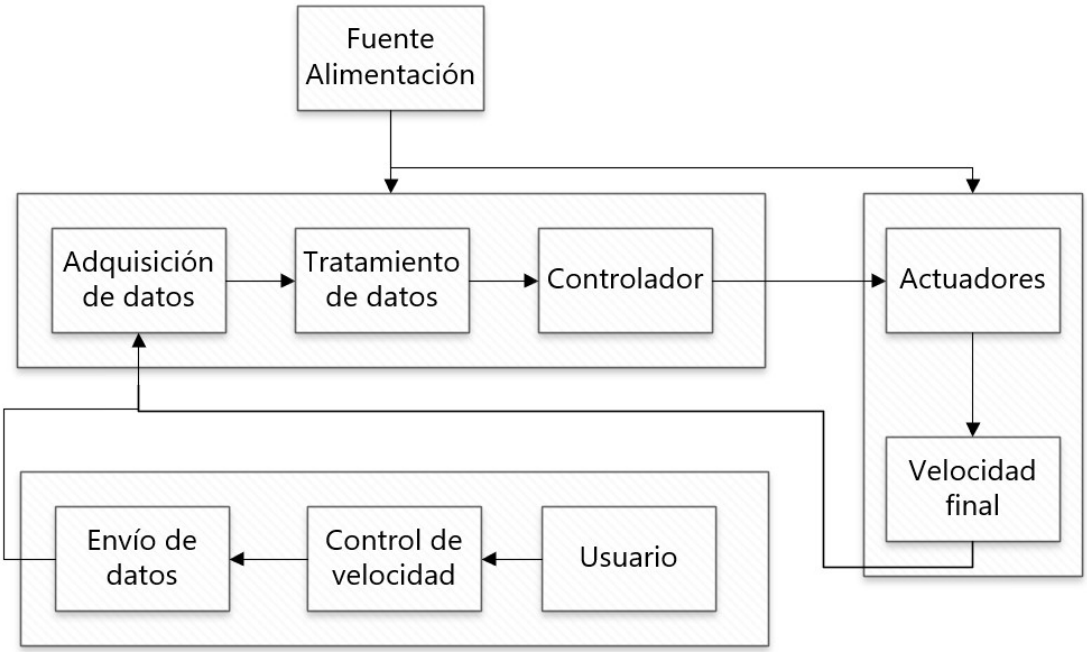

Figura 24. Diagrama de bloques

En este diagrama de bloques se observa que la plataforma móvil cuenta con un controlador principal, un controlador de potencia para los motores, un sensor de medición de velocidad por motor, una fuente regulada, un módulo de comunicación inalámbrica para el envío y recepción de datos, un dispositivo de entrada que actúa como control remoto y una batería para la alimentación de los componentes.

Selección de componentes

Controlador Principal

Dentro de la selección de componentes se seleccionó la tarjeta de control Arduino Mega por su bajo costo, su disponibilidad en el mercado y sus protecciones al ruido, esta tarjeta controladora cumple con el análisis de entradas y salidas detalladas en la Tabla 6, dependiendo de este análisis se puede escoger dentro de la gama de Arduino algún modelo en específico.

Tabla 6. Análisis de entradas y salidas del sistema

Acción	Entradas	Salidas	Observación
Control de velocidad de los cuatro motores	0	4	Señal PWM
Control de dirección de los cuatro motores	0	8	Señal digital
Encoder, recepción de la señal del motor (Frecuencia)	4	0	Señal digital por interrupciones
Comunicación serial Arduino	1	1	-
Habilitar / deshabilitar funcionamiento de los motores	0	4	Señal digital
Total	**5**	**17**	

Se puede concluir con la Tabla 6, que se requieren 20 salidas de las cuales 4 salidas deben ser para señal PWM, las señales de entrada son 5 y 4 de estas requieren que sean mediante interrupciones, este dato es muy importante para la selección del módulo Arduino, dado que no todos los Arduino poseen esta capacidad de entradas por interrupciones.

Controlador de potencia para motores DC

Para el control de los motores DC en la plataforma se necesita un controlador de potencia. El controlador para motores BTS7960 se aplica en motores de corriente continua, en la Tabla 7 se pueden observar las características de este dispositivo.

Tabla 7. Características del driver BTS7960 [13]

Símbolo	Parámetro	Valor	Unidad
V_s	Fuente de alimentación	6 - 27	V_{DC}
V_{ss}	Fuente de alimentación lógica	3,3 - 5	V_{DC}
D	Ciclo de trabajo	0 - 100	%
I_o	Corriente de salida máxima	43	A

Este componente permite el control de dos motores de corriente continua mediante señales TTL que se obtienen del controlador principal. Posee entradas de control las cuales reciben señales de 0V a 5V para otorgar la dirección del motor, en la Figura 25 se puede observar las conexiones del driver BTS7960.

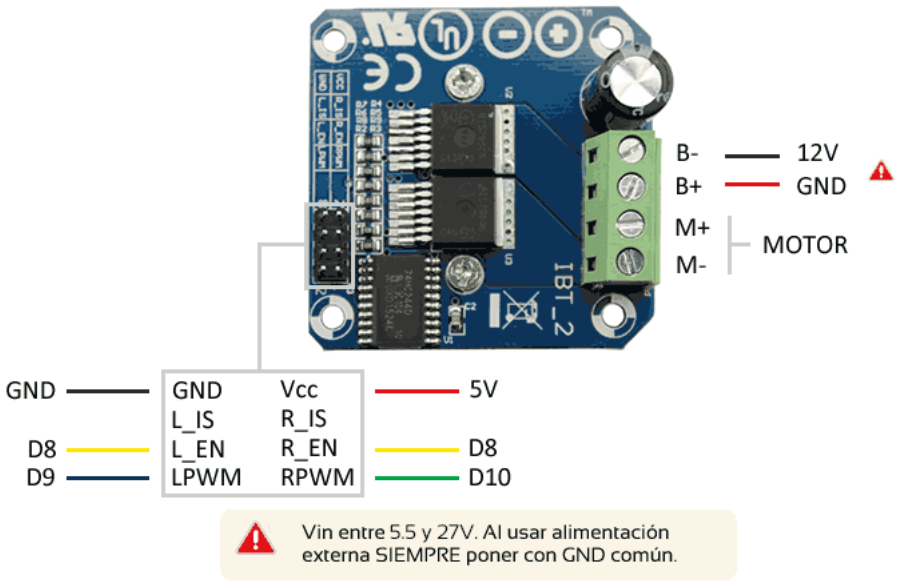

Figura 25. Driver BTS7960 para motores DC [8]

Debido que los motores implementados son de segunda mano, no cuentan con un datasheet que certifique las corrientes que los motores utilizan. Para verificar que corriente consume cada motor se realizaron pruebas experimentales donde se pudo concluir que los motores de 12 VDC consumen alrededor de 2,5 A y los motores de 24 VDC consumen alrededor de 1,5 A cuando se mueven la plataforma a su máxima velocidad. Por lo que este driver abastece la capacidad de corriente que los motores consumen.

Medición de velocidad en los motores DC

Se selecciona el encoder óptico FC-034 como se puede observar en la Figura 26, debido a que su frecuencia de medición se encuentra alrededor de 100 kHz y su alimentación es de 5 V, esto se debe a que los cuatro motores se encontrarán a una velocidad máxima de 40 RPM. Para la medición se adapta una rueda perforada con 12 agujeros; cada vez que se complete una revolución se detectan 12 pasos y ya que su velocidad máxima es de 40 RPM, se detectarían 480 pasos por minuto; es decir, una frecuencia máxima de 8 Hz. El principio de funcionamiento se basa en que posee dos leds infrarrojos, un emisor y un receptor colocados uno en frente del otro. Cada vez que se interrumpe la señal entre estos el encoder envía una señal lógica (1L), normalmente se diseña una rueda con orificios la misma que se acopla al eje del motor y al momento de girar se interrumpirá la señal de los leds infrarrojos para así capturar la señal.

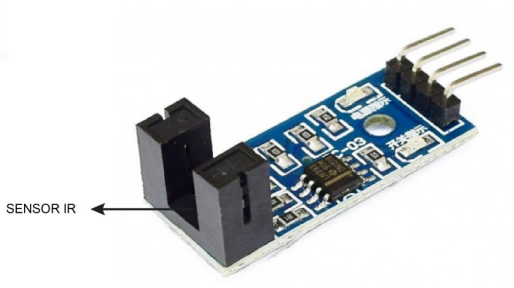

Figura 26. Encoder óptico FC-03 [9]

Fuente regulada de voltaje

Debido a que los motores y los drivers son elementos de potencia, se utiliza una fuente reguladora de voltaje para proteger y aislar la sección de control.

Este es el convertidor de voltaje DC – DC LM2596 capaz de entregar una salida constante inferior al voltaje de entrada soportando diferentes variaciones. Soporta corrientes de salida de hasta 3 A teniendo como voltaje de entrada entre 4,5 V a 40 V y voltaje de salida 1,23 V a 37 V. La selección del voltaje de salida se la efectúa mediante un potenciómetro. El módulo se puede apreciar en la Figura 27.

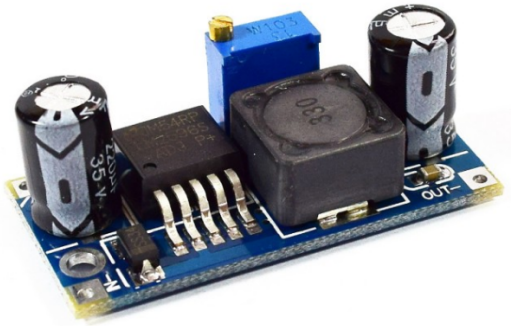

Figura 27. Convertidor de voltaje DC-DC LM2596 [10]

Con todos los elementos antes mencionados se elabora el circuito que se puede observar en el plano electrónico.

Módulo de comunicación inalámbrica

Para el envío y recepción de datos se necesita un módulo de comunicación inalámbrica por el cual se ha decidido utilizar un sistema de telemetría de radio 3dr que esta diseñada

como una fuente de radio de reemplazo de código abierto XBee. En las siguientes configuraciones: placa serie que se encuentra en la plataforma y el USB que se conecta en la estación con la cual el usuario controla la plataforma. El firmware Sik incluye un cargador de arranque que permite actualizaciones de firmware de radio sobre la interfaz serie, y firmware de radio con parámetros configurables [19]. Los módulos se pueden observar en la Figura 28.

Figura 28. Módulos de comunicación inalámbrica

Estos módulos cuentan con las siguientes características:

- Peso alrededor de 4 gramos por módulo sin antena.
- Rango de frecuencias 433 - 434,79 MHz.
- Diferentes configuraciones como ciclo de trabajo, velocidad de transmisión, recepción en su programa open source que se puede observar en el Anexo B.

Dispositivo de entrada

Debido que la plataforma es controlada a distancia, se utiliza un joystick para controlar el movimiento y desplazamiento de la plataforma, el joystick es el mando Logitch F310 como se puede observar en la Figura 29.

Figura 29. Mando Logitech F310 [11]

Baterías de alimentación

Para la selección de baterías, considerando la corriente que se necesita, se optó por una batería de motocicleta debido que estas son fáciles de encontrar en el mercado, se pueden recargar y su precio es accesible. Estas baterías entregan alrededor de 10 Ah y su voltaje nominal es de 12 V. Al poseer cuatro motores y elementos de control se suma la corriente que estos consumen y el resultado es una corriente total de 9 A aproximadamente. La batería seleccionada tiene como características 12 VDC y 12 Ah como se puede observar en la Figura 30.

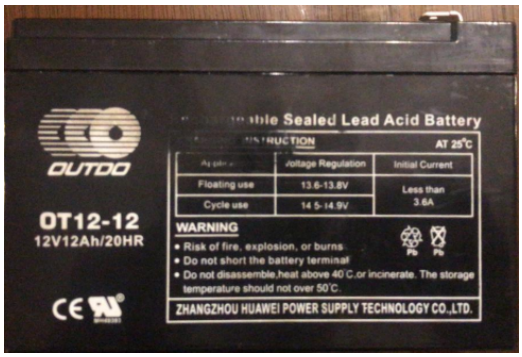

Figura 30. Batería de 12 VDC

6.3. Diseño del control de la plataforma

El control de la plataforma comprende la lógica de la programación, los programas y los algoritmos que se utilizan para el correcto funcionamiento de la plataforma móvil. El diagrama de flujo del control de velocidad se puede observar en la Figura 31.

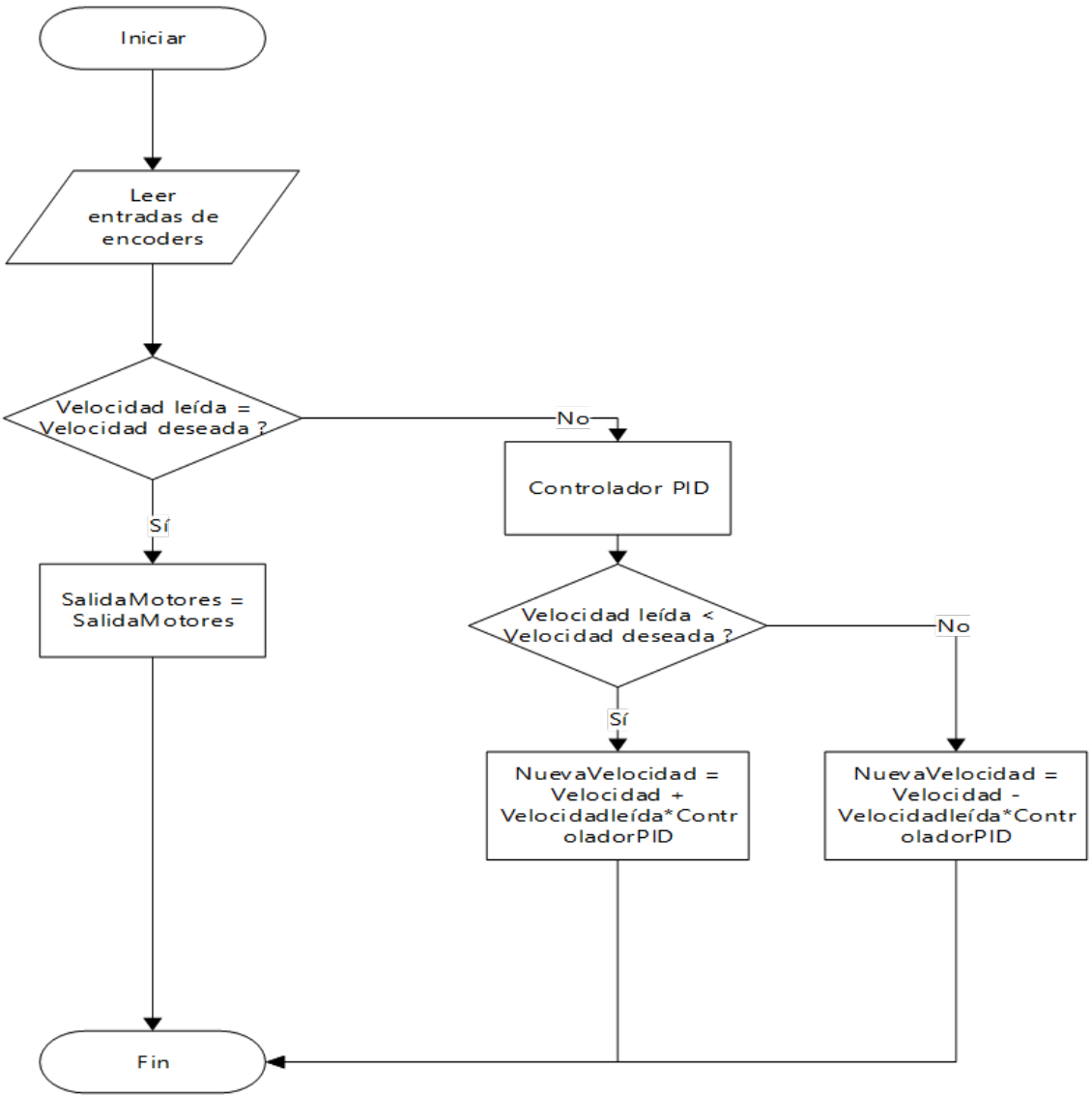

Figura 31. Diagrama de flujo del control de velocidad

Interfaz de control

La interfaz de control es capaz de crear el vínculo entre la plataforma y el usuario mediante una estación, en este caso la estación es una computadora personal que debe tener los siguientes requisitos mínimos:

- Dos puertos USB 2.0
- Procesador 1 GHz
- RAM 1.5 GB
- 10 GB de almacenamiento

- Ubuntu 16.04
- ROS Kinetic Kame

ROS Kinetic Kame

El proyecto utiliza la distribucion ROS Kinetic Kame que fue lanzada el 23 de mayo del 2016. Robotic Operating System (ROS) es un framework open source flexible que sirve para simplificar el comportamiento de diferentes plataformas robóticas. ROS tiene una colección de herramientas, bibliotecas y convenciones que ayudan simplificando la tarea de crear un comportamiento de robots [20].

Para entender un poco de cómo funciona la arquitectura de ROS y entender algunas herramientas de líneas de comando, se explican algunos términos como:

- **Nodos:** Un nodo en ROS es similar a un archivo ejecutable en Windows, este se utiliza para comunicarse con otros nodos.
- **Mensajes:** Tipo de dato de ROS.
- **Paquetes:** Los paquetes en ROS son el elemento principal ya que contienen procesos en tiempo de ejecución (nodos), conjuntos de datos y configuraciones.
- **Roscore:** Comando para iniciar ROS.
- **Topic:** Comando para que un nodo pueda publicar mensajes.

Desde la estación que contiene ROS se procede a enviar información de dirección y modo de ejecución a la plataforma móvil y recibir los datos de las velocidades de los motores mediante los paquetes JOY y ROSSERIAL.

Paquete JOY

El paquete de JOY es un driver para un joystick genérico en Linux. El paquete JOY contiene joy_node, un nodo que interconecta un joystick genérico de Linux con ROS. Este nodo publica un mensaje de "JOY", que contiene el estado actual de cada uno de los botones y ejes [21].

Para este caso en específico se utilizan tres comandos de este paquete. Para obtener los datos del joystick publicados sobre ROS, se necesita iniciar el nodo JOY. Primero, se establece en el nodo JOY qué dispositivo de joystick usar, esto depende del puerto de conexión de la estación (plataforma), en este caso es el $js2$.

```
rosparam set joy_node /dev"/dev/input/js2"
```

Luego se inicia el nodo JOY

```
rosrun joy joy_node
```

Por útimo, si se desean observar los datos que el joystick envía se ingresa el siguiente código en una nueva terminal y así cada acción que se ejecute en el joystick se podrá observar en la pantalla.

```
rostopic echo joy
```

Paquete ROSSERIAL

El paquete ROSSERIAL es un protocolo para envolver mensajes serializados ROS estándar y multiplexar múltiples temas y servicios a través de un dispositivo de caracteres, como un puerto serie o un zócalo de red [22]. Este paquete es compatible con los Arduinos Uno, Leonardo, MEGA, DUE.

Para este caso en específico se utiliza el comando `rosserial_python` que maneja automáticamente la configuración, publicación y suscripción de un dispositivo habilitado para ROSSERIAL conectado [23].

Con el siguiente comando se ejecuta el nodo con el puerto específico y una velocidad en baudios previamente seteada con 57600.

```
rosrun rosserial_python serial_node.py  /dev/ttyUSB0
```

Comunicación Arduino-ROS

Para la comunicación Arduino-ROS es necesario tener la librería `ros_lib`, la cual permite que Arduino interactúe con ROS. En el IDE de Arduino al momento de colocar las librerías se incluyen las siguientes:

```
#include <ros.h>
#include <std_msgs/Float32.h>
#include <sensor_msgs/Joy.h>
```

La primera línea de comando permite la comunicación con ROS, la segunda indica el tipo de dato que llega al Arduino y el tercero significa que llegan mensajes desde el paquete JOY.

De esta manera se enlaza la comunicación Arduino-ROS. Para continuar con el funcionamiento de la plataforma se pueden observar los diferentes modos de funcionamiento en el Anexo C. Los comandos que se reciben desde el joystick hacia el Arduino son:

```
joy.axes[7] == 1
joy.buttons[1] == 1
```

Tomando en cuenta que la primera línea de código es el movimiento de los potenciómetros analógicos en el joystick y el segundo comando el activar o pulsar algún botón.

Para el envío de datos desde Arduino se ingresan los siguientes comandos en el script de Arduino que se encuentra en su totalidad en el Anexo D.

```
ros::Publisher chatter("chatter", &str_msg);
nh.subscribe(sub1);
nh.advertise(chatter);
str_msg.data = hello;
chatter.publish( &str_msg );
```

Estos comandos publican mensajes en el entorno de ROS en forma de STRING. Para observar los datos que Arduino envía de forma serial se ingresa el siguiente código en una nueva terminal y asi cada que el Arduino envíe un código, este saldrá en pantalla.

```
rostopic echo chatter
```

Controlador PID

El objetivo de la plaforma móvil es controlar la velocidad mediante un controlador PID. Para ello se ejecutan las siguientes líneas de código en Arduino. La señal de entrada se calcula a través de los pulsos que entrega el encoder que serán contados periódicamente para

obtener la velocidad angular de los motores. Esto se realiza para cada encoder acoplado a cada motor y se puede observar de forma detallada a continuación.

```
double Compute(void)
{
  unsigned long Tactual = millis();
  unsigned long CambioTiempo = (Tactual - TiempoAnt);
  if ( CambioTiempo >= TMuestreo)
  {
    RPMMotor1 = (60 * 1000 / pulsosporvuelta) / (TMuestreo) * PulsosMotor1;
    /* Calcula las revoluciones por minuto*/
    PulsosMotor1 = 0;
    /* Inicializa los pulsos*/
    Entrada  = RPMMotor1;
    /*Calcula el error proporcional*/
    error  = (setpoint - Entrada)  * kp;
    /*Calcula el error derivativo*/
    eDer = (Entrada - EntradaAnt) * kd;
    /*El error integral se auto-ajusta a las circunstancias del motor*/
    if (eDer == 0.0)  eInt += (error * ki); else eInt -= (eDer * ki);
    /* Acota el error integral para eliminar el "efecto windup"*/
    if (eInt > outMax) eInt = outMax; else if (eInt < outMin) eInt = outMin;
    /* Suma todos los errores, es la salida del control PID*/
    double Output = error + eInt - eDer;
    if (Output > outMax) Output = outMax;
    /*Acota la salida para que el PWM pueda estar entre outMin y outMax*/
    else if (Output < outMin) Output = outMin;
    /*Se guarda la posición para convertirla en pasado*/
    EntradaAnt = Entrada;
    /* Se guarda el tiempo   para convertirlo en pasado*/
    TiempoAnt  = Tactual;
```

```
    /* Devuelve el valor de salida PID*/
    return Output;
  }
}
```

Se puede observar cómo se calculan los errores, tanto proporcional como derivativo, para después ser multiplicados por las constantes respectivas, en el caso del error integral se lo realiza de forma diferente por el efecto $windup$ ya que este es un efecto indeseable de la saturación en la actuación. Cualquier integrador de un controlador continuará integrando aún mientras la entrada se encuentra saturada; así, el estado del integrador en cuestión puede alcanzar valores excesivos, que deteriorarán la respuesta transitoria del sistema, generalmente produciendo grandes sobrevalores [7].

El comportamiento de los motores en los diferentes modos con diferentes setpoints se puede observar en las siguientes Figuras 32, 33, 34.

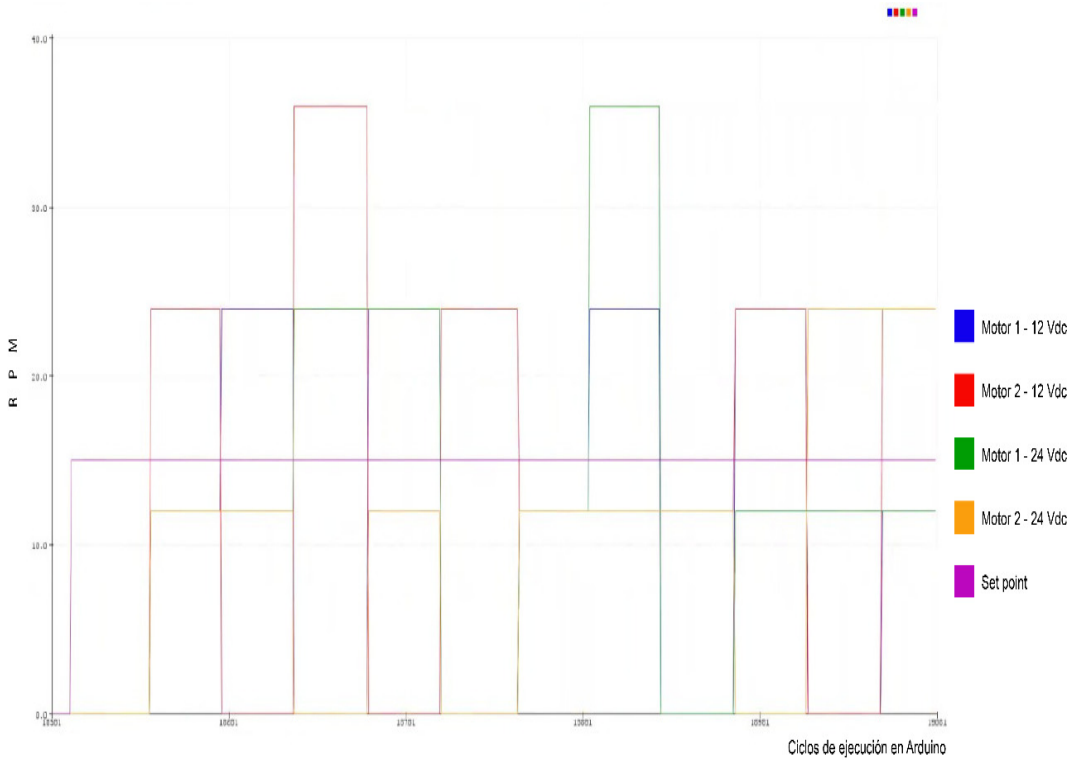

Figura 32. Modo 1, Setpoint = 15RPM

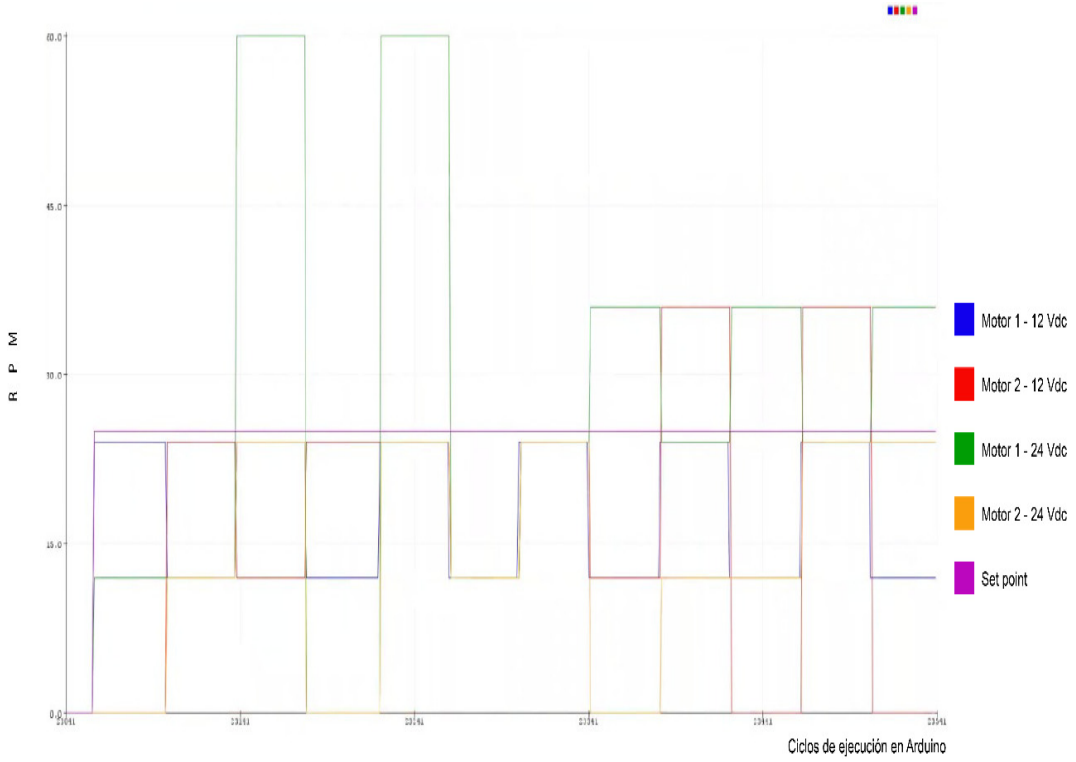

Figura 33. Modo 3, Setpoint = 25RPM

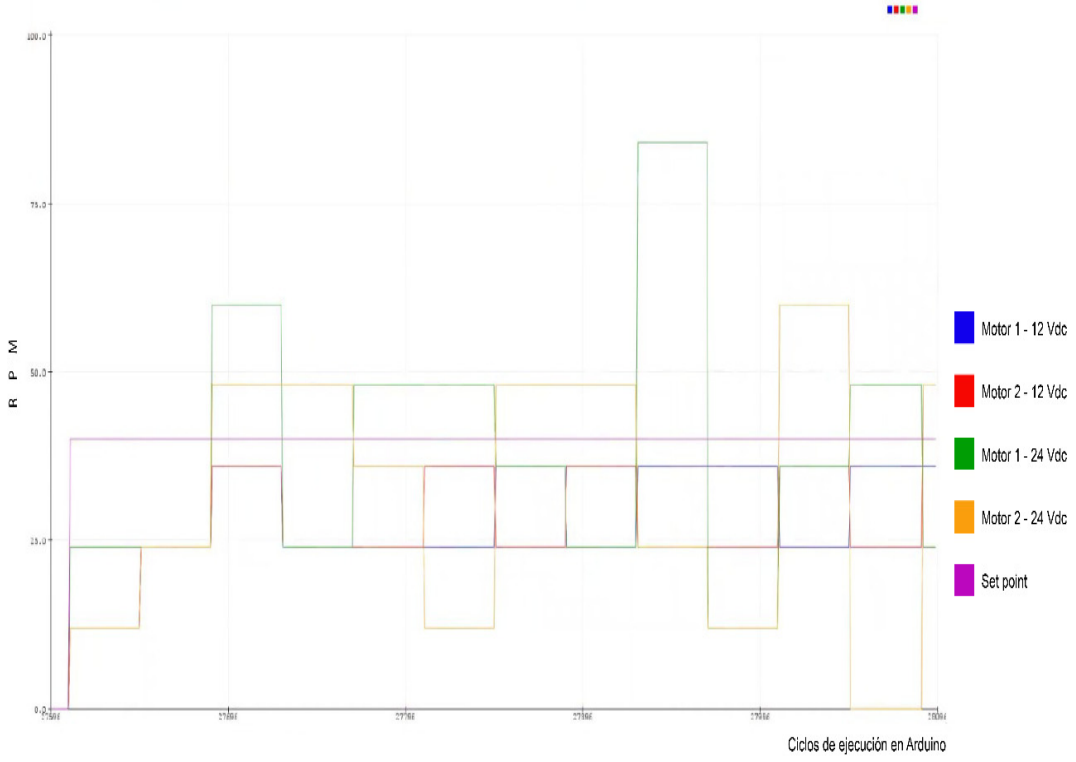

Figura 34. Modo 4, Setpoint = 40RPM

Determinación de las variables PID

Para realizar el control de la plataforma, se determinan los parámetros de la función de transferencia del motor utilizado. En el Anexo E se encuentra detallado el procedimiento para obtener los valores de forma teórica y así poder partir con una base y, según se requiera, modificarlos para obtener las constantes finales.

La obtención de las diferentes constantes kp, kd, ki se realizó experimentalmente. Se inicia con las constantes kp, ki y kd igual al valor de la Tabla 8, que se encontraron con el procedimiento detallado en el Anexo E.

Tabla 8. Parámetros PID iniciales para los motores DC

Motor	P	I	D
Motor 12 VDC (1)	12,72	0,02	0,05
Motor 12 VDC (2)	13,59	0,035	0,06
Motor 24 VDC (1)	11,24	0,059	0,02
Motor 24V DC (2)	10,79	0,043	0,019

Partiendo de estos valores se procede con un aumento de kp hasta encontrar una oscilación del sistema aceptable, esto significa que el motor no tenga cambios exagerados en su velocidad e intente oscilar lo menos posible. Luego se incrementa la constante kd hasta observar que las oscilaciones disminuyan en su mayoría y, por último, observando que los motores se encuentren estables con oscilaciones muy pequeñas se adecúa la constante ki, para que se estabilice lo antes posible al momento de encontrar perturbaciones.

Para las pruebas de funcionamiento se modificaron las ganancias del controlador PID a discreción basándose en el criterio de la Sintonía empírica basada en reglas [7], los pasos a seguir son los siguientes:

- Variar las constantes en el orden de kp, kd y ki.
- Un parámetro kp muy bajo hará que el motor no llegue al setpoint establecido. Por el contrario, un valor alto hará que el motor se encienda y se apague bruscamente. Una kp lo suficientemente buena hará que el motor avance con oscilaciones casi indetectables.
- Una vez que se establece kp, se ajusta kd. Un buen valor de kd disminuirá las oscilaciones hasta que el motor esté casi estable. Además, la cantidad correcta de kd

mantendrá el motor en una velocidad constante.

- Por último, se establece el parámetro ki. El motor llega a su setpoint en menor tiempo cuando este valor se encuentre correcto, también mejora el tiempo de llegar otra vez al setpoint si se ha desviado por alguna perturbación.

Después de realizar estas pruebas con diferentes parámetros las diferentes constantes concluyeron con los valores indicados en la Tabla 9.

Tabla 9. Parámetros PID finales para los motores DC

Motor	P	I	D
Motor 12 VDC (1)	11,27	0,075	0,0125
Motor 12 VDC (2)	14.53	0,075	0,016
Motor 24 VDC (1)	11,09	0,09	0,02
Motor 24 VDC (2)	11,36	0,09	0,025

7. Implementación

Para la implementación y puesta a punto se inició obteniendo los valores de las constantes para el control de velocidad por cada motor, luego se realizaron las mejoras en el aspecto mecánico como son los sistemas de suspensión y chasis, se llevaron a cabo pruebas de desplazamiento de la plataforma móvil en diferentes escenarios y se finalizó cambiando las constantes de manera experimental para un mejor funcionamiento.

7.1. Pruebas de funcionamiento

En las pruebas de funcionamiento se observa el comportamiento de la plataforma en distintos terrenos, pendientes y tiempos de pruebas.

Desplazamiento en pendientes

Para la realización de la prueba de desplazamiento en pendientes se debe tener diferentes inclinaciones. Se seleccionan tres diferentes pendientes y se mide su longitud, la prueba se la realiza midiendo el tiempo que la plataforma se demora en llegar al final de la pendiente en los diferentes modos (Ver Anexo C) y se calcula la velocidad. La velocidad que alcance la plataforma es directamente proporcional a la relación peso/potencia que se

puede observar en el Anexo F. Las pruebas realizadas se pueden observar en las Figuras 35, 36, 37.

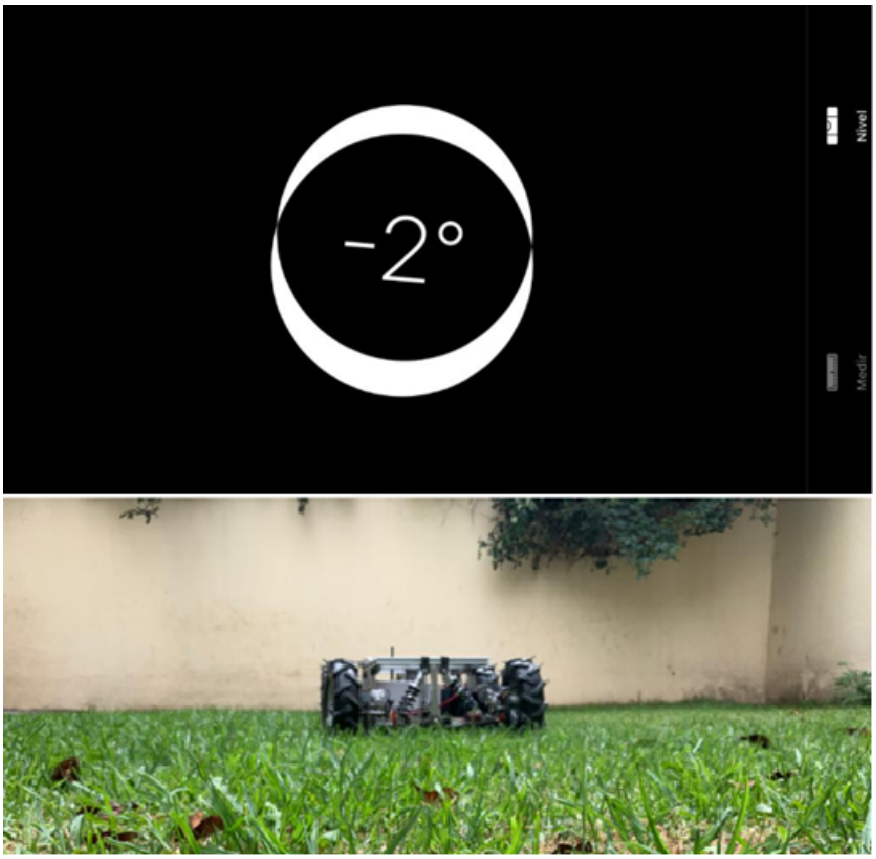

Figura 35. Primera Prueba

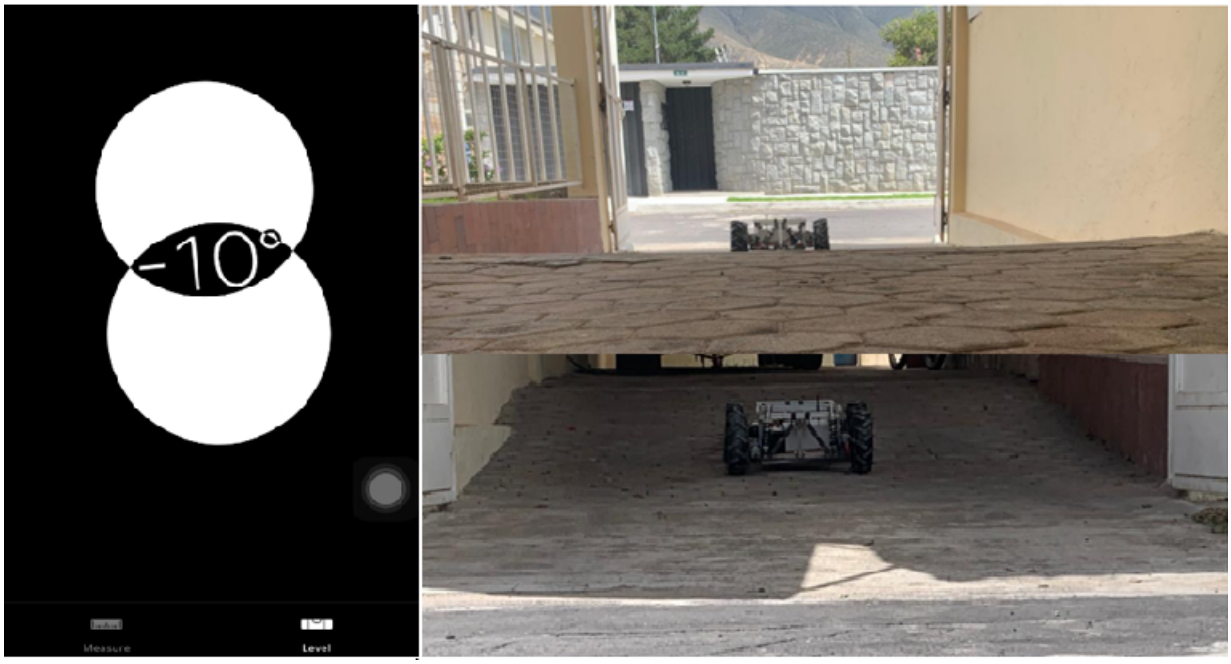

Figura 36. Segunda Prueba

Figura 37. Tercera Prueba

Se realizaron pruebas en los diferentes modos de desplazamiento de la plataforma móvil y los resultados se pueden observar en las Tablas 10, 11, 12.

Tabla 10. Prueba: Modo 1

Prueba	Inclinación [^{o}]	Longitud [m]	Tiempo [s]	Velocidad [km/h]
Primera	2	9	43	0,75
Segunda	10	6	62	0,34
Tercera	17	15	125	0,432

Tabla 11. Prueba: Modo 3

Prueba	Inclinación [^{o}]	Longitud [m]	Tiempo [s]	Velocidad [km/h]
Primera	2	9	29	1,11
Segunda	10	6	55	0,39
Tercera	17	15	89	0,60

Tabla 12. Prueba: Modo 2 y Modo 4

Prueba	Inclinación [^{o}]	Longitud [m]	Tiempo [s]	Velocidad [km/h]
Primera	2	9	20	1,62
Segunda	10	6	28	0,77
Tercera	17	15	65	0,83

Autonomía de Funcionamiento

En esta prueba lo que se busca es conocer la autonomía de la plataforma sin importar el terreno de operación.

Para esta prueba se cargó la batería al máximo, se encendió la plataforma, se tomó el tiempo y se operó la plataforma móvil hasta el momento donde la plataforma no pueda moverse. Se pueden observar los resultados en la Tabla 13.

Tabla 13. Resultados Pruebas de Autonomía

Prueba	Tiempo [min]
Primera	44
Segunda	38
Tercera	42
Promedio	**41,33**

Desplazamiento en diferentes terrenos

En esta prueba lo que se busca es conocer la velocidad de la plataforma móvil en tres diferentes terrenos, midiendo el tiempo y distancia para calcular la velocidad en los diferentes terrenos de operación que tengan un grado de inclinación similar.

Para esta prueba se cargó la batería al máximo, se encendió la plataforma, se tomó el tiempo, se estableció una distancia de 9 metros y se operó la plataforma móvil hasta la distancia previamente mencionada. Los resultados se pueden observar en las siguientes Tablas 14, 15, 16.

Tabla 14. Terreno: Pavimento

Prueba	Modo	Tiempo [s]	Velocidad [km/h]
Primera	1	42	0,77
Segunda	3	28	1,15
Tercera	2 y 4	19	1,7

Tabla 15. Terreno: Césped

Prueba	Modo	Tiempo [s]	Velocidad [km/h]
Primera	1	43	0,75
Segunda	3	29	1,11
Tercera	2 y 4	20	1,62

Tabla 16. Terreno: Tierra

Prueba	Modo	Tiempo [s]	Velocidad [km/h]
Primera	1	53	0,61
Segunda	3	31	1,04
Tercera	2 y 4	22	1,47

7.2. Resultados

Una vez concluidas las pruebas no se realiza ningún cambio en el sistema de control PID de la plataforma, se comparan los datos obtenidos con los datos de la Casa de la Calidad, esta comparación se puede observar en la Tabla 17.

Tabla 17. Resultados Obtenidos

	Casa de la calidad	Resultados
Peso máximo	25 [kg]	23,13 [kg]
Dimensiones máximas	700x600x300 [mm]	707x657x294 [mm]
Autonomía	40 [min]	41,33 [min]
Pendiente máxima	25°	17°
Estructura	Modular	Modular
Sistemas de suspension	Independientes	Independientes

Con los datos obtenidos en la anterior tabla se puede concluir que la plataforma se encuentra dentro de los parámetros requeridos dentro de la Casa de la Calidad dado que cumple con los requerimientos del usuario.

8. Conclusiones y Recomendaciones

8.1. Conclusiones

- Se construyó un prototipo de plataforma móvil con tracción diferencial y control de velocidad adaptado a las necesidades del proyecto, la plataforma cuenta con un control PID, sistemas de suspensión independientes, es teleoperada por radio frecuencia usando el sistema operativo ROS y cuenta con un motor DC por punto de apoyo.

- El control PID se lo realizó mediante el método de la curva de respuesta del motor con la tabla de Ziegler Nichols para obtener un punto de partida y continuar experimentalmente cambiando las diferentes constantes. Debido que las constantes obtenidas mediante este método fueron para cada motor en específico sin carga se tuvieron que

realizar ajustes una vez que toda la plataforma fue ensamblada, los primeros datos funcionaron como punto de partida para las constantes.

- En la plataforma móvil se mejoró el aspecto mecánico con la repotenciación de los sistemas de suspensión y el chasis a través de la inclusión de soldadura para que la estructura sea más rígida.

- El chasis de la plataforma al ser construida en perfiles de aluminio tipo V permite la incorporación de distintos elementos con mayor rapidez y que son accesibles en el mercado.

- El uso del sistema operativo ROS permite la integración de diferentes paquetes que pueden ser agregados o reemplazados según necesidades del proyecto debido a su modularidad, como por ejemplo tf2-ros, opencv3, robot-state-publisher, moveit-planners-chomp, entre otros.

- En cuanto a las pruebas obtenidas se concluye que en el peso máximo la plataforma cumple con los requisitos del usuario mientras que en las dimensiones tiene un error del 2,83 %

8.2. Recomendaciones

- Para mejorar en la autonomía de la batería se recomienda implementar un banco de baterías, se le puede agregar una batería extra para mejorar el tiempo o con baterías LIPO para reducir el peso, de esta manera mejora la relación peso/potencia.

- Se pueden implementar dos módulos como el Sabertooth 2 X 12 para optimizar espacio y no tener cuatro módulos para los motores. También se pueden utilizar módulos de comunicaciones con un mayor alcance.

- Para obtener mayor fuerza de arranque en subidas, se recomienda reemplazar los motores de 24V debido que estos no tienen caja reductora, por lo que son más rápidos que los motores de 12V, pero con menos torque, lo que hace que en subidas solo funcionen los motores de 12V.

BIBLIOGRAFÍA

[1] T. Braunl, *Embedded Robotics Mobile Robot Design and Applications with Embedded Systems*, segunda ed. Springer - Australia, 2006.

[2] Motor eléctrico. [En línea]. Disponible: https://www.areatecnologia.com/EL20MOTOR20ELECTRICO.htm

[3] M. Alvarez, “Modelo matemático de un motor de corriente continua separadamente excitado: Control de velocidad por corriente de armadura,” *Sin editorial*, Marzo 2012.

[4] "summit-xl | robotnik", robotnik. [En línea]. Disponible: https://www.robotnik.es/robots-moviles/summit-xl/.

[5] "dr robot inc.: Wifi 802.11 robot, network-based robot, robotic, robot kit, humanoid robot, oem solution", dr robot. [En línea]. Disponible: http://jaguar.drrobot.com/specification-lite.asp.

[6] Talon tracked military robot - army technology", army technology. [En línea]. Disponible: https://www.army-technology.com/projects/talon-tracked-military-robot/.

[7] A. Karl y H. Tore, *Control PID avanzado*, primera ed. PEARSON educacion - España, 2009.

[8] Driver bts7960. [En línea]. Disponible: https://www.luisllamas.es/controla-motores-de-gran-potencia-con-arduino-y-bts7960/

[9] Fc-03. [En línea]. Disponible: https://naylampmechatronics.com/robotica-movil/240-encoder-infrarrojo.html

[10] Convertidor de voltaje dc-dc lm2596. [En línea]. Disponible: https://naylampmechatronics.com/conversores-dc-dc/196-convertidor-voltaje-dc-dc-step-down-3a-lm2596.html

[11] Mando de control joystick. [En línea]. Disponible: https://www.logitechg.com/es-es/products/gamepads/f310-gamepad.html

[12] J. Wong, *Theory of Ground Vehicles*, 3rd ed. Otawa - Canada, 2001.

[13] Datasheet bts7960. [En línea]. Disponible: https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf

[14] Robot, "seekur jr mobile robot". [En línea]. Disponible: https://www.generationrobots.com/en/402399-robot-mobile-seekur-jr.html. [Fecha de consulta: Mes 2019]

[15] Robot, themachinelab.com. [En línea]. Disponible: http://www.themachinelab.com/Products/Basic/MMP/40/Specs.html.

[16] Treaded robots | sdr tactical, sdrtactical.com. [En línea]. Disponible: http://sdrtactical.com/lt2-bloodhound/.

[17] ïnspectorbot", generation robots. [En línea]. Disponible: https://www.generationrobots.com/en/402970-inspectorbots-super-mega-bot.html.

[18] "summit-xl steel | robotnik", robotnik. [En línea]. Disponible: https://www.robotnik.es/robots-moviles/summit-xl-steel/.

[19] Módulo telemetría 3dr. [En línea]. Disponible: https://www.infodomo.es/204/1120150089

[20] ros robotig operating system". [En línea]. Disponible: https://www.ros.org/about-ros/

[21] Joy package summary. [En línea]. Disponible: http://wiki.ros.org/joy

[22] Rosserial package summary. [En línea]. Disponible: http://wiki.ros.org/rosserial

[23] Rosserial_python package summary. [En línea]. Disponible: http://wiki.ros.org/rosserial_python

[24] "ziegler nichols". [En línea]. Disponible: https://www.academia.edu/7448768/ziegler-nichols

Anexo A: Implementación de Casa de Calidad

Mediante la Casa de la Calidad se definen especificaciones técnicas para el diseño y construcción de la plataforma móvil.

Debido a que algunos de los objetivos de la plataforma móvil son que sea tele operada y pueda desplazarse en terrenos exteriores se realiza un análisis para obtener las características del diseño y construcción.

Voz del Usuario

La voz del usuario describe los requerimientos de la persona que utilizará la plataforma móvil. Los requerimientos son:

- Ligera
- Que sea de fácil manejo
- Que sea fácil de desmontar
- Que pueda llevar carga
- Que tenga autonomía
- Que se pueda utilizar en diferentes terrenos
- Que se pueda acoplar sensores
- Que pueda subir pendientes

Voz del Ingeniero

El ingeniero define los siguientes parámetros para cumplir las demandas del usuario, y de esta manera lograr el diseño.

- Peso máximo 25kg
- Dimensiones máximas 700 mm x 600 mm x 300 mm
- Capacidad batería: 12 Vdc a 12 Ah

- Estructura modular
- Pendiente máxima: 25 grados
- Tiempo de funcionamiento 40 min
- Sistemas de suspensión independientes

Implementación de la Casa de la Calidad

Según la voz del usuario y la voz del ingeniero se desarrolla la Casa de la Calidad que se puede apreciar en la siguiente Figura.

Voz del ingeniero / Voz de usuario		Peso Máximo	Dimensiones	Capacidad de Batería	Tiempo de funcionamiento	Estructura	Resistencia al entorno		UIDE	Competencia 1	Competencia 2	Objetivos	Indice de mejora	Importancia	Ponderación	Ponderacion en %
Ligera		9	9	2	2	9	4		9	5	2	5	1	5	10	19%
Fácil manejo		6	6	0	0	2	2		6	9	8	3	1,1	3	4	8%
Fácil desmontar		0	1	0	0	9	1		9	8	5	4	1	5	8	15%
Autonomía		4	0	9	9	0	0		4	6	9	3	1,7	3	6	11%
Acoplar sensores		0	0	2	0	9	2		8	3	4	2	1	4	5	9%
Carga		0	0	0	0	9	4		6	3	3	3	1	3	5	9%
Diferentes terrenos		4	1	0	0	4	9		7	8	8	5	1	5	8	15%
Pendientes		6	0	9	0	4	3		5	8	9	4	1,7	3	7	13%
															53	100%
UIDE		9	9	4	4	9	8									
Competencia 1		5	8	6	6	9	9									
Competencia 2		2	8	9	9	9	9									
Incidencia		212	130	147	74	320	179	1062								
Incidencia en %		20%	12%	14%	7%	30%	17%	100%								
		25 kg	700x600x300	12VDC a 12Ah	45 min	Modular	Suspensión independiente									

Figura 1. Casa de la calidad

Anexo B: Programa 3DR Radio Config

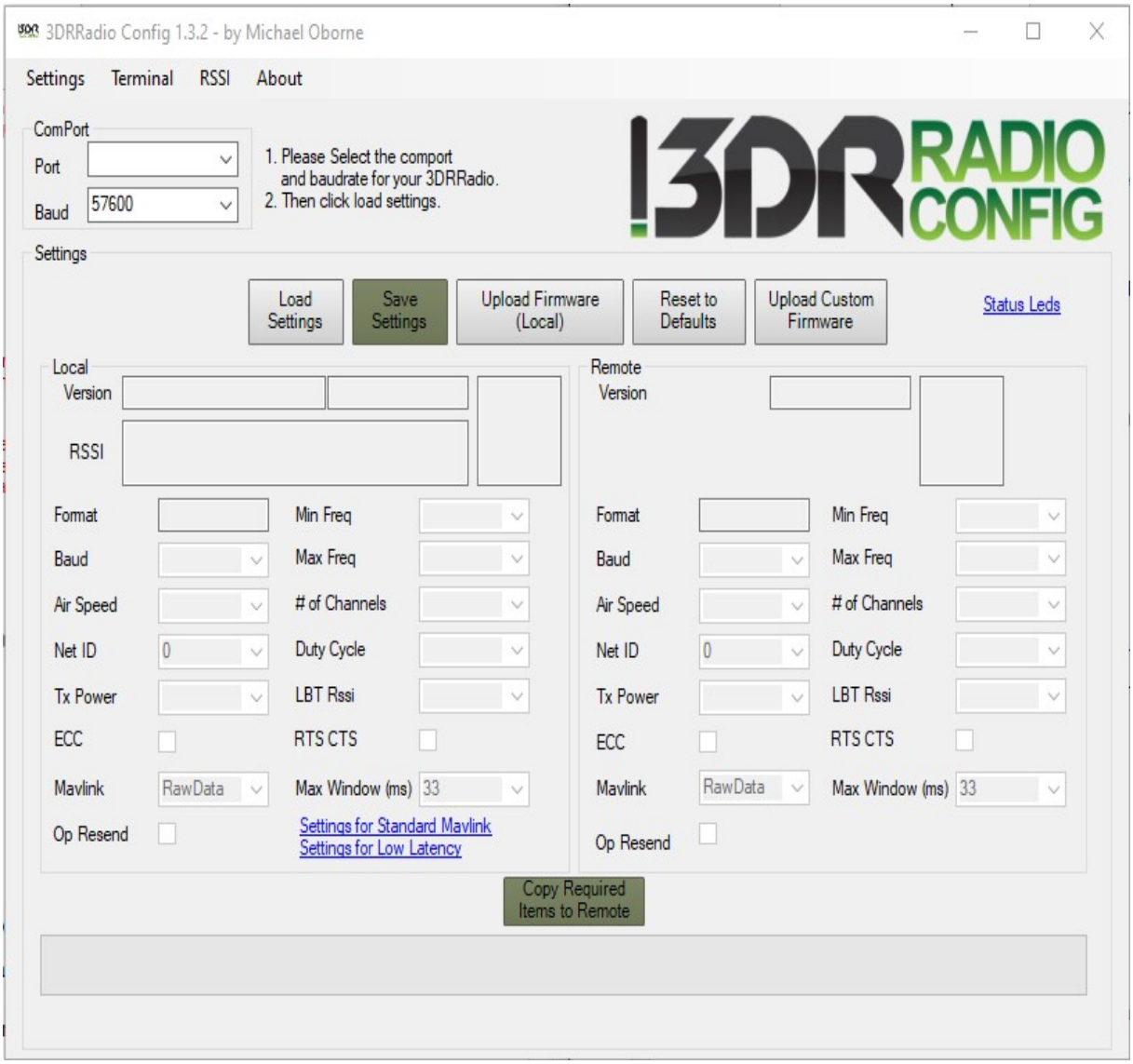

Figura 2. Programa 3DR Radio Config

Anexo C: Modos de funcionamiento

Para el desplazamiento de la plataforma móvil se tiene 4 modos de funcionamiento, los cuales se pueden ingresar pulsando los botones del control. En la siguiente Figura 3. se puede observar el botón con el que se acceden a los diferentes modos. De la figura anterior

Figura 3. Modos de funcionamiento

se puede observar que se cuenta con 4 modos de funcionamiento.

- **Modo 1.** Este modo cuenta con PID y su setpoint es el más bajo con un valor de 15 RPM. Para ejecutar el movimiento de los motores en este modo, a diferencia del Modo 2 se debe mover el análogo derecho al lado superior en su totalidad y el de la dirección según se desee.

- **Modo 2.** Este modo no cuenta con PID y la velocidad de los motores se encuentra establecida por el selector analógico derecho del control, esto significa que si se encuentra presionado al lado superior los motores tendrán su máxima velocidad; mientras que si no es presionado los motores no se moverán.

- **Modo 3.** Este modo cuenta con PID y su setpoint es de 25 RPM. Para ejecutar el movimiento de los motores en este modo, a diferencia del Modo 2 se debe mover el selector analógico derecho al lado superior en su totalidad y el de la dirección según se desee.

- **Modo 4.** Este modo cuenta con PID y su setpoint es el más alto con un valor de 25 RPM. Para ejecutar el movimiento de los motores en este modo, a diferencia del Modo

2 se debe mover el selector analógico derecho al lado superior en su totalidad y el de la dirección según se desee.

En las Figuras 4, 5, 6, 7, se puede observar los selectores analógicos y las direcciones que se deben realizar para ejecutar dichos movimientos en la plataforma móvil.

Figura 4. Dirección: Adelante

Figura 5. Dirección: Atrás

Figura 6. Dirección: Izquierda

Figura 7. Dirección: Derecha

Anexo D: Script Arduino

```
/// Librerias ///
#include "BTS7960.h"
#include <stdint.h>
#include <stdlib.h>
#include <ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Float32.h>
#include <sensor_msgs/Joy.h>


/// PINES ///

/*Motor 1 12VDC :*/
const uint8_t EN1 = 14;
const uint8_t L_PWM1 = 8;
const uint8_t R_PWM1 = 9;
#define Enc1M12 2
BTS7960 motorController1(EN1, L_PWM1, R_PWM1);
/*Motor 2 12VDC:*/
const uint8_t EN2 = 13;
const uint8_t L_PWM2 = 10;
const uint8_t R_PWM2 = 11;
#define Enc2M12 19
BTS7960 motorController2(EN2, L_PWM2, R_PWM2);

/*Motor 1 24VDC :*/
const uint8_t EN3 = 3;
const uint8_t L_PWM3 = 4;
```

```
const uint8_t R_PWM3 = 5;
#define Enc1M24 20
BTS7960 motorController3(EN3, L_PWM3, R_PWM3);
/*Motor 2 24VDC:*/
const uint8_t EN4 = 15;
const uint8_t L_PWM4 = 6;
const uint8_t R_PWM4 = 7;
#define Enc2M24 21
BTS7960 motorController4(EN4, L_PWM4, R_PWM4);

/// Variables ///
static volatile unsigned long Trebote = 0;
/*Tiempo del rebote.*/
unsigned int pulsosporvuelta = 20;
/* Número de muescas que tiene el disco del encoder.*/
const int DiametroRueda = 250;
/*Diámetro de la rueda pequeña[mm]*/
volatile byte PulsosMotor1 = 0;
/*Número de pulsos leidos por el Arduino en un segundo*/
volatile byte PulsosMotor2 = 0;
volatile byte PulsosMotor3 = 0;
volatile byte PulsosMotor4 = 0;
/*Medir Velocidad velocidad Motor 1 12VDC*/
unsigned int RPMMotor1 = 0;
/*Medir Velocidad velocidad Motor 2 12VDC*/
unsigned int RPMMotor2 = 0;
/*Medir Velocidad velocidad Motor 1 24VDC*/
unsigned int RPMMotor3 = 0;
/*Medir Velocidad velocidad Motor 2 24VDC*/
unsigned int RPMMotor4 = 0;
```

```
/*Radio Frecuencia*/
float dato1 = 0;
int inicio = 0;
int ContadorS = 0;
int Prueba = 0;


/// PID ///
double setpoint = 0;
/* setpoint en RPM 12-39*/
double outMin = 0.0, outMax = 255;
/* Límites para no sobrepasar la resolución del PWM.*/
/* Motor 1 12VDC */
double kp = 0.0, ki = 0.0, kd = 0;
/*Constantes: proprocional, integral y derivativa.*/
double eInt    = 0.0, eDer     = 0.0, EntradaAnt = 0.0;
/*Variables de error integral, error derivativo y posición anterior del motor*/
unsigned long TiempoAnt = 0, TiempoAct = 0, TMuestreo = 0;
/*Variables tiempo*/
double Entrada = 0;
/*Entrada de datos*/
double error = 0.0;
/*Desviación o error entre la posición real del motor y la posición designada.*/
byte pwm1 = 0;
/* Es el PWM, se transformará en voltaje real en las bobinas de los motores.*/
/* Motor 2 12VDC*/
double kp2 = 0.0, ki2 = 0.0, kd2 = 0;
double eInt2    = 0.0, eDer2     = 0.0, EntradaAnt2 = 0.0;
unsigned long TiempoAnt2 = 0, TiempoAct2 = 0, TMuestreo2 = 0;
```

```
double Entrada2 = 0;
double error2 = 0.0;
byte pwm2 = 0;
/*Motor 1 24VDC*/
double kp3 = 0.0, ki3 = 0.0, kd3 = 0;
double eInt3    = 0.0, eDer3     = 0.0, EntradaAnt3 = 0.0;
unsigned long TiempoAnt3 = 0, TiempoAct3 = 0, TMuestreo3 = 0;
double Entrada3 = 0;
double error3 = 0.0;
byte pwm3 = 0;
/*Motor 2 24VDC*/
double kp4 = 0.0, ki4 = 0.0, kd4 = 0;
double eInt4    = 0.0, eDer4     = 0.0, EntradaAnt4 = 0.0;
unsigned long TiempoAnt4 = 0, TiempoAct4 = 0, TMuestreo4 = 0;
double Entrada4 = 0;
double error4 = 0.0;
byte pwm4 = 0;


void joydata ( const sensor_msgs::Joy& joy) {

  if (joy.buttons[0] == 1) {
    if (inicio == 1) {
      inicio = 0;
    }
    else if (inicio == 0)  {
      inicio = 1;
    }
    Serial.println(inicio);
  }
```

```
else if (joy.buttons[1] == 1) {
  if (inicio == 2) {
    inicio = 0;
  }
  else if (inicio == 0) {
    inicio = 2;
  }
  Serial.println(inicio);
}
else if (joy.buttons[2] == 1) {
  if (inicio == 3) {
    inicio = 0;
  }
  else if (inicio == 0) {
    inicio = 3;
  }
  Serial.println(inicio);
}
else if (joy.buttons[3] == 1) {
  if (inicio == 4) {
    inicio = 0;
  }
  else if (inicio == 0) {
    inicio = 4;
  }
  Serial.println(inicio);
}

if (inicio == 1) {
```

```
  motorController1.Enable();
  motorController2.Enable();
  motorController3.Enable();
  motorController4.Enable();
  setpoint = 15;

    if (joy.axes[7] == 1 && joy.axes[4] == 1 ) {
      Prueba = 1;
      }
    else if (joy.axes[6] == 1 && joy.axes[4] == 1 ) {
      Prueba = 2;
      }
    else if (joy.axes[6] == -1 && joy.axes[4] == 1 ) {
      Prueba = 3;
      }
      else if (joy.axes[7] == -1 && joy.axes[4] == 1 ) {
      Prueba = 4;
    }
    else {
      Prueba = 0;
      setpoint = 0;
    }
}

else if (inicio == 2) {
  motorController1.Enable();
  motorController2.Enable();
  motorController3.Enable();
  motorController4.Enable();
  Prueba = 10000;
```

```
if (joy.axes[4] >= 0 && joy.axes[3] == 0) {
  if (joy.axes[7] == 1) {
    //prueba directa
    dato1 = float (joy.axes[4]) * 255;
    Serial.print("Adelante: "); Serial.println(dato1);

    motorController1.TurnLeft(dato1);

    motorController2.TurnLeft(dato1);

    motorController3.TurnLeft(dato1);

    motorController4.TurnLeft(dato1);

  }
  else if (joy.axes[6] == 1) {
    //prueba directa
    dato1 = float (joy.axes[4]) * 255;
    Serial.print("Izquierda: "); Serial.println(dato1);

    motorController1.TurnRight(dato1);

    motorController2.TurnRight(dato1);

    motorController3.TurnLeft(dato1);

    motorController4.TurnLeft(dato1);
  }
  else if (joy.axes[6] == -1) {
```

```
    //prueba directa
    dato1 = float (joy.axes[4]) * 255;
    Serial.print("Derecha: "); Serial.println(dato1);

    motorController1.TurnLeft(dato1);

    motorController2.TurnLeft(dato1);

    motorController3.TurnRight(dato1);

    motorController4.TurnRight(dato1);
  }
  else {
    setpoint = 0;
    motorController1.TurnRight(0);
    motorController2.TurnRight(0);
    motorController3.TurnRight(0);
    motorController4.TurnRight(0);
  }
}
if (joy.axes[4] <= 0 && joy.axes[3] == 0) {
  if (joy.axes[7] == -1) {
    //prueba directa
    dato1 = float (joy.axes[4]) * -255;
    Serial.print("Atras: "); Serial.println(dato1);

    motorController1.TurnRight(dato1);

    motorController2.TurnRight(dato1);
```

```
      motorController3.TurnRight(dato1);

      motorController4.TurnRight(dato1);

    }
    else {
      setpoint = 0;
      motorController1.TurnRight(0);
      motorController2.TurnRight(0);
      motorController3.TurnRight(0);
      motorController4.TurnRight(0);
    }
  }
}
 else if (inicio == 3) {

  motorController1.Enable();
  motorController2.Enable();
  motorController3.Enable();
  motorController4.Enable();
  setpoint = 25;

    if (joy.axes[7] == 1 && joy.axes[4] == 1 ) {
      Prueba = 1;
      }
    else if (joy.axes[6] == 1 && joy.axes[4] == 1 ) {
      Prueba = 2;
      }
    else if (joy.axes[6] == -1 && joy.axes[4] == 1 ) {
```

```
      Prueba = 3;
      }
      else if (joy.axes[7] == -1 && joy.axes[4] == 1 ) {
      Prueba = 4;
    }
    else {
      Prueba = 0;
      setpoint = 0;
    }
}
else if (inicio == 4) {

  motorController1.Enable();
  motorController2.Enable();
  motorController3.Enable();
  motorController4.Enable();
  setpoint = 40;

    if (joy.axes[7] == 1 && joy.axes[4] == 1 ) {
      Prueba = 1;
      }
    else if (joy.axes[6] == 1 && joy.axes[4] == 1 ) {
      Prueba = 2;
      }
    else if (joy.axes[6] == -1 && joy.axes[4] == 1 ) {
      Prueba = 3;
      }
      else if (joy.axes[7] == -1 && joy.axes[4] == 1 ) {
      Prueba = 4;
    }
```

```
      else {
        Prueba = 0;
        setpoint = 0;
      }
  }
}

ros::NodeHandle  nh;
ros::Subscriber<sensor_msgs::Joy> sub1("joy", joydata);
std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);

char hello[27] = " ";
String str,mov;

void setup() {
  /// PINES I/O ///
  pinMode(Enc1M12, INPUT);
  pinMode(Enc2M12, INPUT);
  pinMode(Enc1M24, INPUT);
  pinMode(Enc2M24, INPUT);

  /// Comunicacion Serial ///
  Serial.begin(57600);
  /*Configuración del puerto serie*/
  nh.initNode();
  nh.subscribe(sub1);
  nh.advertise(chatter);
  /// Interrupciones ///
  attachInterrupt(digitalPinToInterrupt(Enc1M12), contador, RISING);
```

```
/*Configuración de la interrupción, donde esta conectado en el flanco de subida.*/
attachInterrupt(digitalPinToInterrupt(Enc2M12), contador2, RISING);
attachInterrupt(digitalPinToInterrupt(Enc1M24), contador3, RISING);
attachInterrupt(digitalPinToInterrupt(Enc2M24), contador4, RISING);
/// Variables ///
PulsosMotor1 = 0;
RPMMotor1 = 0;
PulsosMotor2 = 0;
RPMMotor2 = 0;
PulsosMotor3 = 0;
RPMMotor3 = 0;
PulsosMotor4 = 0;
RPMMotor4 = 0;

//Constantes PID iniciales
setpoint = 0;
/*Asignacion de Setpoint en RPM*/
TMuestreo = 250;
/*Asignacion tiempo de muestreo en milis*/

/*constantes motor 1*/
kp = 1.1;
ki = 0.9;
kd = 1.25;
/*Constantes motor 2*/
kp2 = 1.45;
ki2 = 0.9;
kd2 = 1.5;
/*Constantes motor 3*/
kp3 = 1.1;
```

```
  ki3 = 0.75;
  kd3 = 0.2;
  /*Constantes motor 4*/
  kp4 = 1.1;
  ki4 = 0.75;
  kd4 = 0.22;

}

void loop() {
  nh.spinOnce();

 if (Prueba!=0){
  double Out = Compute();
  pwm1 = abs(Out);
  double Out2 = Compute2();
  pwm2 = abs(Out2);
  double Out3 = Compute3();
  pwm3 = abs(Out3);
  double Out4 = Compute4();
  pwm4 = abs(Out4);
  if (Prueba==1){
    mov = "Adelante";
    motorController1.TurnLeft(pwm1);
    motorController2.TurnLeft(pwm2);
    motorController3.TurnLeft(pwm3);
    motorController4.TurnLeft(pwm4);
    }
  else if (Prueba == 2 ) {
      mov = "Izquierda";
```

```
      motorController1.TurnLeft(pwm1);
      motorController2.TurnRight(pwm2);
      motorController3.TurnRight(pwm3);
      motorController4.TurnLeft(pwm4);
      }
  else if (Prueba == 3 ) {
        mov = "Derecha";
        motorController1.TurnRight(pwm1);
        motorController2.TurnLeft(pwm2);
        motorController3.TurnLeft(pwm3);
        motorController4.TurnRight(pwm4);
        }
  else if (Prueba == 4 ) {
        mov = "Atras";
        motorController1.TurnRight(pwm1);
        motorController2.TurnRight(pwm2);
        motorController3.TurnRight(pwm3);
        motorController4.TurnRight(pwm4);
        }
//        str = mov + " " + String(pwm1)+" "+ String(pwm2)+" "+
String(pwm3)+" "+ String(pwm4)+" "+ String(setpoint);
        str = mov + " " + String(RPMMotor1)+" "+ String(RPMMotor2)+" "+
        String(RPMMotor3)+" "+ String(RPMMotor4)+" "+ String(setpoint);
        str.toCharArray(hello,27);
        //Serial.println(hello);
  }
 else {
  mov = "Nada";
  pwm1 = 0;
  pwm2 = 0;
```

```
  pwm3 = 0;
  pwm4 = 0;
  motorController1.TurnLeft(0);
  motorController2.TurnRight(0);
  motorController3.TurnRight(0);
  motorController4.TurnLeft(0);
  str = mov + " " + String(pwm1)+" "+ String(pwm2)+" "+ String(pwm3)+"
  "+ String(pwm4)+" "+ String(setpoint);
//  str = mov + " " + String(RPMMotor1)+" "+ String(RPMMotor2)+" "+
String(RPMMotor3)+" "+ String(RPMMotor4)+" "+ String(setpoint);
  str.toCharArray(hello,27);
  //Serial.println(hello);
 }
  str_msg.data = hello;
  chatter.publish( &str_msg );
}

void contador()
{
  if (  digitalRead (Enc1M12) && (micros() - Trebote > 500) &&
  digitalRead (Enc1M12) ) {
/*Vuelve a comprobar que el encoder envia una señal buPWM1 y luego comprueba*/
/*que el tiempo es superior a 1000 microsegundos y vuelve a comprobar que la */
/*señal es correcta*/
    Trebote = micros();
    /*AlmacPWM1 el tiempo para comprobar que no contamos el rebote que hay en*/
    /*la señal*/
    PulsosMotor1++;
     /*Suma el pulso bueno que entra*/
  }
```

```
  else ;
}
void contador2()
{
  if (digitalRead(Enc2M12) && (micros() - Trebote > 500) && digitalRead(Enc2M12))
  {
    Trebote = micros();
    PulsosMotor2++;
  }
  else ;
}
void contador3()
{
  if (digitalRead(Enc1M24) && (micros() - Trebote > 500) && digitalRead(Enc1M24))
  {
    Trebote = micros();
    PulsosMotor3++;
  }
  else ;
}
void contador4()
{
  if (digitalRead(Enc2M24) && (micros() - Trebote > 500) &&
  digitalRead(Enc2M24))
  {
    Trebote = micros();
    PulsosMotor4++;
  }
  else ;
}
```

```
double Compute(void)
{
  unsigned long Tactual = millis();
  /* Toma el número total de milisegundos que hay en ese instante.*/
  unsigned long CambioTiempo = (Tactual - TiempoAnt);
  /* Resta el tiempo actual con el último tiempo que se guardó*/
  if ( CambioTiempo >= TMuestreo)
  /*Si se cumple el tiempo de muestreo entonces calcula la salida.*/
  {
    RPMMotor1 = (60 * 1000 / pulsosporvuelta ) / (TMuestreo) *
    PulsosMotor1;
    /*Calculamos las revoluciones por minuto*/
    PulsosMotor1 = 0;
    /* Inicializamos los pulsos.*/
    Entrada  = RPMMotor1;
    error  = (setpoint - Entrada)  * kp;
    /*Calcula el error proporcional.*/
    eDer = (Entrada - EntradaAnt) * kd;
    /*Calcula el error derivativo*/
    /*Esta línea permite dos cosas: 1) Suaviza la llegada a la meta.*/
    /*2) El error integral se auto-ajusta a las circunstancias del motor.*/
    if (eDer == 0.0)  eInt += (error * ki); else eInt -= (eDer * ki);
    /*Acota el error integral para eliminar el "efecto windup".*/
    if (eInt > outMax) eInt = outMax;
    else if (eInt < outMin) eInt = outMin;
    /*Suma todos los errores, es la salida del control PID.*/
    double Output = error + eInt - eDer;
    if (Output > outMax) Output = outMax;
    else if (Output < outMin) Output = outMin;
```

```
    /*Acota la salida para que el PWM pueda estar entre outMin y outMax.*/
    /* Se guarda la posición para convertirla en pasado.*/
    EntradaAnt = Entrada;
    /*Se guarda el tiempo   para convertirlo en pasado.*/
    TiempoAnt  = Tactual;
    return Output;
    /*Devuelve el valor de salida PID.*/
  }
}
double Compute2(void)
{
  unsigned long Tactual2 = millis();
  unsigned long CambioTiempo2 = (Tactual2 - TiempoAnt2);
  if ( CambioTiempo2 >= TMuestreo)
  {
    RPMMotor2 = (60 * 1000 / pulsosporvuelta ) / (TMuestreo) * PulsosMotor2;
    PulsosMotor2 = 0;
    Entrada2  = RPMMotor2;
    error2  = (setpoint - Entrada2)  * kp2;
    eDer2 = (Entrada2 - EntradaAnt2) * kd2;
    if (eDer2 == 0.0)  eInt2 += (error2 * ki2);
    else eInt2 -= (eDer2 * ki2);
    if (eInt2 > outMax) eInt2 = outMax;
    else if (eInt2 < outMin) eInt2 = outMin;
    double Output2 = error2 + eInt2 - eDer2;
    if (Output2 > outMax) Output2 = outMax;
    else if (Output2 < outMin) Output2 = outMin;
    EntradaAnt2 = Entrada2;
    TiempoAnt2  = Tactual2;
    return Output2;
```

```
  }
}
double Compute3(void)
{
  unsigned long Tactual3 = millis();
  unsigned long CambioTiempo3 = (Tactual3 - TiempoAnt3);
  if ( CambioTiempo3 >= TMuestreo)
  {
    RPMMotor3 = (60 * 1000 / pulsosporvuelta ) / (TMuestreo) * PulsosMotor3;
    PulsosMotor3 = 0;
    Entrada3  = RPMMotor3;
    error3  = (setpoint - Entrada3)  * kp3;
    eDer3 = (Entrada3 - EntradaAnt3) * kd3;
    if (eDer3 == 0.0)  eInt3 += (error3 * ki3);
    else eInt3 -= (eDer3 * ki3);
    if (eInt3 > outMax) eInt3 = outMax;
    else if (eInt3 < outMin) eInt3 = outMin;
    double Output3 = error3 + eInt3 - eDer3;
    if (Output3 > outMax) Output3 = outMax;
    else if (Output3 < outMin) Output3 = outMin;
    EntradaAnt3 = Entrada3;
    TiempoAnt3  = Tactual3;
    return Output3;
  }
}
double Compute4(void)
{
  unsigned long Tactual4 = millis();
  unsigned long CambioTiempo4 = (Tactual4 - TiempoAnt4);
  if ( CambioTiempo4 >= TMuestreo)
```

```
  {
    RPMMotor4 = (60 * 1000 / pulsosporvuelta ) / (TMuestreo) * PulsosMotor4;
    PulsosMotor4 = 0;
    Entrada4  = RPMMotor4;
    error4  = (setpoint - Entrada4)  * kp4;
    eDer4 = (Entrada4 - EntradaAnt4) * kd4;
    if (eDer4 == 0.0)  eInt4 += (error4 * ki4);
    else eInt4 -= (eDer4 * ki4);
    if (eInt4 > outMax) eInt4 = outMax;
    else if (eInt4 < outMin) eInt4 = outMin;
    double Output4 = error4 + eInt4 - eDer4;
    if (Output4 > outMax) Output4 = outMax;
    else if (Output4 < outMin) Output4 = outMin;
    EntradaAnt4 = Entrada4;
    TiempoAnt4  = Tactual4;
    return Output4;
  }
}
```

Anexo E: Determinación de las variables teóricas del PID

En primera instancia se realiza un acople para el eje del motor con 1, 8, 12, agujeros por vuelta, no hacen contacto, interactúan pero de forma óptica con el sensor infrarrojo que actúa como encoder. El número de eslabones viene relacionado directamente con el error, dado que el motor que se utiliza es de bajas revoluciones y alto torque, se incrementó el número de orificios hasta tener un error aceptable.

Se realiza la prueba con un Arduino, sensor infrarrojo (encoder) y Simulink (Matlab). Se programa el Arduino para enviar señales mediante comunicación serial la que será captada por medio de Simulink en el PC. Para la comunicación Arduino-Simulink se implementa el siguiente código en Arduino para el envío y recepción de datos.

```
#define Stby 0
/*On-Off motores*/
#define DirA1 4
/*Direccion motor 1*/
#define DirA2 5
/*Direccion motor 1*/
//#define DEBUG(a) Serial.println(a)
#define PWM1 9
/*Pines PWM */
byte pwm = 0;
/*Es el PWM, se transformará en voltaje real en las bobinas de los motores.*/
int data = 0;
volatile float contador=0,RPM=0;

void setup() {
  Serial.begin(9600);

  attachInterrupt(0,interrupcion0,RISING);
  /*Interrupcion 0 (pin2) */
```

```
  pinMode(PWM1, OUTPUT);
  /*Definir pines*/
  pinMode(DirA1, OUTPUT);
  /*Definir pines*/
  pinMode(DirA2, OUTPUT);
  /*Definir pines*/

  analogWrite(PWM1,150);
  digitalWrite(Stby,HIGH);
  /*encender motores*/

  digitalWrite(DirA1, HIGH);
  /* Setear Pines de direccion*/
  digitalWrite(DirA2, LOW);
  /*Setear Pines de direccion*/

}

void loop() {
  if ((millis()\%250)==0){

    if (contador!=0){
      RPM=contador;
      /* Como son dos interrupciones por vuelta (contador * (60/2))*/
      Serial.println(contador);
    }
    contador = 0;

      data = Serial.parseInt();
```

```
      if (data!=0){
      pwm = (data);
      /*Transfiere a la variable pwm el valor absoluto de Out.*/
      analogWrite(PWM1,pwm);
      }
  }
  //Serial.println(RPM);
}

void interrupcion0()
/*Funcion que se ejecuta durante cada interrupion*/
{
  contador++;
  /*Se incrementa en uno el contador*/
}
```

Este código en primera instancia setea los diferentes pines que van a ser utilizados como entrada y salidas del motor, como dirección, PWM, y el envío de datos según la cantidad de agujeros que tiene la rueda encoder. Toda esta información se envía al Simulink para ser observada de manera gráfica.

1.1. Diagrama de bloques Simulink

En la Figura 8 se observa como se utilizan los diferentes bloques en Simulink para observar las RPM del motor.

En el bloque de Query Instrument se configura de acuerdo al puerto que se encuentre conectado el Arduino, también se debe tomar en cuenta la velocidad de comunicación, para este caso se utiliza 9600 baudios.

Se realiza una variación de voltaje con el propósito de poder observar las diferentes revoluciones, por lo que se obtiene la Tabla 1 en donde se representan los valores máximos, medios y mínimos en RPM según el voltaje aplicado.

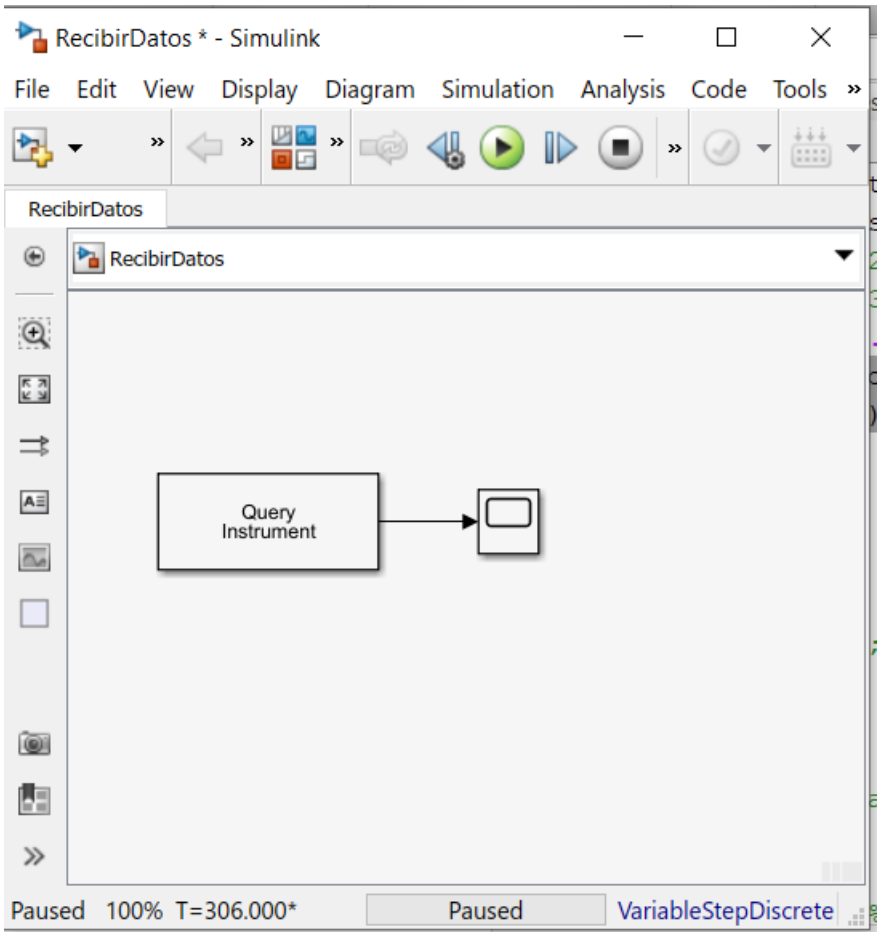

Figura 8. Bloques en Simulink

Tabla 1. RPM según su Voltaje

Voltaje	5 Vdc	7 Vdc	9 Vdc	11 Vdc
Valor Máximo	200	220	240	260
Valor Medio	150	170	200	230
Valor Mínimo	110	130	180	200

Después de obtener los datos del motor, se grafican con el fin de comprobar si esta planta posee un comportamiento lineal en determinados rangos para la implementación de un controlador que permita regular su velocidad. Como se observa en la Figura 9, el motor tiene un comportamiento lineal, por lo que se realiza la variación en tres casos:

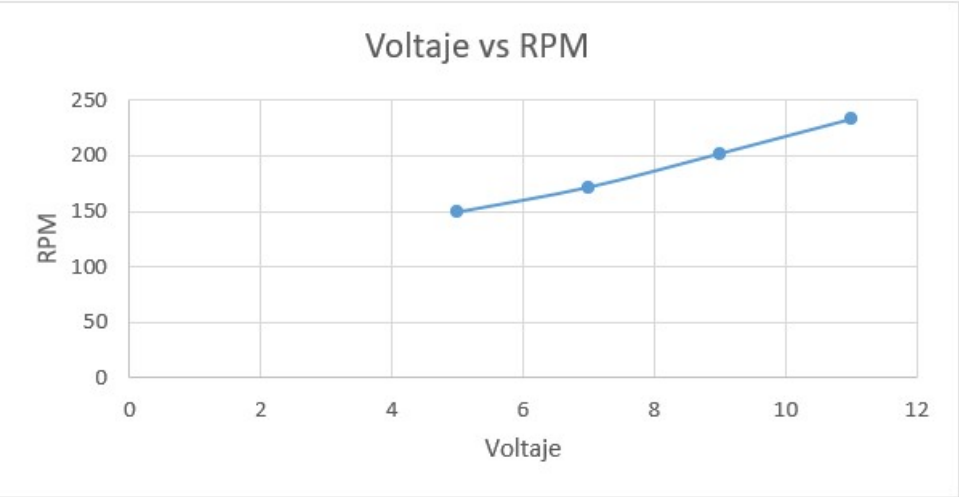

Figura 9. Gráfico Voltaje vs RPM

Variación de 5 a 7 V:

Primero se calcula la variación de velocidad que tiene el rango de 5 a 7 V.

$$\Delta\omega = \omega_2 - \omega_1$$

$$\Delta\omega = 170 - 150 = 20 \ [\mathrm{RPM}]$$

Después de ello se obtiene el 63 % de la variación de la velocidad angular.

$$\Delta\omega \cdot 63\,\% = 20 \ [\mathrm{RPM}] \cdot 63\,\%$$

$$\Delta\omega \cdot 63\,\% = 13 \ [\mathrm{RPM}]$$

Experimentalmente el valor de τ=0,1 debido a que el cambio de velocidad con respecto al cambio de voltaje es tan rápido, que asume ese valor y un valor del valor de τ; además ya se obtiene el valor de k de la siguiente función de transferencia de orden 1, a la cual también se puede simular a un motor DC, debido que la función de transferencia de un motor es de segundo orden.

$$G_{(s)} = \frac{k}{\tau \cdot s + 1}$$

$$k = \frac{\Delta\omega}{\Delta V_a}$$

$$k = 10{,}5$$

Variación de 7 a 9 V

Cumpliendo con el mismo procedimiento de la variación de 5 a 7 V se calcula la variación de la velocidad.

$$\Delta\omega = \omega_2 - \omega_1$$

$$\Delta\omega = 200 - 170 = 30 \ [\mathrm{RPM}]$$

Luego se obtiene el 63 % de la variación de la velocidad angular.

$$\Delta\omega \cdot 63\,\% = 30 \ [\mathrm{RPM}] \cdot 63\,\%$$

$$\Delta\omega \cdot 63\,\% = 19 \ [\mathrm{RPM}]$$

Por último, se otorga un valor a τ de 0.1 y se obtiene el valor de k de la siguiente función de transferencia de orden 1. Se puede simular un motor DC con una función de transferencia de orden 1 a pesar de que la función de un motor es de orden 2, debido que esta es linear.

$$G_{(s)} = \frac{k}{\tau \cdot s + 1}$$

$$k = \frac{\Delta\omega}{\Delta V_a}$$

$$k = 15$$

Variación de 9 a 11 V

Cumpliendo con el mismo procedimiento anterior se calcula la variación de la velocidad.

$$\Delta\omega = \omega_2 - \omega_1$$

$$\Delta\omega = 230 - 200 = 30 \ [\mathrm{RPM}]$$

Luego se obtiene el 63 % de la variación de la velocidad angular.

$$\Delta\omega \cdot 63\,\% = 30\ [\mathrm{RPM}] \cdot 63\,\%$$

$$\Delta\omega \cdot 63\,\% = 19\ [\mathrm{RPM}]$$

Por último, se otorga un valor a τ de 0.1 y se obtiene el valor de k de la siguiente función de transferencia de orden 1 a la cual también se puede simular a un motor DC.

$$G_{(s)} = \frac{k}{\tau \cdot s + 1}$$

$$k = \frac{\Delta\omega}{\Delta V_a}$$

$$k = 15$$

Para finalizar se calcula el promedio de k y τ para formar una sola función de transferencia para el motor DC donde:

$$k_{PROMEDIO} = 13{,}5$$

$$\tau_{PROMEDIO} = 0{,}1$$

Al final se obtienen 3 funciones de transferencia con los resultados anteriores

$$G_{1(s)} = \frac{10}{0{,}1 \cdot s + 1}$$

$$G_{2(s)} = G_{3(s)} = \frac{15}{0{,}1 \cdot s + 1}$$

De estas tres funciones de transferencia se pueden obtener los parámetros PID con la tabla de Ziegler Nichols como se muestra en la Tabla 2.

Tabla 2. Parámetros PID [24]

Controlador	k_p	T_i	T_d
P	T/L		0
PI	0,9 T/L	T/0,3	0
PID	1,2T/L	2L	0,5L

De esta tabla se conoce que L es el tiempo de retardo, T es la constante de tiempo y todo se ve reflejada en una recta tangente en el punto de inflexión de la curva con forma de

S, como se puede observar en la Figura 10.

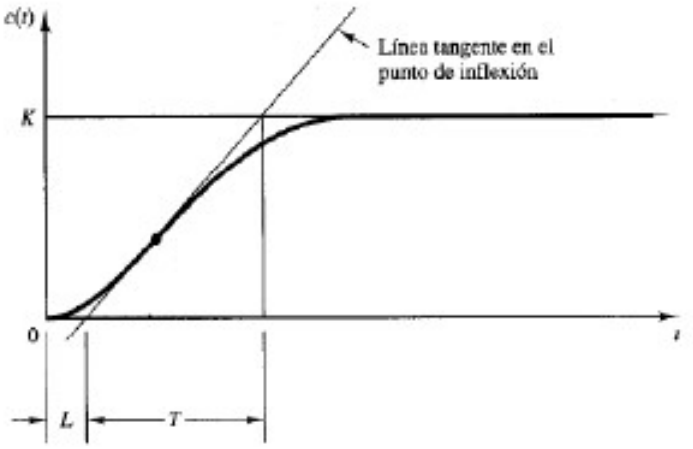

Figura 10. Curva de respuesta [24]

Para esto se realizó un script en Matlab que se puede observar a continuación con la función de transferencia que se obtuvo previamente, para poder realizar la recta tangente y obtener los valores anteriormente mencionados.

/*Encontrar L y T para Zieger Nichols */

```
H = tf ([16.667],[0.1 1]);
step(H);
hold on
dt = 0.01;
/* Vector de tiempo*/
t = 0:dt:8;
y=step(H,t)';
dy=diff(y)/dt;
/*m en el maximo y p es donde se encuentra */
[m,p] = max(dy);
y1=y(p);
t1=t(p);
plot (t1,y1,'*g')
hold on
t2=0:1:10;
y2=m*(t2-t1)+y1;
```

```
hold on
plot (t2,y2,'r')
plot (y2,t1,'or')
```

Dando como resultado la siguiente Figura 11.

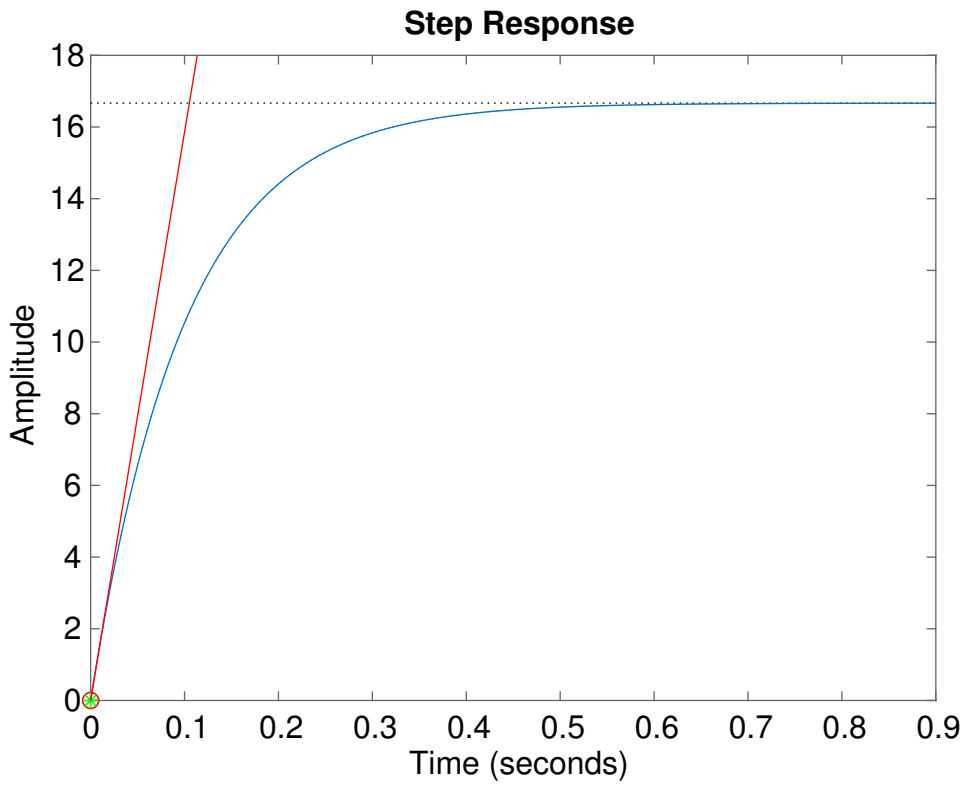

Figura 11. Curva del motor

De la anterior figura y con el script anterior, se obtuvo los valores de $L = 0{,}01$ y $T = 0{,}116$ para los que se pudieron sacar las constantes iniciales del controlador PID para un Motor de 12 V; en la siguiente Tabla 3 se puede observar las constantes.

Tabla 3. Constantes PID

	k_p	T_i	T_d
PID	12,72	0,02	0,05

Anexo F: Relación Peso/Potencia

La relación peso/potencia es un simple ratio que mide la potencia disponible para mover cada kilo de peso de la plataforma en este caso.

Como ejemplo se pueden observar dos coches con una relación peso-potencia de 1 kg/CV. Uno de ellos pesa 600 kg y tiene 600 CV de potencia. El otro, pesa 3.000 kg y tiene 3.000 CV de potencia.

Ambos tienen una relación potencia-peso de 1kg/CV, ¿pero cuál sería el coche más rápido y efectivo? A nivel de velocidad punta, el coche de 3.000 CV sería mucho más rápido, el peso no es tan relevante a la hora de alcanzar una buena punta. Sin embargo, en un circuito revirado, las enormes inercias del coche de 3.000 kilos lastrarán enormemente su paso por curva, necesitará enormes frenos para detenerse con su inmensa masa y no acelerará tan rápido en las cortas rectas del trazado. Es por ello que se tiene en cuenta el peso de un coche con respecto a su relación peso/potencia.

Por lo que esta relación representa un índice de la capacidad de aceleración, para las velocidades bajas y medias, así como de su marcha por una cuesta.