



# **Control Predictivo para el Seguimiento de Trayectorias en Vehículos Autónomos Terrestres**

**Juan Carlos Tique Rangel**

**Facultad de Ingeniería  
Programa de Ingeniería Electrónica  
Ibagué, 2020**



# **Control Predictivo para el Seguimiento de Trayectorias en Vehículos Autónomos Terrestres**

**Juan Carlos Tique Rangel**

Trabajo de grado presentado como requisito parcial para optar al título de:  
**Ingeniero Electrónico**

Director:  
PhD Ing. Oscar Barrero Mendoza  
Profesor Universidad de Ibagué

Facultad de Ingeniería  
Programa de Ingeniería Electrónica  
Ibagué, 2020



# Dedicatoria

A mis padres  
Con amor, respeto y admiración



# **Agradecimientos**

Inicialmente me gustaría agradecer en estas líneas a mis padres, por sus esfuerzos, cariño, apoyo y comprensión brindada en cada una de las etapas de este proceso.

A todos mis amigos, compañeros y colegas que estuvieron ahí brindándome su amistad sincera, apoyo y consejos para lograr este objetivo.

Finalmente a toda la planta docente del programa de Ingeniería Electrónica, por el apoyo, comprensión y todas las enseñanzas brindadas, en especial a mi tutor el Dr. Oscar Barrero Mendoza, por su orientación, paciencia, apoyo y comprensión hasta último momento en el desarrollo del presente trabajo de grado.

# Resumen

El departamento del Tolima es un gran referente agrícola para el país, por ello surge la necesidad de emplear nuevas tecnologías para optimizar los procesos agrícolas y con ello incrementar la rentabilidad del mismo. Es por esto que el presente trabajo de grado desarrolla un sistema de seguimiento de trayectorias para vehículos con arquitectura skid-steering empleando control predictivo, con el fin de tener una herramienta tecnológica que brinde al sector agrícola la automatización de tareas como la siembra, tomar muestras localizadas y tareas de laboreo primario/secundario. El sistema cuenta con un filtro de Kalman para estimar la posición y dirección georreferenciadas del vehículo en espacios abiertos, el cual permite fusionar la información de posición y dirección de una unidad de medición inercial (IMU), encoders y GPS (Global Positioning System) instalados en el vehículo. De otro lado, para el seguimiento autónomo de trayectorias, se diseñaron dos controladores: 1) control de velocidad lineal: con base a un controlador proporcional en función de la distancia del vehículo con respecto a la trayectoria, y 2) control de dirección: con base a un control predictivo en función de la dirección de vehículo con respecto a la dirección de la trayectoria. El diseño del controlador predictivo (MBPC) se realizó con base en el modelo matemático dinámico de dirección de un vehículo tipo skid-steering y sus parámetros se estimaron utilizando datos reales de entrada y salida. Los resultados prácticos de esta implementación, muestran que la estrategia desarrollada presenta buen desempeño con errores promedio en el seguimiento de una trayectoria en exteriores menores a un metro, algo que para aplicaciones en agricultura es prometedor.

**Palabras clave:** Vehículos autónomos, robot tipo skid-steering, seguimiento de trayectorias, control predictivo basado en modelo, filtro de Kalman.

## Abstract

The department of Tolima is a great agricultural reference for the country, so there is a need to use new technologies to optimize agricultural processes and thus increase the profitability of it. That is why this work develops a system of tracking trajectories for vehicles with skid-steering architecture using predictive control, in order to have a technological tool that provides the agricultural sector automation of tasks such as planting, taking samples located and primary / secondary tillage tasks. The system has a Kalman filter to estimate the georeferenced position and direction of the vehicle in open spaces, which allows merging the position and direction information of an inertial measurement unit (IMU), encoders and GPS (Global Positioning System) installed in the vehicle. On the other hand, for the autono-

---

mous tracking of trajectories, two controllers were designed: 1) linear speed control: based on a proportional controller as a function of the distance of the vehicle from the trajectory, and 2) direction control: based on a predictive control as a function of the direction of the vehicle from the direction of the trajectory. The design of the predictive controller (MBPC) was made based on the mathematical dynamic model of direction of a vehicle type skid-steering and its parameters were estimated using real data input and output. The practical results of this implementation show that the developed strategy presents good performance with average errors in tracking an outdoor trajectory of less than one meter, something that for applications in agriculture is promising.

**Keywords:** Autonomous vehicles, skid-steering vehicle, track following, model based predictive control, Kalman Filter.

# Contenido

<b>Resumen .....</b>	<b>VII</b>
<b>Lista de figuras .....</b>	<b>XI</b>
<b>Lista de tablas .....</b>	<b>XII</b>
<b>Introducción.....</b>	<b>1</b>
<b>1 Vehículos autónomos terrestres</b>	<b>4</b>
1.1 Marco Teórico . . . . .	4
1.1.1 Modelo robot Skid-Steered . . . . .	4
1.1.2 Modelo Cinemático . . . . .	6
1.1.3 Filtro de Kalman . . . . .	7
1.1.4 Modelo Dinámico . . . . .	8
1.1.5 Control Predictivo por Modelo . . . . .	8
1.1.6 ROS . . . . .	11
1.2 Antecedentes . . . . .	13
<b>2 Materiales y Equipos</b>	<b>16</b>
2.1 Ubiquity . . . . .	17
2.2 Descripción del Nivel gráfico de computo de ROS . . . . .	18
2.3 Sensores en ROS . . . . .	19
2.3.1 rc_control . . . . .	19
2.3.2 Roboteq . . . . .	20
2.3.3 xsens_mti_driver . . . . .	20
2.3.4 GPS . . . . .	21
2.4 Nodos de software en ROS . . . . .	23
2.4.1 Odom . . . . .	23
2.4.2 IMU . . . . .	24
2.4.3 Kalman_Filter . . . . .	24
2.4.4 Main_Control . . . . .	24
<b>3 Metodología</b>	<b>26</b>
3.1 Implementación del Filtro de Kalman . . . . .	26
3.2 Diseño del controlador de trayectoria . . . . .	30

<i>CONTENIDO</i>	<i>CONTENIDO</i>
3.2.1 Generador de Trayectoria . . . . .	30
3.2.2 Modelo Dinámico . . . . .	32
3.2.3 Control de trayectoria . . . . .	38
<b>4 Resultados</b>	<b>41</b>
4.1 Filtro de Kalman y Control de Trayectoria . . . . .	41
<b>5 Conclusiones y Recomendaciones</b>	<b>44</b>
5.1 Conclusiones . . . . .	44
5.2 Recomendaciones . . . . .	45
<b>Bibliografia</b> .....	<b>49</b>

# **Lista de Figuras**

1.1	Marco de referencia local ( $X_L, Y_L$ ) y global del robot ( $X_g, Y_g$ ). . . . .	5
1.2	Algoritmo del Filtro de Kalman. . . . .	7
1.3	La diferencia entre el control PID y el MPC. . . . .	9
1.4	Robotic Operating System . . . . .	11
2.1	Vehiculo Tipo Rover de la Universidad de Ibagué. . . . .	16
2.2	Receptor SPEKTRUM AR8000. . . . .	19
2.3	RoboteQ driver. . . . .	20
2.4	Xsens MTi-30. . . . .	20
2.5	GPS con magnetómetro 3D Robotics. . . . .	21
2.6	Mapa de nodos y tópicos del sistema. . . . .	23
2.7	Encoder Autonics E50S. . . . .	23
2.8	Señales de PWM . . . . .	25
3.1	Proyección de la trayectoria . . . . .	30
3.2	Puntos intermedios de la trayectoria . . . . .	31
3.3	Planteamiento del modelo matemático. . . . .	32
3.4	Diagrama de bloques del modelo de dirección del vehículo. . . . .	33
3.5	PWM-Ciclo util. . . . .	34
3.6	Prueba para la estimación de parámetros 1. . . . .	35
3.7	Prueba para la estimación de parámetros 2. . . . .	35
3.8	Prueba para la estimación de parámetros 3 . . . . .	35
3.9	Ajuste del modelo con datos de entrenamiento para calcular $a$ , $b$ y $K$ . . . . .	36
3.10	Validación del modelo con los parámetros estimados prueba #1. . . . .	36
3.11	Índice de error de la estimación #1 . . . . .	37
3.12	Validación del modelo con los parámetros estimados prueba #2. . . . .	37
3.13	Índice de error de la estimación #2 . . . . .	37
3.14	Respuesta impulso del modelo. . . . .	38
3.15	Lazos de control. . . . .	39
3.16	Validación en simulación del Controlador. . . . .	40
4.1	Trayectoria recorrida por el vehículo en metros (m). . . . .	41
4.2	Señal de control del sistema y seguimiento de la dirección de vehículo. . . . .	42
4.3	Aceleraciones lineales y velocidad angular del vehículo . . . . .	42
4.4	Trayectoria recorrida en UTM. . . . .	43

# **Lista de Tablas**

2.1	Sentencia NMEA GPRMC. . . . .	21
2.2	Sentencia NMEA GPGGA. . . . .	22
3.1	RMSE de las pruebas realizadas. . . . .	38

# Introducción

El departamento del Tolima ha sido un gran referente agrícola, dada su variedad climática y riqueza en sus tierras, con ello se deja expuesto el potencial que tiene para convertirse en un proveedor de productos agrícolas a nivel nacional e internacional, pues goza de grandes extensiones de suelo aptas para el cultivo de productos como arroz, maíz y café, de vital importancia en la económica de exportación en nuestro país [1] .

Según un artículo publicado por la revista de ciencias agrícolas [2], Colombia es un país poco competitivo y con baja participación en el mercado mundial, en el artículo se evidencia que entre los aspectos que generan la baja participación es el bajo nivel de tecnificación en sus procesos, con ello se ve afectada su estructura de costos. De lo anterior, se manifiesta la necesidad de usar herramientas tecnológicas que ayuden al agricultor a hacer un uso óptimo de su cultivo, reduciendo el impacto ambiental y aumentando su rentabilidad. Razones por las cuales surge el concepto de agricultura de precisión como se menciona en [3], donde el uso de vehículos eléctricos aéreos y terrestres no tripulados son parte fundamental para el éxito de esta nueva metodología en el manejo agronómico de los cultivos.

Uno de los aspectos importantes a mejorar dentro del manejo agronómico de un cultivo es el uso eficiente del terreno, donde los vehículos autónomos terrestres son una alternativa importante, puesto que pueden desarrollar tareas con alta precisión como lo son: sembrar semillas, muestrear el suelo, llevar a cabo labores de monitoreo y realizar surcos con precisión, entre otras. Para realizar esta clase de tareas los vehículos necesitan un sistema de navegación autónoma de alta precisión, en donde la ingeniería de control automático juega un papel importante.

El grupo de investigación D+TEC con su semillero D+TAP ha desarrollado en los últimos años un prototipo de vehículo autónomo tipo Rover con un sistema de seguimiento de trayectorias basado en controladores PID [4], en este proyecto se plantea seguir avanzado en el desarrollo del prototipo haciendo uso de técnicas avanzadas de control, con el fin de mitigar tendencias del vehículo y defectos de fabricación a través de optimización y robustez presentes en técnicas de control avanzado como el control predictivo. Dicho control de trayectoria emplea una unidad de medición inercial (IMU), magnetómetro, GPS y odometría. Hacer uso de técnicas de control avanzado como el control predictivo, se fundamenta en las ventajas que tiene sobre los sistemas de control tradicionales. La tesis de maestría [5] menciona que el PID conoce solo lo que ha sucedido, osea trabaja con datos del pasado y ajusta el controlador basado en la medición de error; mientras que el controlador predictivo

(MPC) sigue su modelo y valor actual medido, lo anterior se ve reflejado en la robustez y la estabilidad del sistema como lo divulga Valencia-Palomo, G., & Rossiter, J. A en [6], haciendo la comparación entre un controlador PI sintonizado automáticamente, un controlador predictivo y un controlador predictivo funcional, dicha comparativa arroja como resultado al controlador predictivo como el sistema de mayor rendimiento y el único que se ajusta a todos los sistemas presentados en el documento.

Vale la pena mencionar el uso técnicas de control avanzado para el seguimiento de trayectorias, como lo presenta el artículo [7], donde se desarrolla un control predictivo con base en el modelo, para un automóvil urbano. En el cual evalúan el rendimiento del controlador bajo distintas velocidades del vehículo, presentando resultados favorables en robustez y velocidad del control predictivo. Con base en lo mencionado anteriormente, proyecto pretende evaluar el desempeño de técnicas modernas de control en espacio de estados para el seguimiento de trayectorias en espacios abiertos, de igual forma brindar una herramienta robusta capaz de realizar tareas que ayuden al campesino y agricultor de la región.

## **Objetivo General**

Diseñar e implementar un sistema de seguimiento de trayectoria para vehículos con arquitectura skid-steering, empleando control predictivo.

## **Objetivos Específicos**

- Desarrollar nodos de software en ROS para los elementos de percepción, acción y control del Robot.
- Obtener un modelo en espacio de estado lineal a partir de técnicas de identificación de sistemas lineales.
- Diseñar e implementar un filtro de Kalman con base en el modelo en espacio de estado identificado en ROS.
- Diseñar, implementar y validar en campo de un controlador predictivo para seguimiento de trayectorias en ROS.



# **Capítulo 1**

## **Vehículos autónomos terrestres**

Los vehículos autónomos terrestres, son aquellos capaces de desplazarse a través de un terreno de forma independiente, llevando sensores a bordo que permiten determinar la posición, velocidad, orientación del vehículo y la detección de objetos [8]. El uso de todos o algunos de los sensores anteriormente mencionados depende de la aplicación o enfoque que tenga el vehículo.

Los vehículos autónomos terrestres tienen un sistema que se divide en dos capas: en primer lugar se encuentra el hardware; allí se encuentran los sensores, actuadores del vehículo y la unidad de procesamiento y control, que brindan la información y datos necesarios para la capa de software. En la zona de software se realizan tareas de percepción para reconocer el entorno del vehículo y la localización del mismo. Hay que mencionar también la planificación y el control como elementos fundamentales del software, pues la planificación se encara de decidir que movimiento hacer, cual ruta seguir y el control es el encargado de ejecutar dichas acciones sobre los actuadores [9].

### **1.1. Marco Teórico**

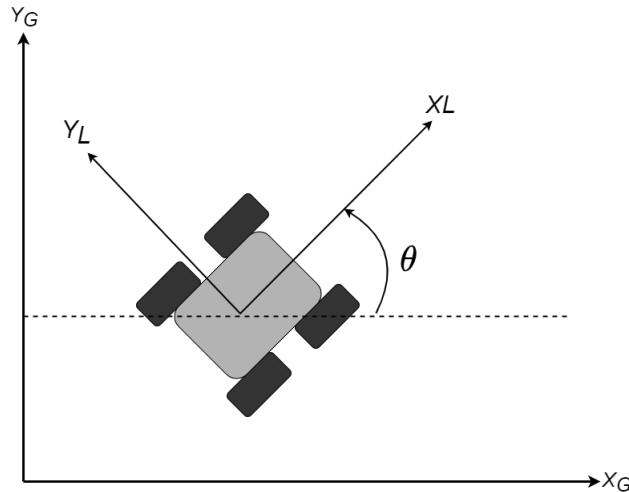
Como se mencionó en la anterior sección los vehículos autónomos terrestres están compuestos por una capa de software y otra de hardware, a continuación se exponen en su mayoría los componentes de software que tendrá el presente proyecto, permitiendo al lector comprender los elementos básicos para que un vehículo terrestre sea autónomo.

#### **1.1.1. Modelo robot Skid-Steered**

La posición de los robots móviles es definida por la cinemática del mismo expresada en un modelo matemático, el cual representa la configuración del robot. En vehículos terrestres la configuración mas usada es de tracción diferencial, debido a que es un sistema sencillo y adecuado para entornos cotidianos no muy exigentes. Su direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales [10], [11]. De la configuración diferencial se desprenden otras tipologías las cuales permiten a los vehículos tener un mayor rendimiento

en terrenos exigentes como lo son los robots de dirección deslizante.

En la actualidad existen diversas configuraciones en los robots móviles terrestres, entre ellas se encuentran del tipo dirección deslizante o skid-steered, dicha configuración es empleada en vehículos que son expuestos a terrenos exigentes o exteriores en general, pues se caracterizan por una estructura mecánica robusta y una gran maniobrabilidad [12].



**Figura 1.1:** Marco de referencia local ( $X_L, Y_L$ ) y global del robot ( $X_g, Y_g$ ). Fuente: Autor

Como se observa en la figura 1.1 tomada de [13] se observa que el modelo cinemático de un vehículo skid-steering se describe a partir de las siguientes ecuaciones:

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{(\omega_L + \omega_R)r}{2} \\ 0 \\ \frac{(-\omega_L + \omega_R)r}{2Y_G} \end{bmatrix} \quad (1.1)$$

Donde  $V_x$  es la velocidad de avance del vehículo,  $\omega$  es la velocidad angular,  $r$  es el radio de las ruedas,  $Y_G$  es la ubicación del centro de rotación instantáneo,  $\omega_L$  y  $\omega_R$  son las velocidades angulares de las ruedas derecha e izquierda respectivamente. Como se está trabajando con tiempos de muestreo se procede a convertir a tiempo discreto el modelo 1.1, teniendo en cuenta a [14], se obtiene que:

$$\Delta_{S_k} = T_s \frac{(\omega_{R_k} + \omega_{L_k})r}{2} \quad (1.2)$$

$$\Delta_{\theta_k} = T_s \frac{(\omega_{R_k} - \omega_{L_k})r}{2Y_G} \quad (1.3)$$

Donde  $\Delta_{S_k}$  es el incremento de la distancia recorrida en metros del vehículo en  $k$  y  $\Delta_{\theta_k}$  es el desplazamiento angular del vehículo en el instante  $k$ . De este modo se logra estimar la posición del robot a partir de dichos diferenciales.

$$\begin{bmatrix} X_k \\ Y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} X_{k-1} \\ Y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta_{S_{k-1}} \cos \left( \theta_{k-1} + \frac{\Delta_{\theta_k}}{2} \right) \\ \Delta_{S_{k-1}} \sin \left( \theta_{k-1} + \frac{\Delta_{\theta_k}}{2} \right) \\ \Delta_{\theta_k} \end{bmatrix} \quad (1.4)$$

### 1.1.2. Modelo Cinemático

La orientación del vehículo esta definida por yaw ( $\psi$ ), pitch ( $\theta$ ), y roll ( $\phi$ ), ángulos de Euler que se estiman utilizando las velocidades angulares obtenidas de la IMU. Las medidas de velocidad angular se representan en el marco local. Las aceleraciones del vehículo también se expresan en el marco local, Por otro lado, la velocidad y los datos de posición obtenidos de los encoders y GPS se expresan en el marco inercial [15]. Para convertir las aceleraciones del marco local al marco inercial es necesario multiplicarlas por la matriz de rotación que esta dada de la siguiente forma:

$$R'_B = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta c\psi & c\phi s\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

Para el presente trabajo se usó únicamente como orientación yaw ( $\psi$ ), dado que es el ángulo que apunta en la dirección a la que se mueve el vehículo. Razón por la cual los valores de pitch ( $\theta$ ) y roll ( $\phi$ ) se definen nulos. Como resultado la matriz de rotación al sistema inercial se reduce a la siguiente forma:

$$r'_B = \begin{bmatrix} c\psi & -s\psi \\ s\psi & c\psi \end{bmatrix} \quad (1.5)$$

La unidad de medición inercial (IMU) provee 3 entradas al estimador de estado, las aceleraciones del vehículo en  $A_{x_L}, A_{y_L}$  y la velocidad angular ( $\omega$ ) en el marco de referencia local:

$$A_L = [A_{x_L}, A_{y_L}]^T \quad (1.6)$$

$$\omega_L = [\omega_{z_L}] \quad (1.7)$$

Teniendo en cuenta la velocidad angular y las aceleraciones provenientes de la IMU, los vectores de estado y de entrada al estimador de estado se definen como:

$$x_k = [P_x, P_y, \psi, V_x, V_y]^T \quad (1.8)$$

$$u_k = [\omega_{z_L}, A_{x_L}, A_{y_L}]^T \quad (1.9)$$

### 1.1.3. Filtro de Kalman

El Filtro de Kalman es una herramienta matemática desarrollada en 1960 por Rudolf E. Kalman con el fin de solucionar el problema del filtrado lineal de datos discretos [16]. El algoritmo de Kalman se puede emplear en la identificación de las variables de estado de un sistema dinámico lineal, el cual apoya las estimaciones de las variables de estado pasadas, presentes e incluso futuras, y así mismo puede hacerlo cuando se desconoce la naturaleza precisa del sistema modelado [17]. Las aplicaciones del filtro de Kalman abarcan muchos campos, como se menciona el libro *Kalman Filtering: Theory and Practice Using Matlab* [18], esta herramienta tiene dos propósitos:

- **Estimación de estado en sistemas dinámicos:** El filtro de Kalman permite la estimación óptima de las variables de estado en sistemas dinámicos, asumiendo que estos poseen ruido de proceso y medición con distribución Gaussiana. Lo que quiere decir que dada la información de las variables de entrada y salida provenientes del sistema, el algoritmo de Kalman es capaz de estimar el valor de las variables de estado, en física y en teoría de control se le conoce como observador de estado.
- **Análisis del rendimiento de los sistemas de estimación:** El objetivo del análisis de diseño es determinar la mejor manera de utilizar diferentes tipos de sensores para un conjunto determinado de criterios de rendimiento, a partir de ello se obtiene una mayor precisión en la etapa de predicción del algoritmo de Kalman.

El Filtro de Kalman es un algoritmo que procesa información de forma estructurada. El presente algoritmo fue diseñado para ser implementado en un ordenador, por ello debe trabajar siguiendo una serie de instrucciones como se muestra en la figura 1.2.

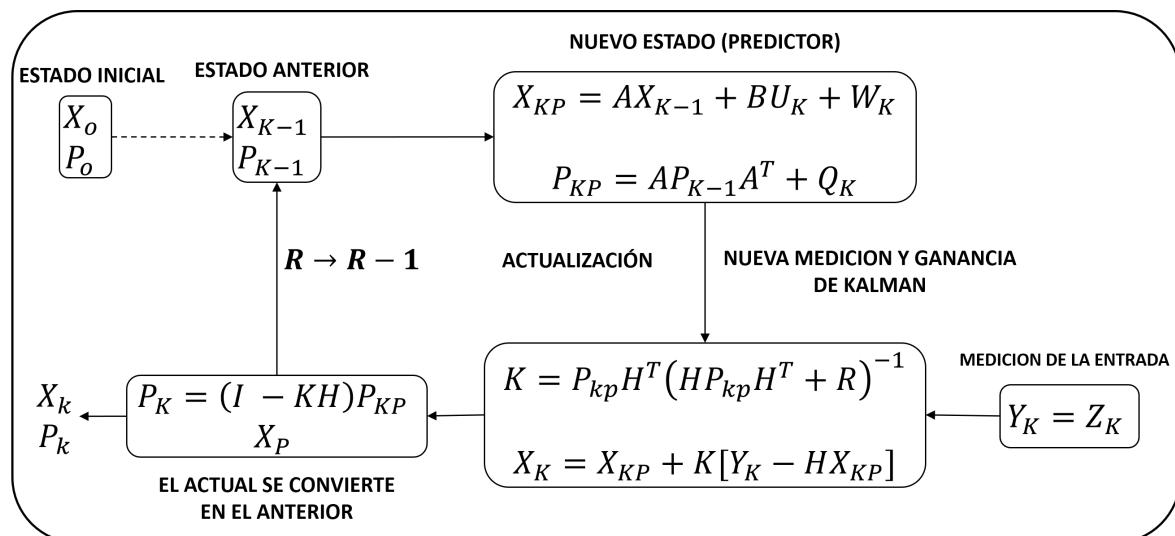


Figura 1.2: Algoritmo del Filtro de Kalman. Fuente: Autor

### 1.1.4. Modelo Dinámico

El modelo dinámico es un pilar fundamental al momento de implementar cualquier técnica de control. En la literatura se encuentra una gran variedad de métodos para obtener dicho modelo, estos pueden ser a partir de la identificación de todas sus variables físicas o la medición de las mismas. Existen procesos que dada su complejidad, determinar el valor de dichas variables resulta en un proceso de gran complejidad, por lo que se emplean técnicas de modelamiento en caja gris o caja negra.

El modelo matemático en caja gris emplea las ecuaciones diferenciales que caracterizan el sistema, pero dadas las dificultades para conocer sus parámetros, se procede al uso de herramientas para la estimación de los mismos, a partir de datos de entrada y salida, permitiendo observar la evolución del sistema y lograr una estimación de parámetros.

Por otro lado el método de modelamiento en caja negra se realiza a partir de la toma de datos de entrada y salida, observando la evolución del sistema y a partir de ello determinar un modelo matemático como lo indica [4] en su trabajo de grado en la sección identificación de modelos dinámicos.

Teniendo en cuenta lo anteriormente mencionado, MATLAB ofrece herramientas para realizar este tipo de moldeamientos, para el caso de caja negra, usualmente se emplea Ident y en versiones posteriores System Identification, encargado de determinar el modelo matemático a partir de datos de entrada y salida del sistema. En cuanto al modelamiento por caja gris, se recurre a la extensión de SIMULINK, Parameter Estimation, el cual identifica los parámetros del sistema, a través de datos de entrada y salida del mismo, empleando métodos estadísticos de desviación estándar de una distribución normal, funciones de optimización como gradiente descendente y mínimos cuadrados no lineales entre otros [19].

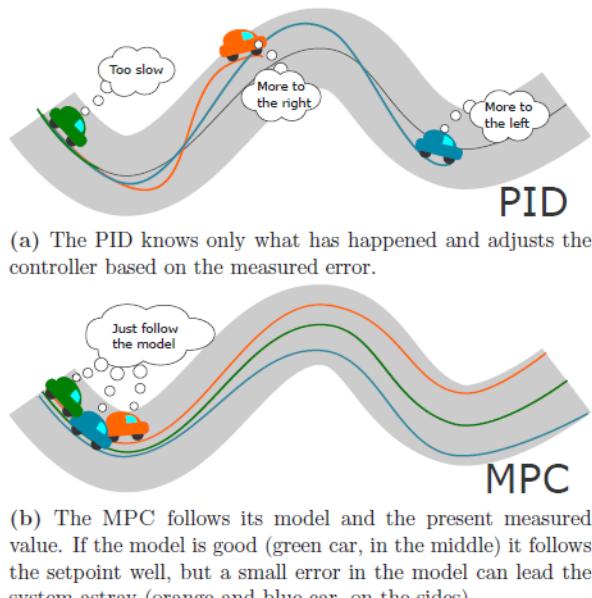
### 1.1.5. Control Predictivo por Modelo

El control predictivo, es un controlador compuesto por: Un horizonte móvil, en donde a partir de un modelo matemático interno y una estrategia de optimización, predice las salidas del sistema dentro de un intervalo de tiempo [20]. Este tipo de controladores incorpora restricciones y un modelo de sistema explícito usado para predecir la dinámica futura de la planta, como lo menciona el doctor Liuping Wang en su libro Model Predictive Control System Design and Implementation Using MATLAB [21].

El control predictivo basado en el modelo (MPC), presenta grandes ventajas como: (1) Evitar violaciones de las restricciones de entradas y salidas, (2) Conducir algunas variables de salida a su configuración óptima, al tiempo que se mantienen otras dentro de los rangos específicos, (3) Evitar el movimiento excesivo de las variables de entrada y (4) Controlar tantas variables de proceso como sea posible cuando un sensor o actuador no está disponible [22].

Se hace énfasis en los sistemas de control avanzado como lo es el control predictivo, tomando de apoyo la tesis de maestría de Ylva Lindberg [5]. En ella realiza una comparación entre

el control predictivo y el control PID, este ultimo amplia mente usado en la industria hoy en día. En sus resultados expresa aspectos en los que sobresale el control predictivo sobre el PID dado que su total funcionamiento se basa en el modelo, como se muestra en la figura 1.3



**Figura 1.3:** La diferencia entre el control PID y el MPC. Fuente: [5]

A partir de lo expuesto en libro Predictive Control with Constraints [23], en el capítulo titulado A basic formulation of predictive control. Se emplea la formulación en espacios de estado para un modelo lineal, descrito de la siguiente forma:

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) \quad (1.10)$$

Donde  $x(k)$  es el vector de estado,  $u(k)$  es el vector de entrada y  $y(k)$  es el vector de las variables de salida. Como se pudo observar en el espacio de estado 1.10, el modelo expresa el estado  $x_k$  en términos de los valores de entrada de  $u(k)$ . Pero la función de costo penaliza los cambios en la entrada,  $\Delta u$ , en lugar de los valores de entrada en si mismos. Por lo tanto, es conveniente para muchos fines considerar que el controlador produce la señal  $\Delta u$ , y la planta tiene esta señal como entrada. Es decir, es conveniente considerar un integrador en tiempo discreto de  $\Delta u$  a  $u$  incluido en la dinámica de la planta.

Existen varias maneras de incluir el integrador en un modelo de espacios de estado. Todos ellos involucran aumentar el vector de estado. Para ello el primer paso es definir el modelo incremental:

$$z(k+1) = Mz(k) + N\Delta u(k) \quad (1.11)$$

$$y(k) = Qz(k) \quad (1.12)$$

Una vez definida la estructura del modelo aumentado, se procede a realizar el espacio de estado aumentado con el integrador embebido, el cual viene dado a partir de las siguientes expresiones:

$$\begin{aligned} \overbrace{\begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix}}^{z(k+1)} &= \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_M \overbrace{\begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}}^{z(k)} + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_N \Delta u(k) \\ y(k) &= \underbrace{\begin{bmatrix} C & 0 \end{bmatrix}}^Q \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \end{aligned} \quad (1.13)$$

Se le llama modelo incremental, debido a que las variables en el vector de estado aumentan. Como se puede observar el vector de estado aumentado ahora posee una variable mas  $u(k)$ , del mismo modo se observa el incremento en la matriz dinámica  $M$  y en el vector de entrada  $B$ . Hecha esta salvedad y siguiendo con nuestro análisis, la estimación del vector de salida  $y(k)$  resulta en la siguiente ecuación:

$$\hat{y}(k+j) = QM^j\hat{z}(k) + \sum_{i=0}^{j-1} QM^{j-i-1}N\Delta u(k+i) \quad (1.14)$$

La ecuación 1.14 es la predicción obtenida en función del modelo aumentado, donde  $\hat{z}(k)$  es el vector de estado aumentado estimado,  $M, N$  y  $Q$  son las matrices aumentadas mencionadas anteriormente. Dicha ecuación se puede resumir a la siguiente expresión:

$$y = F\hat{z}(k) + Hu \quad (1.15)$$

Donde  $y$  es un vector que contiene todas las estimaciones del instante  $k$  hasta el horizonte de predicción  $Np$ . En el caso de  $F$  es un vector que contiene el producto de  $QM^j$  y  $j$  es del tamaño de  $Np$ , lo mismo sucede para el caso de  $Hu$ , el cual describe la expresión  $QM^{j-i-1}N\Delta u(k+i)$  de forma matricial.

$$F = \begin{bmatrix} QM \\ QM^2 \\ \vdots \\ QM^{Np} \end{bmatrix} H = \begin{bmatrix} QN & 0 & \cdots & 0 \\ QMN & QN & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ QM^{Np-1} & QM^{Np-2}N & \cdots & QN \end{bmatrix} u = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+Np-1) \end{bmatrix}$$

Llegados a este punto, se da a conocer la función de costo, para el calculo de la acción de control de un sistema en espacio de estado aumentado sin restricciones, la cual viene dada de la siguiente forma:

$$J = (Hu + F\hat{z}(k) - r)^T R_w (Hu + F\hat{z}(k) - r) + u^T Q_w u \quad (1.16)$$

De la función de costo descrita en 1.16, los elementos desconocidos son,  $r$  que representa la referencia,  $R_w$  es una matriz de pesos que hace énfasis en el seguimiento fielmente de la referencia y  $Q_w$  es otra matriz de pesos, que afecta directamente la acción de control  $u$  haciendo que la misma sea de menor o mayor valor dependiendo del peso que se fije. Para llevar la función de costo a una acción de control  $u$ , se tiene que minimizar  $J$  en función de  $u$ , obteniendo la posterior representación:

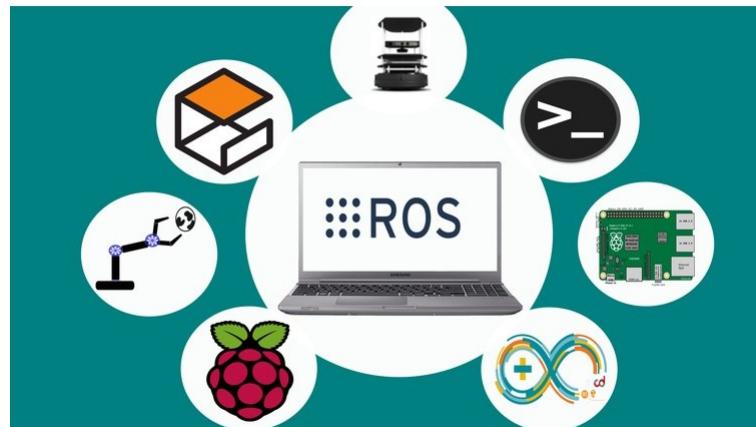
$$\min_u J \quad u = (H^T R_w H + Q_w)^{-1} H^T R_w (r - F\hat{z}(k)) \quad (1.17)$$

$$K = (H^T R_w H + Q_w)^{-1} H^T R_w \quad (1.18)$$

Después de minimizar  $u$  de tal forma que satisfaga  $J$ , se obtuvo la acción de control  $\Delta u$ . De lo anterior hay que aclarar que se toma solo la primer fila  $K(1, :)$ , que correspondiente a la acción de control.

$$\Delta u(k) = K(1, :)(r - F\hat{z}(k)) \quad (1.19)$$

### 1.1.6. ROS



**Figura 1.4:** Robotic Operating System. Fuente: [24]

Robot Operating System (ROS) es un entorno de trabajo flexible para desarrollar software para robots. Como se comenta en [25], ROS se compone por un conjunto de herramientas y librerías, que tienen como objetivo simplificar la tarea de crear un sistema robótico complejo y robusto, mediante una amplia variedad de plataformas robóticas.

ROS contiene herramientas y facilidades estándar de un sistema operativo como lo menciona A. Martínez y E. Fernández en su libro [26], entre ellas cabe destacar la abstracción de hardware, control de dispositivos de bajo nivel, flujo de datos entre los procesos y gestión de paquetes. Esto es basado en una arquitectura gráfica con una topología centralizada donde

los procesos toman lugar en nodos que pueden publicar o recibir información de múltiples sensores, lazos de control, estados del sistema, actuadores entre otros. Actualmente este framework funciona de manera estable en plataformas basadas en Unix, como es el caso de Ubuntu y Debian

En el presente proyecto se utilizó Ubiquity Robotics el cual es un sistema operativo basado en Ubuntu Xenial 16.04, con el fin de emplearse en dispositivos como la Raspberry pi. Este tipo de software es optimizado para el hardware de una placa computadora<sup>1</sup>. Por otro lado Ubiquity Robotics viene con ROS pre instalado, una ventaja al momento de querer soporte técnico y actualizaciones en el dispositivo.

Se opta por emplear el framework del sistema operativo robótico (ROS) en el presente trabajo de grado ya que cuenta con una diversidad de aplicaciones y herramientas permitiendo la unión de las distintas áreas que componen el proyecto. Algunas utilidades se mencionan en el artículo [27], como lo son:

- Un nodo central que cumple la función de coordinación denominado *roscore*
- Multi-lenguaje, dado que se diseña para las distintas preferencias de usuario, como también la compensación entre tiempo de programación, facilidad de depuración, sintaxis, eficiencia en el tiempo de ejecución entre otras.
- Permite el flujo de datos mediante nodos de comunicación nombrados tópicos a través de distintos tipos de mensajes. Dichos nodos pueden ser programados en Python o C++.
- Cuenta con un middleware<sup>2</sup> que permite ejecutar nodos simultáneamente, manipular y controlar la frecuencia publicaciones de los mensajes, medir el ancho de banda usado, imprimir la información y el tipo de los mensajes, entre otros.
- Emplea un reloj global, que permite conocer el instante de tiempo en el que se ejecuta dicho nodo en el sistema, como también obtener una sincronía entre tópicos.
- Es gratis y de código abierto, facilitando la depuración en todos los niveles de software, como también

Actualmente existe una diversidad de empresas que emplean ROS en sus robots o software, con el fin de realizar tareas en ambientes industriales igualmente en entornos controlados como lo son laboratorios, mesas de trabajo y software de simulación. Cabe mencionar algunas de ellas visto que sus aportes permiten el desarrollo exitoso de trabajos relacionados, como el presente trabajo de grado. Entre ellas están:

- **Sony:** Con el robot mascota Aibo robot dog

---

<sup>1</sup> Ordenador de placa reducida, en inglés: Single Board o SBC, se refiere a una computadora completa en un solo circuito impreso

<sup>2</sup> Middleware es un software que se encuentra en el medio de un sistema operativo y las aplicaciones, permitiendo la interacción entre si [28].

- **Clearpath Robotics:** Compañía canadiense que desarrolla vehículos móviles basados en ROS
- **Fetch Robotics:** Empresa acargo del desarrollo de robots móviles para la investigación rebotica y transporte de materiales.
- **Pal Robotics:** Compañía Española conocida por su trabajo en robots humanoides.
- **Robotnik:** Fabricante a cargo de la creación y diseño de robots manipuladores móviles y vehículos terrestres no tripulados de diferentes tipos.
- **Yujin Robots:** Compañía coreana especializada en robots vacuum cleaning.
- **Robotis:** Empresa conocida en el ámbito por sus servos Dynamixel y en ROS por el diseño del robot Turtlebot 3.
- **Shadow Robot:** Fundada en Londres, Shadow Robot es una empresa dedicada al desarrollo de manos humanoides roboticas.
- **Husarion:** Es una empresa polaca que vende robots móviles autónomos simples y compactos llamados ROSbots.
- **Neobotix:** Fabricante de robots para una amplia gama de aplicaciones industriales como el transporte de material.
- **Gaitech:** Es una compañía china que se dedica principalmente a la distribución de robots ROS, y productos ROS en genera
- **Ubiquity Robotics:** Grupo de trabajo encargado del desarrollo de robots de aprendizaje y de un sistema operativo con ROS nativo.

## 1.2. Antecedentes

Cando a vehículos autónomos se refiere, Tesla es una compañía referente en el mercado de vehículos autónomos urbanos, en el cual su icónico TESLA Model S resalta por la capacidad de mantener una navegación autónoma activa desde la rampa de entrada a la de salida de una autopista, conservar la distancia y velocidad con el vehículo procedente, hacer un cambio de carril. También cuenta con un sistema que permite al automóvil acudir automáticamente hasta donde se encuentre su propietario y finalmente su habilidad de aparcamiento automático paralelo y perpendicular con un simple botón [29].

Para fines del presente proyecto, el sistema de seguimiento de trayectorias autónomo se enfoca en el sector agrícola, en el cual distintas compañías como Case IH con su tractor autonomo ACV destacan por contar con un sistema de guiado automático AccuGuide, con una geolocalización a partir del Case IH RTK+GPS para obtener un guiado de alta precision. La funcion del mencionado tractor es realizar actividades como la siembra y el laboreo primario/secundario [30]. Con respecto al punto anterior, otra compañía que ha presentado

un sistema de guía autónomo es John Deere [31], que cuenta con sistemas de geolocalización RTK para lograr una precisión crítica en operaciones sobre terreno, también posee un sistema que reduce el solapamiento de la trayectoria cuando se hace un giro o pasada sobre el terreno, además de un control de velocidad y asistencia de giro entre otras funciones que brinda la compañía.

Siguiendo con el tema de tractores autónomos, CNH Industrial junto con Autonomous Solutions Incorporated, diseñaron el NEW HOLLAND NHDdrive [32], una máquina sin conductor capaz de llegar al campo de forma autónoma a través de pistas privadas de la granja. Vale la pena resaltar que este vehículo puede ser monitoreado y controlado a través de una computadora de escritorio o mediante una interfaz gráfica en una tablet o portátil. Asimismo, se puede mencionar, además que el tractor NHDdrive sigue caminos optimizados en el campo, que son generados automáticamente por el software, después de haber tenido en cuenta el tamaño y la forma del campo, cualquier obstáculo preexistente y el ancho del implemento a utilizar.

KUBOTA también se unió a la elaboración de vehículos autónomos con fines agrícolas, la compañía japonesa presentó su Prototipo de tractor eléctrico autónomo, que consta de un sistema de conducción autónoma completamente no tripulada, también con algoritmos de inteligencia artificial y bases de datos que realizan y eligen tareas optimizadas a partir de la información meteorológica y datos sobre el cultivo. No está de más mencionar la batería de litio recargable montada y paneles solares que posee el robot de KUBOTA, logrando con ello un funcionamiento silencioso y sin emisión de gases [33].

YANMAR lanzó el proyecto Yanmar Robot Tractor [34]. Es un tractor que posee un sistema autónomo integrado, el mismo puede identificar obstáculos en su camino evitando colisiones, lo cual permite al operario manejar dos tractores a través de una interfaz de funcionamiento intuitivo. Así mismo cuenta con una unidad de control en los motores, la cual optimiza automáticamente la velocidad de desplazamiento en función de las condiciones del terreno, cabe agregar que dispone de una cámara en la cabina del piloto, permitiendo ver las condiciones de funcionamiento con una tablet, a esto se le suma un sistema de GPS+RTK, brindando una posición del tractor bastante precisa.

Dot Technology Corp es una empresa Canadiense la cual diseño un tractor autónomo que se acopla a una sembradora. Así creó el concepto de plataforma para que el tractor trabaje ensamblado con diferentes herramientas. El vehículo dispone de una navegación la cual opera dentro de las rutas prescritas y aprobadas por los agricultores, que se generan mediante la simple creación de límites muy precisos. La información de posición de un receptor GPS RTK montado en Dot asegura que siempre está operando dentro del área aprobada. Si Dot se desvía de su trayectoria, detendrá el movimiento y enviará un mensaje al operador [35].

En el artículo [36] presenta un controlador MPC cinemático junto con una prueba analítica de su estabilidad en el cual se comprueba la robustez del controlador frente a la dinámica no modelada. Al mismo tiempo los autores demuestran que el perfil de velocidad elegido para el subsistema angular permite acotar el horizonte de predicción necesario para resolver el

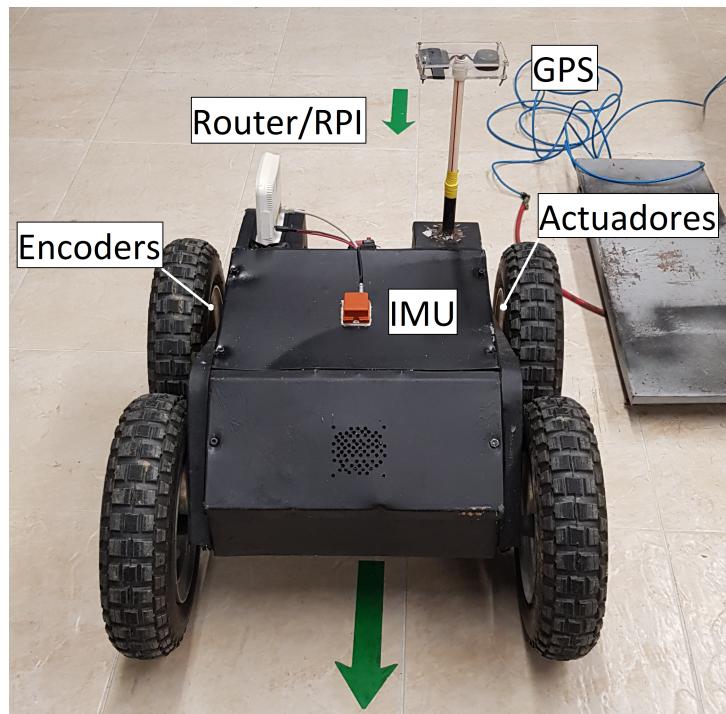
problema de optimización. Esto supone una mejora con respecto a enfoques anteriores, ya que simplifica la prueba de estabilidad e implica la viabilidad de la estabilización en tiempo finito bajo las condiciones dadas, comentadas en el presente documento.

En la tesis de maestría [37] se describe un seguimiento de trayectoria en un vehículo de tipo Skid Steering usando un controlador predictivo con base en el modelo (MPC) con un aprendizaje de modelos en linea. El modelo de velocidad, que representa la relación entre la velocidad real del vehículo y el comando de entrada, se aprende con un proceso gaussiano disperso (GP) en línea. El GP disperso en línea puede reducir la complejidad computacional del GP y también actualizar constantemente el modelo a partir de los datos de conducción. Finalmente, la combinación con el MPC permite generar una política óptima basada en el modelo aprendido. Dados los experimentos realizados por los autores argumentan que los resultados muestran un rendimiento más fiable que el método basado en el modelo convencional con adaptación de parámetros.

# Capítulo 2

## Materiales y Equipos

En el presente trabajo de grado se emplearon diversos sensores y elementos fundamentales para la percepción, acción y control del vehículo, para el desarrollo de dicha actividad se usó como herramienta el framework ROS, puesto que cuenta con características como abstracción de hardware, control de dispositivos de bajo nivel, flujo de datos entre los procesos y gestión de paquetes [26], lo que se traduce a una herramienta robusta y flexible para el desarrollo del proyecto.



**Figura 2.1:** Vehículo Tipo Rover de la Universidad de Ibagué. Fuente: Autor

El semillero de investigación D+TEC cuenta con un vehículo tipo rover, como se muestra en la figura 2.1, su tipología es de configuración Skid-Steering. Su estructura se basa en un armazón metálico el cual posee en sus laterales dos ruedas de 40 cm de diámetro, ademas tiene

dos motores DC de 12V que generan el movimiento de las 4 ruedas de manera sincronizada, esto se logra a partir de una cadena que une cada par lateral de ruedas permitiendo el movimiento de forma simultanea, en síntesis el sentido de giro de las ruedas delanteras depende de las ruedas traseras. Por otra parte cuenta con un grupo de periféricos y herramientas que servirán para el seguimiento de trayectorias de manera autónoma, los cuales son:

- Raspberry pi 3B
- UBlox GPS 3DR
- Imu Xsens MTI-30
- Encoder Autonics E50s
- Driver RoboteQ HDC2450
- Control RC SPEKTRUM DX6i

## 2.1. Ubiquity

Cada periférico es fundamental en el desarrollo del proyecto ya que son los necesarios para la navegación del vehículo. Inicialmente se instala Ubiquity Robotics como sistema operativo en la Raspberry pi 3B. Su selección se debe a que el sistema operativo de Ubiquity Robotics es basado en Ubuntu 16.04, ademas de ello viene con ROS pre instalado, listo para cualquier robot que use una Raspberry pi, brindando un software robusto y accesible a actualizaciones.

En la raspberry pi 3 se efectuaran varios procesos de manera simultanea; Adquisición de datos seriales, lectura de señales digitales, generación de señales pwm y la ejecución de algoritmos en Python y C++, entre otros. En vista de que se llevaran acabo numerosas tareas y procesamiento de datos, se desactiva la capa gráfica del sistema operativo, ya que en ninguna actividad sera necesaria, con el fin de optimizar el trabajo del procesador de la placa base.

Una vez instalado y puesto en marcha Ubiquity dentro de la raspberry pi, se procede a la creación del espacio de trabajo, el cual es una carpeta donde se encuentran los paquetes, sus respectivos códigos, el compilador y los ejecutables. Al espacio de trabajo usualmente se le denomina *catkin\_ws*, este cuenta con una estructura específica y comandos puntuales para trabajar con los paquetes basado en catkin<sup>1</sup>.

<sup>1</sup>Catkin es el sistema de construcción oficial de ROS y el sucesor de rosbld el cual permite la combinación de macros de CMake y scripts de Python para proporcionar cierta funcionalidad, ademas del flujo de trabajo normal de CMake. Catkin es muy similar a CMake, pero soporte para la infraestructura automática de 'encontrar el paquete' y la construcción de múltiples proyectos dependientes al mismo tiempo. [38]

## 2.2. Descripción del Nivel gráfico de computo de ROS

ROS genera una red mediante la cual sus procesos se conectan entre si, a través de un mecanismo que se basa en la publicación y suscripción de mensajes. Los nodos tienen la posibilidad de acceder a dicha red e interactuar entre ellos, observando la información que se está publicando en la red y transmitiendo datos a ella [39].

Tomando como referencia el libro Learning ROS for Robotics Programming [26] y el proyecto de grado de Nickson Garcia y Cristian Molina [39], los conceptos básicos en este nivel son los nodos, el maestro, el servidor de parámetros, los mensajes, servicios, tópicos y bolsas, todos los cuales proporcionan datos a la red de diferentes maneras:

- **Nodos:** Los nodos o nodes en inglés, son aquellos que realizan el proceso computacional necesario para brindar información a la red a través de tópicos; se pueden clasificar en suscriptores, publicadores o ambos. Cuando se necesita que un proceso pueda interactuar con otros nodos, se crea un nodo con dicho proceso para conectarlo a la red que genera ROS. Usualmente, un sistema dispone de varios nodos para controlar diversas funciones. Es recomendable tener varios nodos encargados de una funcionalidad en lugar de un solo nodo haciendo el trabajo de todo el sistema. Los nodos pueden ser escritos en distintos lenguajes de programación y así comunicarse entre ellos. Comúnmente se usa la librería “roscpp” para programarlos en C++ y “rospy” para hacerlo en Python.
- **Maestro:** El Maestro o Master, brinda el registro de nombres y la búsqueda del resto de los nodos. Si el master no se encuentra ejecutando en el sistema, no puede haber comunicación entre los nodos, servicios, mensajes y otros. Como se menciona en [39] en la sección Capa de Software del Sistema, cuando un nodo se inicia, lo primero que hace es buscar al master y registrarse en él con el nombre del nodo. De este modo, el master obtiene los detalles de los nodos que se encuentran activos en el sistema.
- **Servidor de parámetros:** En ROS el servidor de parámetros o Parameter Server, no da la posibilidad de tener los datos almacenados usando llaves en una ubicación central. Con este parámetro se pueden configurar los nodos mientras están ejecutándose o cambiar su funcionamiento.
- **Mensajes:** Los mensajes son la forma mediante la cual los nodos se comunican entre sí. Un mensaje contiene datos que envían información a otros nodos. ROS posee diversos tipos de mensajes como también permite desarrollar tu propio tipo de mensaje empleando mensajes estándar.
- **Tópicos:** Cada mensaje debe tener un nombre para ser enrulado en la red que genera ROS. Cuando un nodo está enviando datos, decimos que está publicando un tópico. Los nodos pueden recibir datos de otros tópicos simplemente suscribiéndose a ellos. Es importante que el nombre del tópico sea único para evitar problemas y confusión entre los tópicos con el mismo nombre.

- **Servicios:** Cuando se publican tópicos, se están enviando datos de manera muy variada, pero cuando se necesita una solicitud o una respuesta de un nodo, no se puede hacer a través de los tópicos. Los servicios brindan la posibilidad de interactuar con los nodos. Además, los servicios deben tener un nombre único. Cuando un nodo tiene un servicio, todos los nodos pueden comunicarse con él, gracias a las bibliotecas de cliente de ROS.
- **Bolsas:** Las bolsas o Bags en inglés, son un formato para guardar y reproducir los datos publicados por los tópicos en ROS. Los bags son un mecanismo importante para almacenar datos, como los datos de sensores, que pueden ser difíciles de recopilar pero son necesarios para desarrollar y probar algoritmos.

## 2.3. Sensores en ROS

Ya dicho todo lo referente a lo que es ROS, se procede a emplearlo en el presente proyecto. El software del Rover está distribuido en una red de ROS compuesta por ocho nodos que realizan tareas de acción, percepción y control a partir de los sensores que componen el vehículo.

### 2.3.1. rc\_control



**Figura 2.2:** Receptor SPEKTRUM AR8000. Fuente: [40]

El nodo `rc_control` es el encargado de hacer la lectura de las señales digitales entregadas por el receptor AR8000 de spketrum que se observa en la figura 2.3.1. Este nodo hace uso de la librería pigpio, para hacer una captura de las variaciones de la señal en los pines de la raspberry Pi. Las mediciones adquiridas provienen de los canales del control remoto AX1, AX2, steering y throttle

### 2.3.2. Roboteq



**Figura 2.3:** RoboteQ driver. Fuente [41]

Este nodo recibe la información proveniente del driver RoboteQ, los datos que entrega son fundamentales para la odometria, en ellos se encuentran el contador de pulsos de los encoders izquierdo y derecho. Por otro lado el nodo RoboteQ se encarga de configurar el driver en el modo mixto que esta disponible como una de las opciones en los controladores de doble canal, donde el canal 1 se usa para mover el robot hacia adelante o atrás, y el canal 2 para dirigir y cambiar la dirección del robot, y así el RototeQ reciba estas dos señales directamente del control. Se seleccionó el modo mixto que permite dar prioridad al giro, ya que teniendo en cuenta la cinemática propia del vehículo no puede alcanzar ciertas velocidades angulares con una velocidad lineal muy alta [4].

### 2.3.3. xsens\_mti\_driver



**Figura 2.4:** Xsens MTi-30. Fuente: Autor

El nodo xsens\_mti\_driver Se trata de un driver ofrecido por el fabricante XSENS, el cual permite la configurar el puerto a donde se conectara la IMU, su baudrate, el modelo del dispositivo, la frecuencia de trabajo y los datos de salida que se desean. En el caso del presente proyecto se configura de modo tal que publique las aceleraciones lineales en  $x,y,z$ , la velocidad angular en  $z$  y el angulo de orientación  $yaw$

### 2.3.4. GPS



**Figura 2.5:** GPS con magnetómetro 3D Robotics. Fuente [42]

GPS es el nodo encargado de la adquisición de datos por medio de comunicación serial, haciendo la lectura del puerto USB de la raspberry pi, al cual llegan la información del gps de la compañía 3DR 2.5. El dispositivo transmite los datos en formato NMEA (National Marine Electronics Association) con distintas sentencias o frases interpretadas como lo son: \$ GPRMC y \$ GPGGA. La sentencia \$ GPRMC traduce, datos mínimos específicos recomendados de GPS/tránsito, dicho formato presenta la información de la siguiente manera:

\$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E\*68

En relación a la información presentada y tomando como referencia la descripción de Glenn Baddeley en [43], el significado de cada uno de los componentes del mensaje, se ve reflejado en la siguiente tabla:

**Tabla 2.1:** Sentencia NMEA GPRMC. Fuente:Autor

Message Component	Description
225446	Time of fix 22:54:46 UTC
A	Navigation receiver warning A = OK, V = warning
4916.45,N	Latitude 49 deg. 16.45 min North
12311.12,W	Longitude 123 deg. 11.12 min West
000.5	Speed over ground, Knots
054.7	Course Made Good, True
191194	Date of fix 19 November 1994
020.3,E	Magnetic variation 20.3 deg East
*68	mandatory checksum

La sentencia \$GPGGA traduce, "Datos fijos del sistema de Posicionamiento Global". El mismo entrega un mensaje con la siguiente estructura:

\$GPGGA,HHMMSS.SS,DDMM.MMMMMM,K,DDDMM.MMMMMM,L,N,QQ,P.p, A.A, M, G.G, M, SSS,RRRR\*C

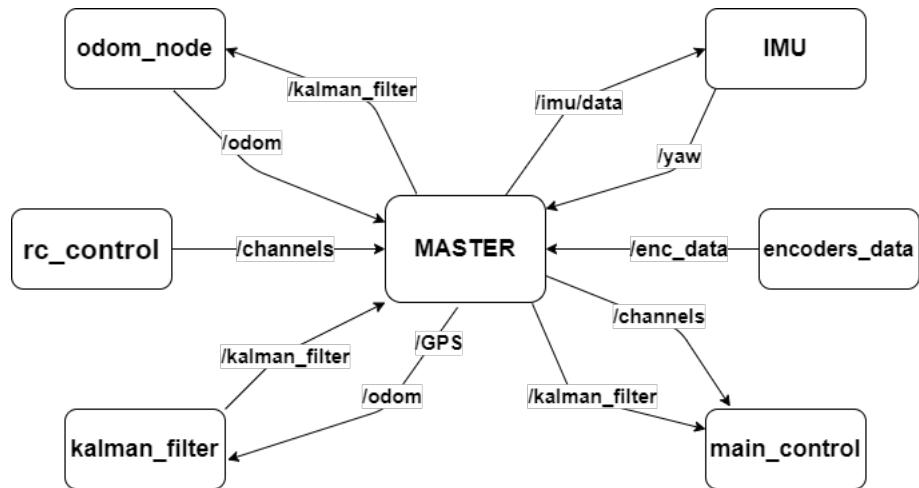
El concepto de cada elemento del mensaje proveniente del GPS se observa en la tabla 2.2 teniendo en cuenta lo comentado en [4]:

**Tabla 2.2:** Sentencia NMEA GPGGA. Fuente:Autor

<b>Message Component</b>	<b>Description</b>
HHMMSS.SS	UTC of Position
DDMM.MMMMMM	Latitude in degrees, minutes, and decimal minutes
K	Latitude indicator; value is N (North latitude) or S (South latitude)
DDDMM.MMMMMM	Longitude in degrees, minutes, and decimal minutes
L	Longitude indicator; value is E (East longitude) or W (West longitude)
N	GPS quality indicator (0=invalid; 1=GPS fix; 2=Diff. GPS fix)
QQ	Number of satellites used in position solution
P.P	Horizontal dilution of precision (HDOP)
A.A	Antenna altitude, in meters, re: mean-sea-level (geoid)
M	Units of antenna altitude (M = meters)
G.G	Geoidal separation (in meters)
M	Units of geoidal separation (M = meters)
SSS	Age of differential corrections, in seconds
RRRR	Differential reference station ID
*CC	Checksum

Una ves obtenidas las coordenadas geográficas de la sentencia \$GPGGA, se procede al calculo del posicionamiento global del robot en  $x$  y  $y$ , empleando las ecuaciones de Coticchia-Surace [44] para el problema directo, el cual trata de conversión de coordenadas geográficas al sistema de coordenadas universal transversal de Mercator (UTM).

## 2.4. Nodos de software en ROS



**Figura 2.6:** Mapa de nodos y tópicos del sistema. Fuente: Autor

El software del robot esta distribuido en una red de ROS constituida por cuatro nodos principales, los cuales se comunican con el nodo máster que se ejecutan en su totalidad en la Raspberry Pi. Lo anterior mencionado da como resultado el siguiente diagrama de nodos y tópicos que se muestran en la figura 2.6. Los algoritmos y rutinas de cada uno de los nodos se pueden encontrar en el repositorio GitHub realizado por el autor [45].

### 2.4.1. Odom



**Figura 2.7:** Encoder Autonics E50S. Fuente: [46]

La función del nodo Odom consiste en determinar la posición del vehículo a partir de los desplazamientos entregados por los encoders Autonics E50S 2.7, los cuales poseen una resolución de hasta 1500 pulsos por revolución . Se usaron dos encoders, uno para cada motor y con ello hacer obtener los desplazamientos en  $X$  y  $Y$  a partir de las siguientes ecuaciones:

$$\Delta s = \frac{\Delta s_r - \Delta s_l}{2} \quad (2.1)$$

$$\Delta \theta = \frac{\Delta s_r - \Delta s_l}{2yg} \quad (2.2)$$

$$\Delta x = \Delta s * \cos\left(\theta + \frac{\Delta\theta}{2}\right) \quad (2.3)$$

$$\Delta y = \Delta s * \sin\left(\theta + \frac{\Delta\theta}{2}\right) \quad (2.4)$$

- $\Delta s_r$  : Es el desplazamiento en metros en un instante de tiempo recorrido por el lateral derecho del vehículo
- $\Delta s_l$  : Es el desplazamiento en metros en un instante de tiempo, recorrido por el lateral izquierdo del vehículo
- $\Delta\theta$  : Hace referencia al desplazamiento direccional del vehículo en un lazo de tiempo producido por la diferencia entre  $\Delta s_r$  y  $\Delta s_l$ .

#### 2.4.2. IMU

En el nodo IMU se reciben los datos provenientes del sensor xsens, entre ellos estan las aceleraciones lineales en  $x,y$  y la velocidad angular en  $z$ . A las aceleraciones se le aplica un filtro butterworth de segundo orden y a la velocidad angular, el mismo pero de primer orden, esto se hace con el fin de disminuir el ruido de medición proveniente de las irregularidades del terreno. En el nodo Imu también aplica la matriz de rotación a las aceleraciones para convertirlas al plano inercial previamente explicado en 1.1.2.

#### 2.4.3. Kalman Filter

Kalman\_Filter es el nodo encargado de realizar de la estimación de posición del vehículo en  $X,Y$  y su orientación  $\theta$ , por medio de un Filtro de Kalman. Dicho calculo se resuelve con mayor detalle en el capítulo 3.1. Hecha esta salvedad volvemos a la función del nodo, el cual publica un mensaje de tipo *Odometry*, permitiendo ubicar ademas de la posición global del vehículo, las aceleraciones lineales y la velocidad angular del mismo.

#### 2.4.4. Main\_Control

En el nodo Main\_Control se desarrolla el código principal del vehículo en el, se encuentran tres modos de trabajo, inicialmente el modo STOP encargado de detener o desactivar el movimiento del robot. El siguiente modo de trabajo es el manual, en el cual se aplica la señal PWM proveniente del control RC a los motores del vehículo, permitiendo desplazarlo al lugar de trabajo. Finalmente es este el modo automático, en el se emplea el controlador predictivo y el seguimiento de trayectorias de manera autónoma. En el presente nota inicialmente se pregunta por el valor digital del canal CH2 del control remoto SPEKTRUM DX6i, pues dependiendo del valor así mismo se asignara el modo de trabajo del vehículo como se muestra a continuación:

**Algorithm 1** Modos de trabajo

---

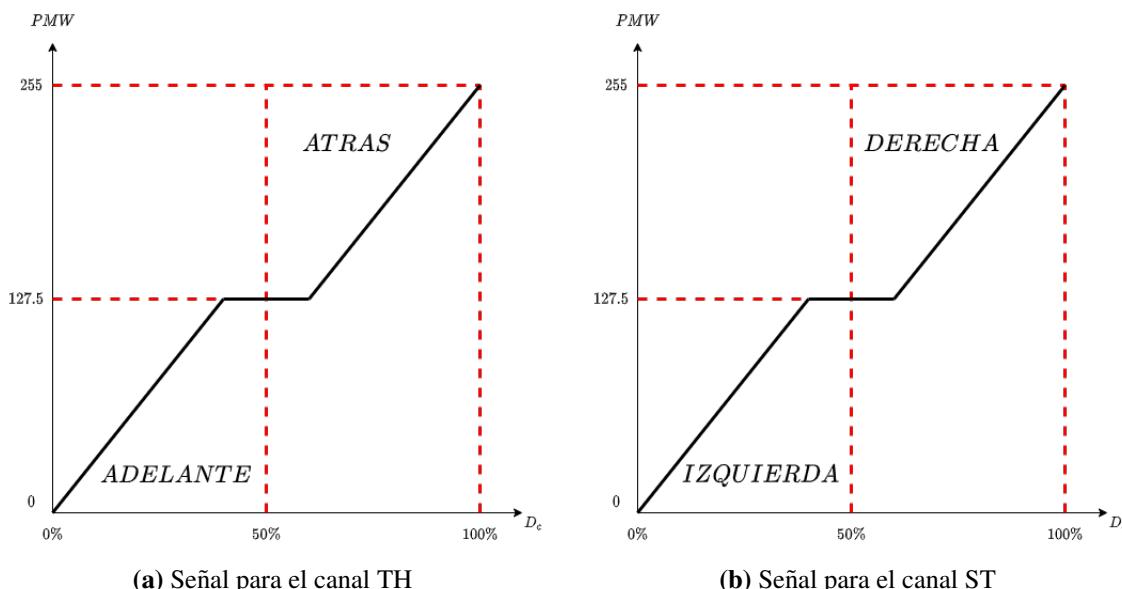
```

procedure MODO(valor)
    if 1900 < valor < 2000 then                                ▷ Llega el valor del modo de trabajo
        stop()                                              ▷ Se pausa totalmente el vehículo
    else if 1400 < valor < 1600 then
        manual()                                         ▷ El vehículo se puede operar manualmente
    else if 900 < valor < 1000 then
        automatico()                                     ▷ Se activa el seguimiento de trayectoria

```

---

Lo que anteriormente se expreso en pseudo código quiere decir, que si el canal designado para determinar el modo de trabajo se encuentra entre 1900 y 2200, el código se va a dirigir a la función stop, la cual detendrá por completo el vehículo, ahora, si el modo de trabajo se encuentra entre 1400 y 1600, el algoritmo se posicionara en la función manual, en donde el vehículo se moverá dependiendo del ancho de pulso que transmitan los canales CH0 y CH1 del control remoto, finalmente, si el modo esta entre 900 y 1000, se ejecutara la función automática, que contiene el control de seguimiento de trayectorias.

**Figura 2.8:** Señales de PWM. Fuente: Autor

# Capítulo 3

## Metodología

### 3.1. Implementación del Filtro de Kalman

Como se menciono en la sección 1.1.3, el filtro de Kalman es una herramienta matemática que sirve como estimador de las variables de estado en sistemas dinámicos. En el presente proyecto, la herramienta brindada por Rudolf E.Kalman cumple la función de estimador de posición del vehículo Rover. Para ello es necesario generar un modelo matemático que determine la cinemática del robot. En la figura 1.2, se observa que la ecuación del estado predictor es la siguiente:

$$x_{kp} = Ax_{k-1} + Bu_k + W_k \quad (3.1)$$

La ecuación 3.1 expresa el modelo matemático en espacio de estados a partir de la cinemática del robot. Donde  $A$  es la matriz de estados,  $x_{k-1}$  es el vector de estado,  $B$  la matriz de entrada,  $u_k$  el vector de entradas y  $w_k$  matriz de ruido de estado, para este caso no se tendrá en cuenta por lo que su valor es 0. El modelo del filtro de Kalman se basa en las siguientes ecuaciones:

$$w_k = \frac{\theta_k - \theta_{k-1}}{T} \quad (3.2)$$

$$\begin{aligned} \frac{d}{dt}v(t) &= Acc(t) \rightarrow \int \frac{d}{dt}v(t) = \int Acc(t) \\ v(t) &= \int Acc(t) \end{aligned} \quad (3.3)$$

$$P(t) = \int v(t) \quad (3.4)$$

Donde  $T$  es el tiempo de muestreo del filtro de Kalman el cual se determina teniendo en cuenta la frecuencia en la que llegan los datos provenientes de los sensores, por lo que su valor depende del sensor mas lento del sistema.  $w_k$  es la velocidad angular estimada por el filtro. Para la cinemática del robot se desea estimar la posición en  $P_{x_k}, P_{y_k}$  el ángulo de orientación  $\theta_k$  y las velocidades en  $V_{x_k}, V_{y_k}$  del vehículo. Teniendo en cuenta las ecuaciones 3.2,3.4 y 3.3, las variables de estado quedan de la siguiente forma:

$$Px_k = T^2 Acc_k + Px_{k-1} - T^2 bx_k \quad (3.5)$$

$$Py_k = T^2 Acc_k + Py_{k-1} - T^2 by_k \quad (3.6)$$

$$\theta_k = Tw_k + \theta_{k-1} \quad (3.7)$$

$$Vx_k = TAccx_k + Vx_{k-1} \quad (3.8)$$

$$Vy_k = TAccy_k + Vy_{k-1} \quad (3.9)$$

En las ecuaciones de posición 3.5-3.6 se encuentran los términos  $bx_k$  y  $by_k$  que hacen referencia al bias de las aceleraciones de la IMU. Se opta por estimar dicho término pues se torno de gran complejidad extraer este a partir del software de XSENS, por tal motivo se agrega como variable al modelo del filtro de Kalman para mejorar el desempeño del mismo. Esta técnica tiene el nombre de modelo de caminante aleatorio en el cual se busca estimar los parámetros constantes del sistema [47]. Para la cinemática del Rover, hay que tener en cuenta que las aceleraciones deben estar en el marco de referencia global con el vehículo, lo cual se logra mediante la multiplicación de las mismas por la matriz de rotación, previamente explicado en la sección 1.1.2.

$$A_g = \begin{bmatrix} c\psi & -s\psi \\ s\psi & c\psi \end{bmatrix} \cdot \begin{bmatrix} Accx_k \\ Accy_k \end{bmatrix} \quad (3.10)$$

Obteniendo las aceleraciones en el marco de referencia global o inercial:

$$A_g = [Acc_{x_g}, Acc_{y_g}] \quad (3.11)$$

Ya definidas las variables de estado, se continuó con el desarrollo del modelo matemático para el filtro de Kalman en espacio de estados descrito de la siguiente forma:

$$\underbrace{\begin{bmatrix} x_k \\ Px_k \\ Py_k \\ \theta_k \\ Vx_k \\ Vy_k \\ bx_k \\ by_k \end{bmatrix}}_{\text{A}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & T & 0 & -T^2 & 0 \\ 0 & 1 & 0 & 0 & T & 0 & -T^2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{B}} \underbrace{\begin{bmatrix} x_{k-1} \\ Px_{k-1} \\ Py_{k-1} \\ \theta_{k-1} \\ Vx_{k-1} \\ Vy_{k-1} \\ bx_{k-1} \\ by_{k-1} \end{bmatrix}}_{\text{C}} + \underbrace{\begin{bmatrix} 0 & T^2 & 0 \\ 0 & 0 & T^2 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{D}} \underbrace{\begin{bmatrix} w_k \\ Accx_{k_g} \\ Accy_{k_g} \end{bmatrix}}_{U_k} \quad (3.12)$$

En el segmento de predicción de estado, el algoritmo de Kalman tiene una expresión que representa el error en la estimación de estado, lo anterior se expresa a partir de la siguiente formulación:

$$P_{kp} = AP_{k-1}A^T + Q_k \quad (3.13)$$

Donde  $A$  es la matriz de estados comentada anteriormente,  $P_{k-1}$  es el error de estimación y  $Q_k$  es la matriz de covarianza del modelo. Para la predicción se recomienda iniciar la matriz  $P_{k-1}$  en un valor grande, dado que el error tiene que tender a 0, esta matriz se expresa de la siguiente forma:

$$P_{k-1} = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \quad (3.14)$$

La matriz de covarianza del ruido de proceso  $Q_k$  es una matriz que expresa la incertidumbre de las variables de estado del modelo dinámico del sistema, la cual se puede determinar analizando el ruido de las señales de entrada y la robustez propia del modelo. Teniendo en cuenta los aspectos anteriores, los valores de la matriz se ajustaron a prueba y error, primero en el laboratorio con datos reales y luego en campo, donde se hizo un ajuste final resultados confiables.

$$Q_k = \begin{bmatrix} 0,2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0,01 \end{bmatrix} \quad (3.15)$$

Antes de seguir con la sección de la actualización del vector de estado, en filtro de Kalman hay un bloque en el que se realiza la nueva medición y con ello calcular el error de predicción. En la figura 1.2 se observa la siguiente ecuación:

$$Y_k = Z_k \quad (3.16)$$

Donde  $Z_k$  es vector de mediciones al momento  $k$  dichas mediciones son las posiciones  $X, Y$  provenientes del gps, las posiciones  $X, Y$  estimadas a partir de la odometria y los ángulos de orientación obtenidos a partir del desplazamiento de las ruedas y del angulo de orientación que entrega la IMU. A partir de las variables mencionadas el vector medición queda la siguiente manera:

$$Y_k = \begin{bmatrix} Px_{gps} \\ Py_{gps} \\ \theta_{imu} \\ Px_{odom} \\ Py_{odom} \\ \theta_{odom} \end{bmatrix} \quad (3.17)$$

En el apartado de actualización del vector de estado se divide en dos partes, inicialmente con el calculo de la ganancia de Kalman y posterior a ello, la actualización del vector de estado. Inicialmente el calculo de la ganancia de Kalman viene dado por la siguiente expresión:

$$K = P_{kp}H^T (HP_{kp}H^T + R)^{-1} \quad (3.18)$$

Donde  $P_{kp}$  es el error en la estimación de estado, como ya se había mencionado anteriormente,  $H$  es la matriz que permite la fusión de los datos del vector de medición  $Y_k$  y  $R$  es la matriz de covarianza de los sensores o variables medidas. Continuando en nuestro análisis, se desea a la salida del filtro la posición  $X$ ,  $Y$  y el angulo de orientación  $\theta$ , para ello es necesario adecuar la matriz  $H$  de tal modo que al multiplicar el vector de estado con el vector medición, el resultado sea una matriz de dimensiones 3x1, por tal motivo la matriz  $H$  es de la siguiente forma:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (3.19)$$

Por ultimo para finalizar con el calculo de la ganancia de Kalman, hay que definir la matriz  $R$  donde se encuentra la covarianza del ruido de las mediciones, su calculo se realizo dejando las variables del vector de estado midiendo en el mismo lugar, sin mover el vehículo, durante un periodo de tiempo, después se saca la varianza de cada una de ellas y se ubica en la matriz  $R$  del siguiente modo:

$$R = \begin{bmatrix} 0,3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,03 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,03 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,01 \end{bmatrix} \quad (3.20)$$

Con la ganancia de Kalman definida, se continua al calculo del vector de estado corregido, el cual es la suma de la predicción con el error de medición multiplicado por la ganancia de Kalman. De forma matemática se expresa así:

$$x_k = x_{kp} + K [Y_k - Hx_{kp}] \quad (3.21)$$

En la ecuación 3.21 se observa a  $X_{kp}$  el cual es el vector de estado predictor mencionado con anterioridad,  $K$  que es la ganancia de Kalman,  $Y_k$  que es el vector medición, la matriz  $H$  la cual va a permitir la relación entre el vector de estado y la medición y  $X_{kp}$  como la predicción del vector de estado. Con lo anteriormente mencionado es que se realiza la corrección del vector de estado. Continuando en el análisis y tomando como referencia el diagrama expuesto en la figura 1.2, el algoritmo de kalman finaliza con el calculo del error de predicción, como se muestra a continuación:

$$P_k = (I - KH) P_{kp} \quad (3.22)$$

Ya calculados  $x_k$  y  $P_k$  estos valores se actualizan convirtiéndose en  $x_{k-1}$  y en  $P_{k-1}$  iniciando nuevamente el ciclo en el algoritmo filtro de Kalman.

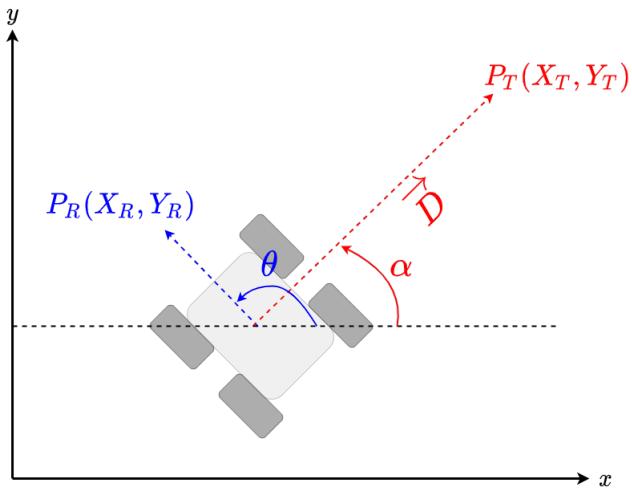
$$x_k = x_{k-1}$$

$$P_k = P_{k-1}$$

## 3.2. Diseño del controlador de trayectoria

### 3.2.1. Generador de Trayectoria

Para el diseño del generador de trayectoria se toma como referencia el artículo “Outdoors Trajectory Tracking Control for a Four Wheel Skid-Steering Vehicle” [13], en donde inicialmente se definen los puntos de camino de la trayectoria deseada, a continuación se determinan los puntos intermedios que indican al vehículo la trayectoria a seguir. Esto se determina considerando  $P_T$  como el punto al que debe llegar el vehículo y  $P_R$  la posición actual del robot.



**Figura 3.1:** Proyección de la trayectoria. Fuente: Autor

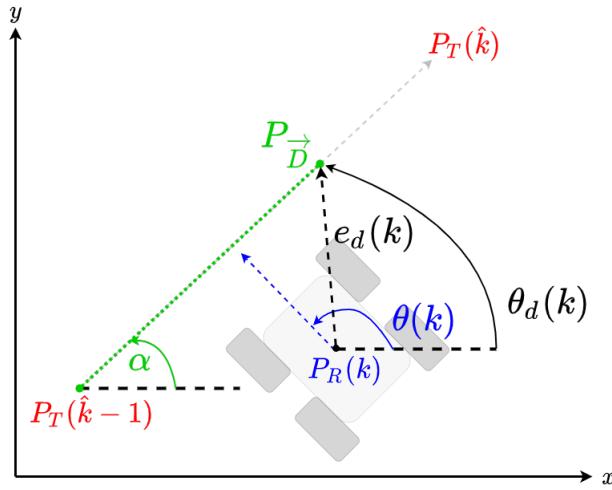
A partir de la figura 3.1 se observa que entre las posiciones  $P_R$  y  $P_T$  resulta vector con una magnitud correspondiente a la distancia que debe recorrer el vehículo y un angulo de dirección  $\alpha$  que describe la dirección de la trayectoria a seguir.

$$|\vec{D}| = \sqrt{|\Delta x|^2 + |\Delta y|^2} \quad (3.23)$$

$$\alpha = \tan^{-1} \left( \frac{\Delta y}{\Delta x} \right) \quad (3.24)$$

Donde  $\Delta x$  es la diferencia entre la posición de la trayectoria  $P_T$  y la posición del robot  $P_R$  en  $x$ , de igual forma sucede con  $\Delta y$ . Como el seguimiento de la trayectoria debe ser lo mas fiel al vector  $\vec{D}$ , se generan puntos intermedios en la trayectoria, esto permite al vehículo hacer un recorrido mas preciso sobre la misma. Consideremos ahora dicha trayectoria como una distribución de puntos en intervalos de tiempo  $\Delta t$ , en el cual se divide el vector en pequeños diferenciales de posición, dados por la velocidad en un intervalo de tiempo como el periodo de muestreo en dirección al vector inicial como se muestra en 3.25

$$\vec{P}_{T(k)} = V_d k T_s e^{j\alpha} + \vec{P}_{T(k-1)} \quad (3.25)$$



**Figura 3.2:** Puntos intermedios de la trayectoria. Fuente: Autor

A partir de la imagen 3.2 y la ecuación 3.25 se pueden hallar las posiciones de los puntos en los ejes del marco global, como se muestra a continuación:

$$P_{\vec{T},x}(k) = V_d T_s k \cos(\alpha) + x_T(\hat{k}-1) \quad (3.26)$$

$$P_{\vec{T},y}(k) = V_d T_s k \sin(\alpha) + y_T(\hat{k}-1) \quad (3.27)$$

Donde  $V_d$  es la velocidad de desplazamiento,  $T_s$  el tiempo de muestreo,  $k$  es un contador que se incrementa en cada muestra para aumentar la distancia al siguiente punto,  $x_T(\hat{k}-1)$  y  $y_T(\hat{k}-1)$  son los puntos iniciales de la trayectoria. El angulo deseado  $\theta_d$  presente en la figura 3.2, es la referencia del lazo de control de dirección de trayectoria, para su calculo se realiza a partir de la diferencia entre los puntos intermedios generados y la posición del vehículo de la siguiente forma:

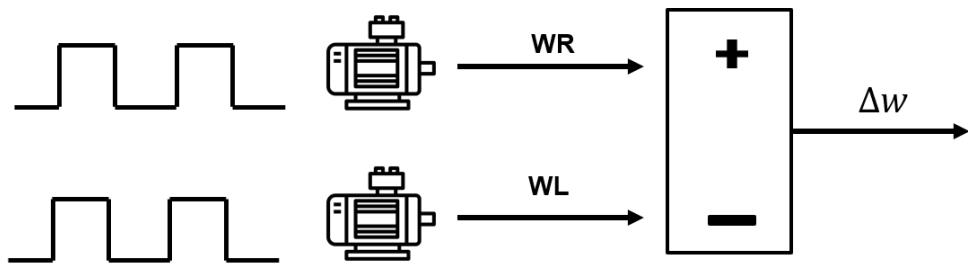
$$\theta_d(k) = \tan^{-1} \left( \frac{P_{\vec{T},y}(k) - P_{R,y}(k)}{P_{\vec{T},x}(k) - P_{R,x}(k)} \right) \quad (3.28)$$

La distancia entre el punto intermedio generado y el robot, se considera como el error para el lazo de control de velocidad lineal del vehículo, su calculo se da a partir de la siguiente expresion:

$$e_d(k) = \sqrt{|P_{\vec{T},x}(k) - P_{R,x}(k)|^2 + |P_{\vec{T},y}(k) - P_{R,y}(k)|^2} \quad (3.29)$$

### 3.2.2. Modelo Dinámico

Para realizar un control de trayectoria, surge la necesidad de establecer un modelo matemático que exprese la dinámica del vehículo para realizar dicho proceso. Inicialmente se establece la plataforma sobre la cual se aplicara control; con efectos del actual proyecto el vehículo a trabajar es un Rover con topología Skid-Steering, lo que nos lleva al siguiente planteamiento:



**Figura 3.3:** Planteamiento del modelo matemático. Fuente: Autor

En la figura 3.3 se observa un motor DC que tiene a su entrada una señal que generalmente es un voltaje  $V_a$ , lo cual produce a la salida del mismo una velocidad angular,  $w_r$  para el motor derecho y  $w_l$  para el motor izquierdo, las cuales multiplicadas por el radio de las ruedas se obtienen sus respectivas velocidades lineales. Si se realiza una diferencia de dichas velocidades lineales, se obtiene la variación de dirección angular del vehículo  $\omega$ , de acuerdo a la siguiente expresión:

$$\begin{aligned} \omega &= \frac{r}{2y_g} (w_r - w_l) \\ \omega &= \frac{r}{2y_g} \Delta w \end{aligned} \quad (3.30)$$

El termino  $r$  en la ecuación 3.30 hace referencia al radio de la rueda y  $y_g$  al centro instantáneo de rotación del robot. Definido  $\omega$ , se sigue adelante con la formulación del modelo dinámico del vehículo. En donde La literatura [48] nos muestra que el modelo matemático de un motor DC en función de la velocidad angular, se plantea de la siguiente forma:

$$\frac{\omega_m(s)}{V_a(s)} = \frac{k}{s^2 + as + b} \quad (3.31)$$

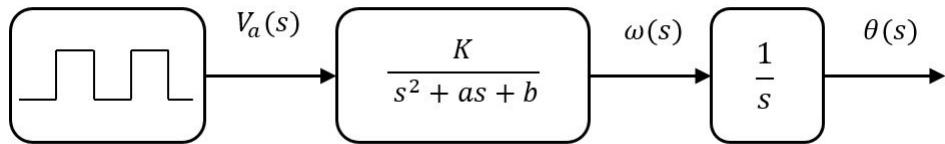
donde  $\omega_m$  es la velocidad angular del motor y  $V_a$  el voltaje de armadura. A partir del planteamiento que se presenta en la figura 3.3, se asume que los dos motores DC del vehículo cuentan con las mismas características físicas y eléctricas, lo cual permite emplear el modelo matemático de un motor DC para ambos ( $\omega_r, \omega_l$ ), de ello se obtiene la expresión genérica 3.32 que representa la velocidad angular de dirección del vehículo Skid Steering.

$$\omega = \frac{K}{s^2 + as + b} \quad (3.32)$$

Como se puede observar en la ecuación 3.32 el numerador cuenta con una variable  $K$  que contiene los valores de  $r$ ,  $y_g$  y la ganancia  $k$  presente en el modelo matemático del un motor DC presentes en la ecuación 3.31. Por tal motivo se simplifica el termino a una ganancia  $K$  con el fin de determinar la misma con ayuda del identificador de parámetros de MATLAB, lo cual se explica con mayor detalle posteriormente.

$$\omega = \frac{rk/2y_g}{s^2 + as + b} = \frac{K}{s^2 + as + b} \quad (3.33)$$

Identificado el modelo matemático de un motor DC, el siguiente paso es determinar como con el mismo se obtiene el angulo requerido para el seguimiento de trayectorias. En primer lugar cabe recordar que la salida de la ecuación 3.31 es una velocidad angular, para llevar esa velocidad angular del motor a un vehículo Skid-Steering, se multiplica por el radio de las ruedas a razón de dos veces el centro instantáneo de rotación, de allí se obtiene la expresión 3.30, pero el elemento de control es el angulo del vehículo. A partir de esa primicia se aplica un integrador continuo a la velocidad velocidad angular, el resultado del producto de ambos se obtiene el angulo de orientación del robot.



**Figura 3.4:** Diagrama de bloques del modelo del vehículo. Fuente: Autor

De acuerdo con la figura 3.4 y la ecuación 3.33 la función de transferencia que relaciona el ángulo de dirección del vehículo ( $\theta(s)$ ) con comando de giro del rover ( $V_a(s)$ ), se puede expresar como se muestra a continuación,

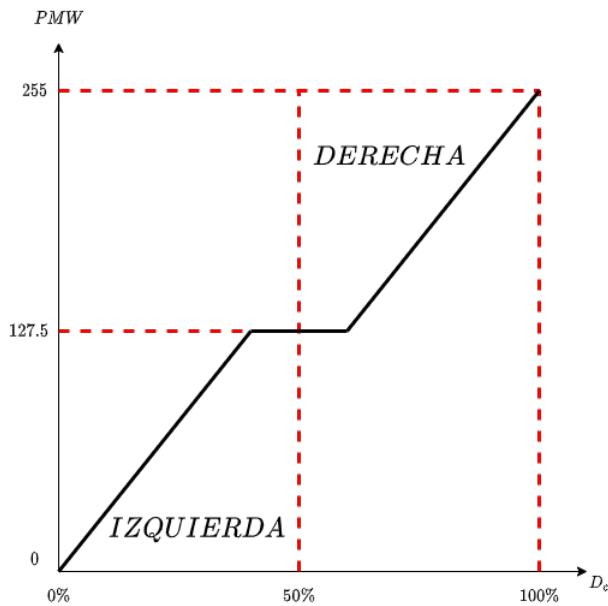
$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(s^2 + as + b)} \quad (3.34)$$

El modelo matemático expresado en función de transferencia de la ecuación 3.34 se pasa a espacio de estado con el fin de que el mismo quede adecuado para emplear un controlador predictivo con base en el modelo por espacios de estado. Dicha conversión se desarrolla de la forma canónica controlable. De ello nos queda el siguiente espacio de estado en tiempo continuo:

$$\frac{d}{dt} \begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \\ \theta \end{bmatrix} = \begin{bmatrix} -a & -b & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \\ \theta \end{bmatrix} + \begin{bmatrix} -K \\ 0 \\ 0 \end{bmatrix} u(t)$$

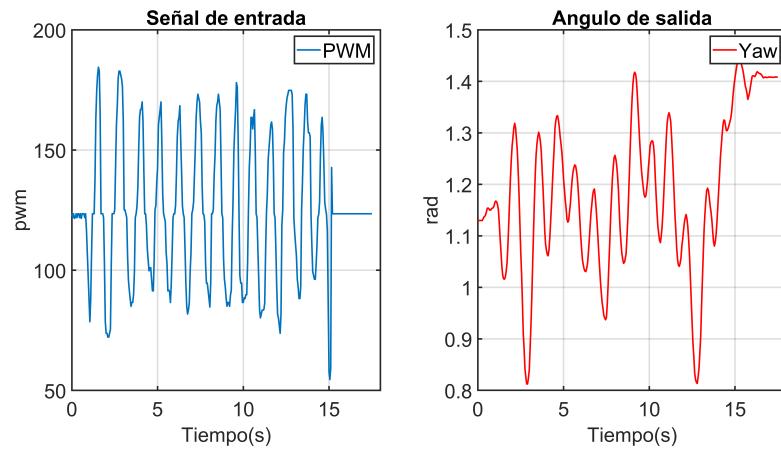
$$y(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \\ \theta \end{bmatrix} \quad (3.35)$$

Con el fin de obtener los valores estimados de los parámetros  $a$ ,  $b$  y  $K$  presentes en el modelo matemático se empleó el toolbox de Simulink-Matlab Parameter Estimation. Para ello, se hicieron tres pruebas manuales, en donde la velocidad lineal del vehículo era constante y la de giro se variaba para recopilar la suficiente información que represente la dinámica del robot. Cada prueba realizada se hizo a una velocidad lineal aproximada de  $2 \text{ m/s}$ , dicha velocidad se termina sobre la marcha dadas las restricciones de espacio y enfoque hacia el que va dirigido el vehículo. El suelo sobre el cual se realiza cada una de las pruebas es de tipo arenoso, debido a que dentro del campus universitario es el único que se asemeja al tipo de terreno presente en varios cultivo de la región.

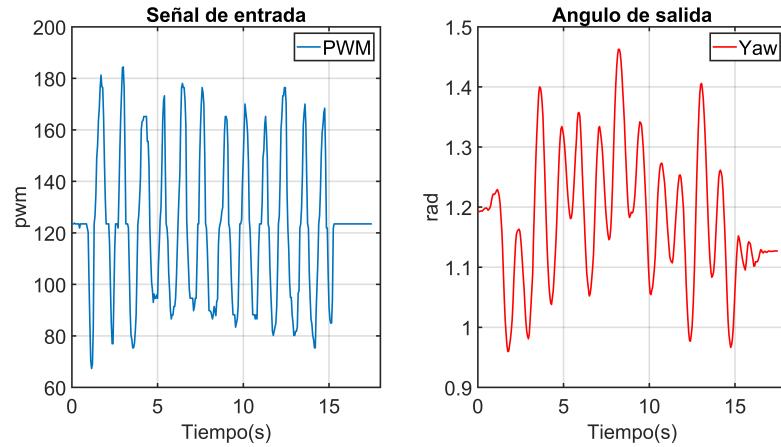


**Figura 3.5:** PWM-Ciclo util. Fuente: Autor

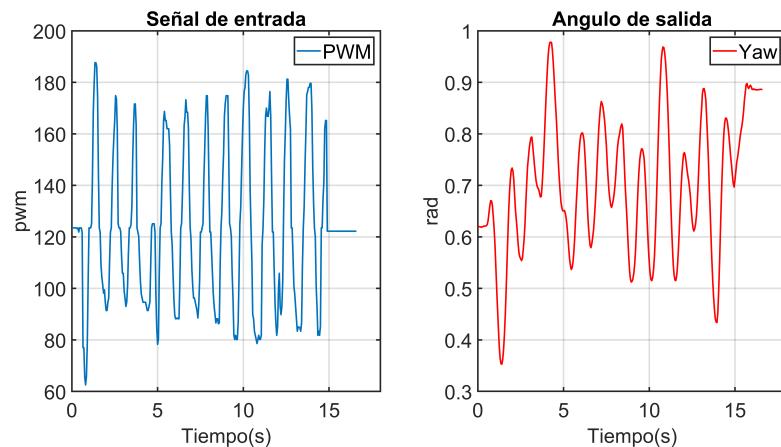
A través de la tarjeta de desarrollo Raspberry Pi, se genera una señal PWM de 8 bits, dicha señal es adquirida por el drive RoboteQ como ciclo útil, el cual interpreta como un ciclo útil de 50 % como una no acción o el no aplicar voltaje a los motores, como se puede observar en la figura 3.5 existe una zona muerta en la que el vehículo aun no tendrá movimiento, esto se debe a que la señal aplicada entre esos rangos no es suficiente para romper la inercia de los motores del vehículo.



**Figura 3.6:** Prueba para la estimación de parámetros 1. Fuente: Autor

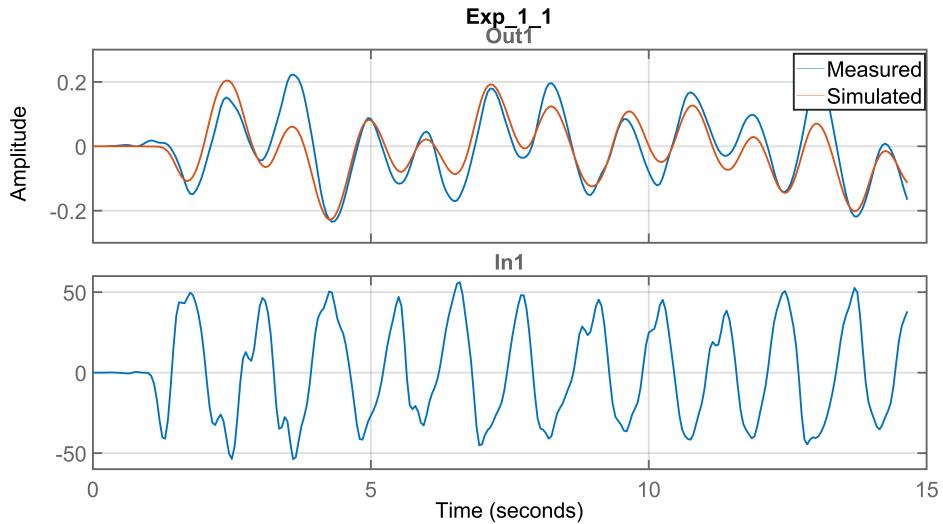


**Figura 3.7:** Prueba para la estimación de parámetros 2. Fuente: Autor



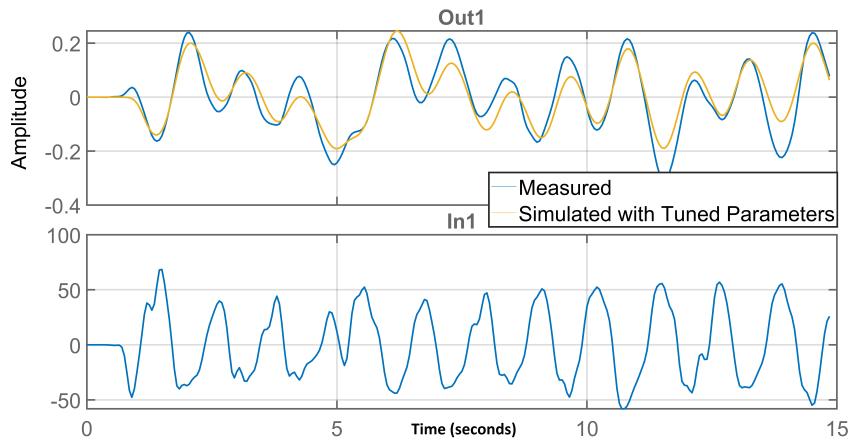
**Figura 3.8:** Prueba para la estimación de parámetros 3. Fuente: Autor

En las figuras 3.6, 3.7 y 3.8 se observa la señal de entrada aplicada y el angulo de salida del vehiculo, dicho angulo proviene de la IMU, que según el fabricante este ya se encuentra pre-procesado por distintos filtros internos que provee XSENS. Para usar las pruebas mencionadas es necesario procesar dichos datos, removiendo la media y las tendencias presentes debido a problemas de construcción del vehículo, por ello es necesario emplear filtros pasa altos y pasa bajos en los datos, buscando remover la tendencia del vehículo con el pasa altos y el ruido de medición con el pasa bajos.

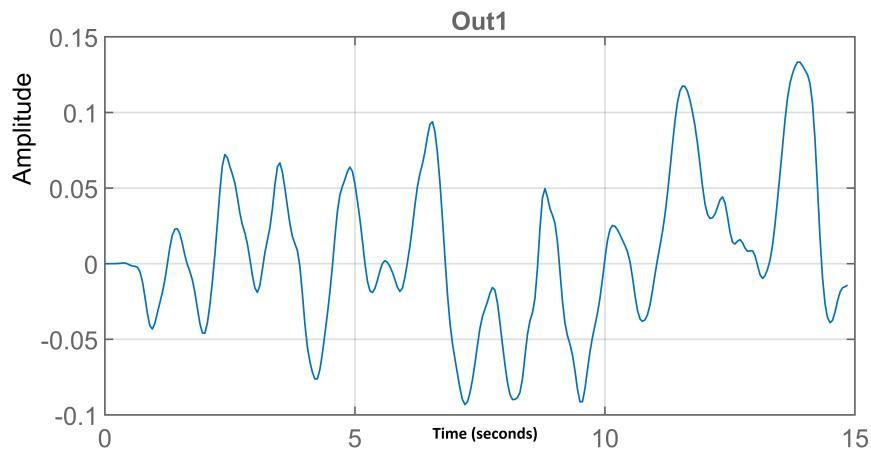


**Figura 3.9:** Ajuste del modelo con datos de entrenamiento para calcular  $a$ ,  $b$  y  $K$ . Fuente: Autor

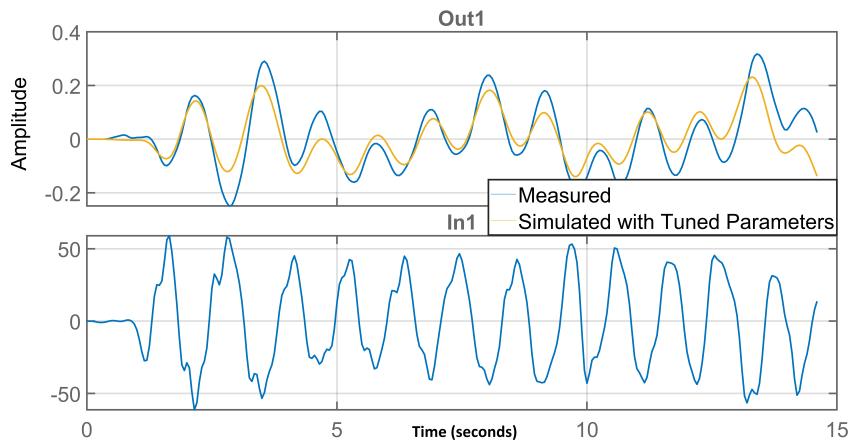
En la figura 3.9 se observa los datos de entrenamiento para la estimación de parámetros, como se mencionaba anteriormente, los datos tuvieron que ser procesados para que la estimación del sistema de los parámetros reflejara la dinámica del vehículo lo mas acercada posible. En las figuras 3.11 y 3.13 se muestran las pruebas de validación, en las cuales la estimación de los parámetros se ajusta adecuadamente al modelo, pues en el error que describen en los indices de error 3.10 y 3.12 es de aproximadamente  $0.15 \text{ rad}$  o  $8.6$  grados.



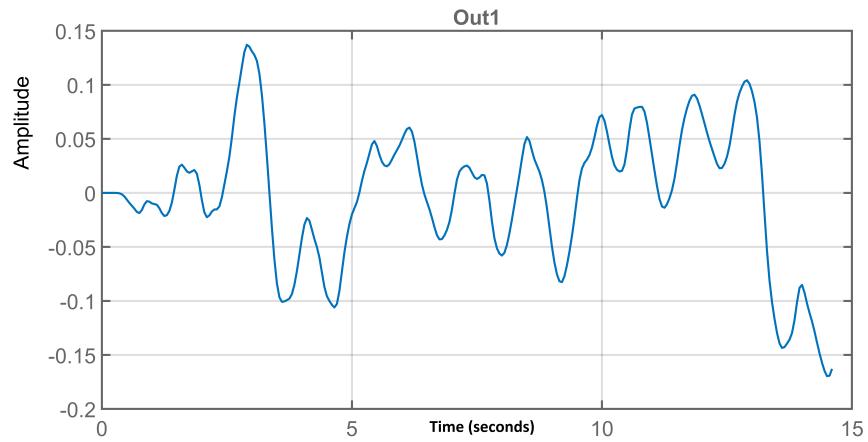
**Figura 3.10:** Validación del modelo con los parámetros estimados prueba #1. Fuente: Autor



**Figura 3.11:** Indice de error de la estimación prueba #1 Fuente: Autor



**Figura 3.12:** Validación del modelo con los parámetros estimados prueba #2. Fuente: Autor



**Figura 3.13:** Indice de error de la estimación prueba #2 Fuente: Autor

Se calcula la raíz del error cuadrático medio o por sus siglas en inglés RMSE de cada una de las pruebas realizadas con el fin de observar la diferencia entre los datos medidos y los estimados por el modelo, a partir de la siguiente expresión.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

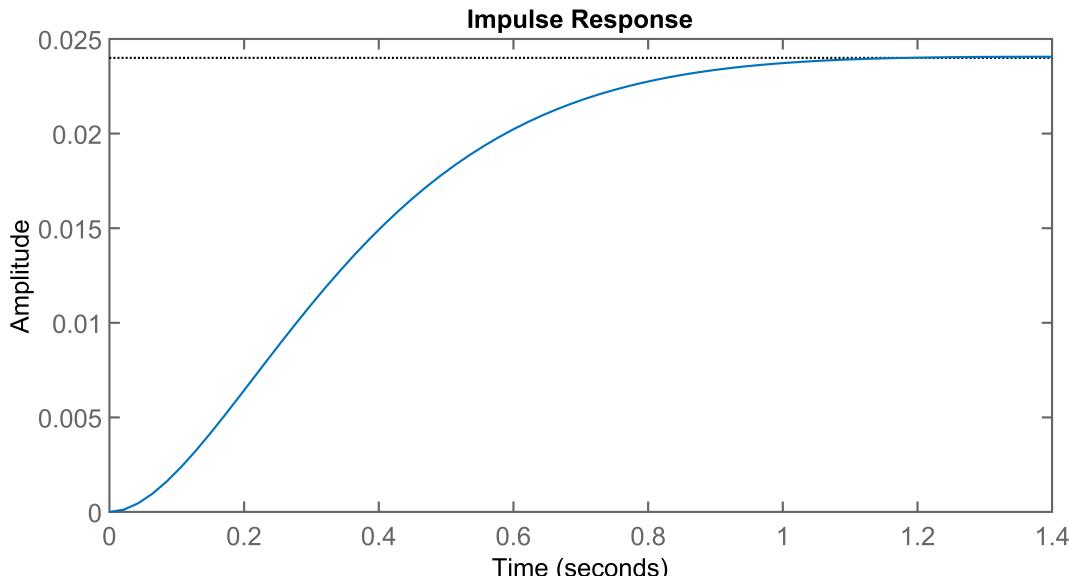
Los valores del RMSE obtenidos en cada una de las pruebas realizadas se pueden observar en la tabla 3.1, de la cual el valor máximo es de 0,0563 rad o 3,2258 grados. Con ello se concluye que hubo un correcto ajuste de los parámetros en el modelo del sistema.

**Tabla 3.1:** RMSE de las pruebas realizadas. Fuente: Autor

Prueba	RMSE
Estimación	0.0527 rad
Validación #1	0.0470 rad
Validación #2	0.0563 rad

### 3.2.3. Control de trayectoria

Para el diseño de un controlador hay que tener en cuenta varios aspectos, como el modelo matemático o la planta sobre la cual se va hacer control, la cual ya está definida en el capítulo previo, otro factor es el tiempo de establecimiento del modelo. Para el presente proyecto se determinó a partir de una entrada impulso sobre el modelo, obteniendo un tiempo de establecimiento aproximado de 1.5 segundos como se observa en la figura 3.14.



**Figura 3.14:** Respuesta impulso del modelo. Fuente: Autor

Definido el tiempo de establecimiento y asumiendo una respuesta de segundo orden del sistema, se continuó con el cálculo del tiempo de muestreo del controlador a partir de su respuesta en el tiempo, tomando como referencia el libro Sistemas de Control Digital [48]. Para el tiempo de muestreo  $T$  se tiene en cuenta la siguiente restricción:

$$\frac{T_d}{30} \leq T \leq \frac{T_d}{10}$$

Donde  $T_d$  se define como:

$$T_d = \frac{2\pi}{w_d} \quad (3.36)$$

Inicialmente se calcula la frecuencia natural amortiguada  $w_n$  y no amortiguada  $w_d$  a partir de las siguientes expresiones:

$$w_n = \frac{4,6}{\sqrt{1 - \zeta^2}}$$

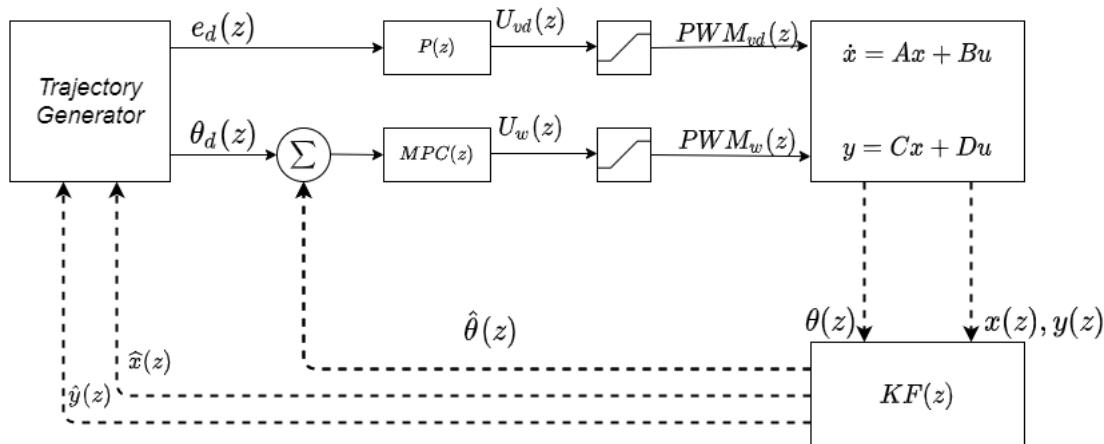
$$w_d = w_n \sqrt{1 - \zeta^2}$$

Considerando  $\zeta$  de 0.8 ya se tienen los datos necesarios para el cálculo del tiempo de muestreo, de ahí que:

$$T_d = 2,732s$$

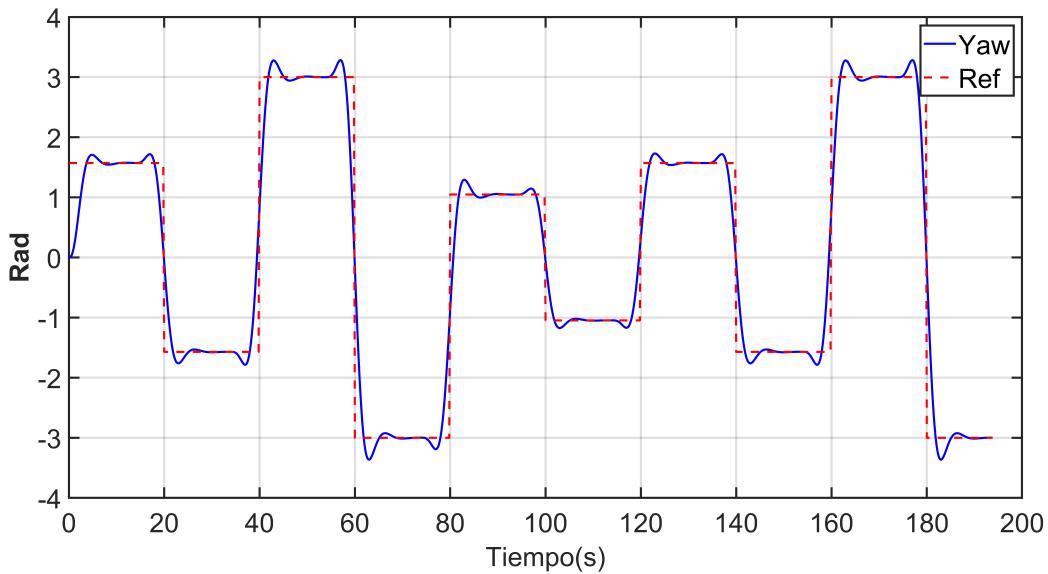
$$0,2732 < T < 0,0911$$

Por lo tanto se establece un tiempo de muestreo de 0.2s. Esto abre paso al sistema de control de seguimiento de trayectorias, que se basa en un control predictivo con base en el modelo para la orientación del vehículo, la cual proviene del generador de trayectoria establecida previamente en código, seguido por un control proporcional al error de distancia para la velocidad lineal del robot.



**Figura 3.15:** Lazos de control. Fuente: Autor

Previamente a la puesta en marcha del controlador sobre el vehículo, inicialmente se validó en simulación, proponiendo un horizonte de predicción de 30 y un horizonte de control de 25 a partir del tiempo de establecimiento del sistema 1.5s dividido en el tiempo de muestreo del filtro de Kalman. También para efectos de una mejor respuesta del sistema se consideraron los pesos para la matriz  $R_w = 2,5$  el cual hace un énfasis en el seguimiento de la trayectoria y  $Q_w = 0,0006$  afectando directamente la acción de control, con dichos valores se obtuvieron resultados favorables como se puede apreciar en la figura 3.16 y posteriormente en el capítulo 4.



**Figura 3.16:** Validación en simulación del Controlador. Fuente: Autor

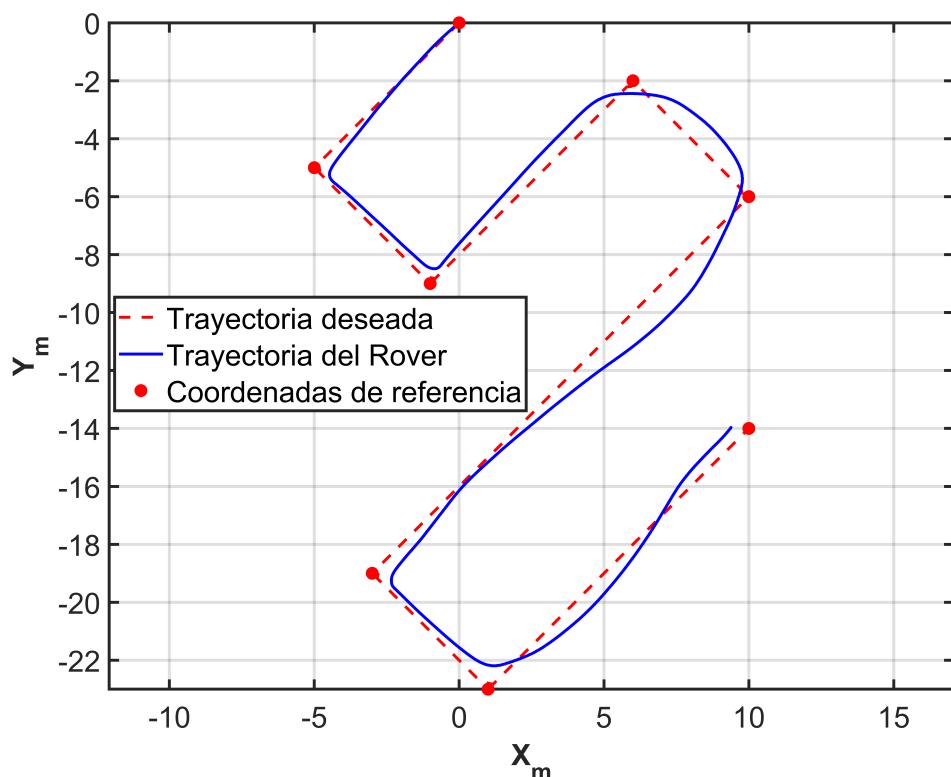
En lo que se refiere al lazo de control para la velocidad lineal, se empleo un control proporcional al error de distancia a los puntos intermedios generados durante la trayectoria, de tal modo que la velocidad máxima sea cuando el error es mayor o igual a dos metros. El valor de dicha ganancia se va ajustando en campo, pues dependiendo de las condiciones del terreno su valor tiende a cambiar, ya que con exceso de humedad el vehículo tiende a deslizarse sobre el mismo y la trayectoria puede crecer de manera desproporcionada. Se elige un control proporcional al error debido a que no es necesario un desarrollo avanzado en este aspecto ya que el driver de los motores cuenta con un controlador que ayuda al vehículo a girar, disminuyendo la velocidad lineal del mismo.

# Capítulo 4

## Resultados

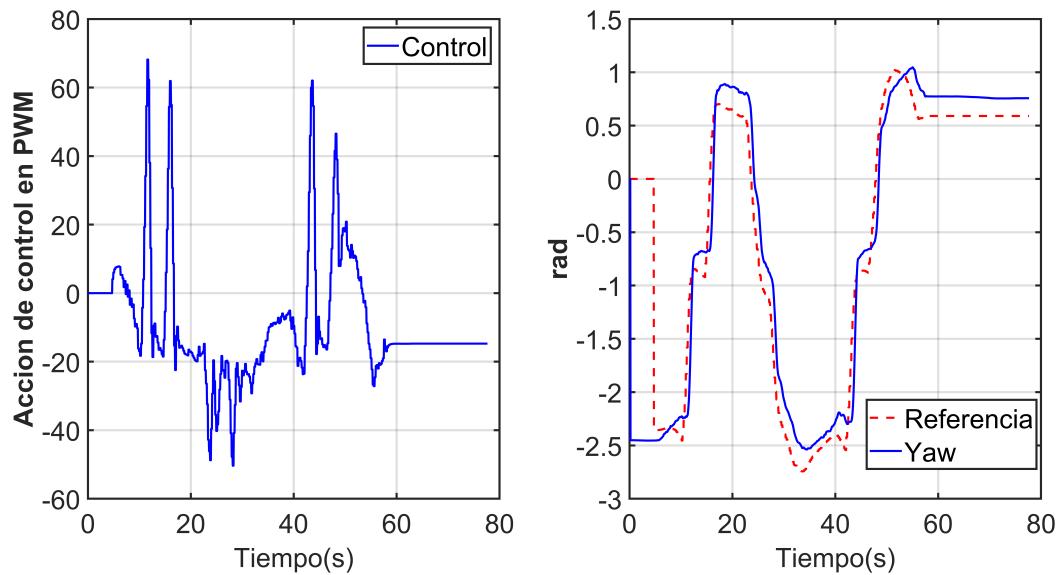
### 4.1. Filtro de Kalman y Control de Trayectoria

En el actual capítulo, se presentan los resultados de una de las pruebas de control de trayectoria. En las siguientes imágenes se puede observar el desempeño del controlador para de mantener el vehículo sobre la trayectoria y del filtro de Kalman al estimar la misma. En la figura 4.1 se observa la trayectoria del vehículo estimada por el filtro de Kalman y la trayectoria deseada.

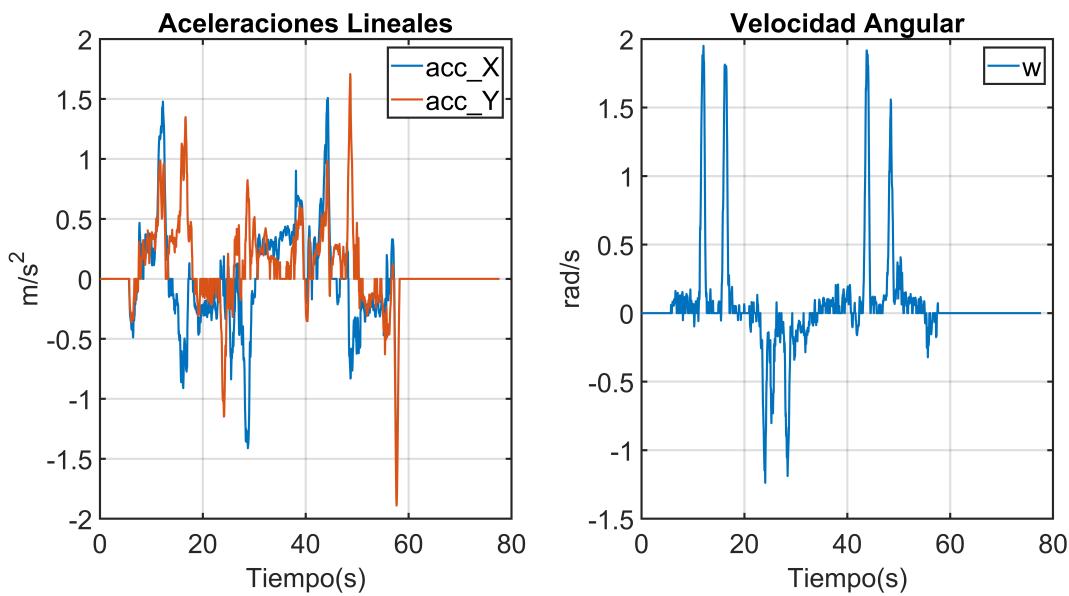


**Figura 4.1:** Trayectoria recorrida por el vehículo Fuente: Autor

A simple vista en la figura 4.1 la estimación de posición proveniente del filtro de Kalman funcionó y el seguimiento de la trayectoria se desempeñó correctamente, pero existen distintos métodos para determinar el error de lo estimado sobre lo deseado, para ello se utilizó la raíz del error cuadrático medio, del cual se obtiene un error de 1.038 metros de la ruta que siguió el vehículo sobre la trayectoria deseada. Hecha esta salvedad a continuación se presentan en las figuras 4.2-4.3 la acción de control, el seguimiento de la referencia por parte del vehículo, las aceleraciones lineales y la velocidad angular del mismo durante la prueba realizada.



**Figura 4.2:** Señal de control del sistema y seguimiento de la dirección del vehículo. Fuente: Autor



**Figura 4.3:** Aceleraciones lineales y velocidad angular del vehículo. Fuente: Autor

La trayectoria que se observa en la figura 4.1 se convirtió a un sistema de coordenadas UTM, las cuales fueron ubicadas sobre un mapa de Google con ayuda del software QGIS. En el se observa que hubo una correcta estimación de la posición en el marco de referencia global por parte del filtro de Kalman como se muestra en la figura 4.4.



**Figura 4.4:** Trayectoria recorrida en UTM. Fuente: Autor

# **Capítulo 5**

## **Conclusiones y Recomendaciones**

### **5.1. Conclusiones**

El desarrollo del sistema en la plataforma Robotic Operating System ROS, permitió integrar exitosamente los elementos de percepción, acción y control que componen el software del robot de forma robusta y con la posibilidad de añadir más componentes al sistema como sensores, actuadores y algoritmos multitarea. El instructivo de como iniciar el sistema y código fuente puede encontrarse disponible online en el siguiente repositorio: <https://github.com/JuanCarlos-TiqueRangel/Rover>

El método de identificación de modelos matemáticos empleando caja gris es una opción sencilla que permite representar la dinámica real del vehículo, pero es necesario procesar los datos como se comentó en la sección 3.2.2, removiendo la media y aplicando un filtro pasa banda pues removiendo las bajas frecuencias se mitigan errores de fabricación que generen tendencias en el vehículo, de igual manera sucede para las altas frecuencias ya que se remueve el ruido de medición proveniente del sensor y las perturbaciones del terreno.

La estimación de posición desarrollada empleando el filtro de Kalman fue una de las tareas mas complejas de realizar, pero se logró efectuar satisfactoriamente como se pudo observar en los resultados. De igual forma cabe comentar que su correcto funcionamiento depende en gran medida de las correctas mediciones entregadas por los sensores, pues dependiendo de los pesos de las matrices de covarianza presentes en la sección 3.1, el algoritmo tiende a estimar los mismos valores medidos o provenientes del sensor. Por tal motivo, a la entrada del filtro de Kalman se agregó la velocidad del GPS junto con la odometría y no la coordenada que el mismo entrega, esto se debe a que la posición del GPS tiene variaciones aproximadas de 3 metros, lo cual perjudica en gran medida la estimación de posición. Por tanto emplear la velocidad que proviene del GPS soluciona la varianza del sensor ademas que también mitiga el problema de deslizamiento que puede presentarse en la estimación de posición únicamente por odometría.

El control predictivo con base en el modelo, presentó resultados gratificantes aun sin implementar restricciones de ningún tipo, por tal motivo se puede decir que es una técnica de

control de acción rápida y robusta, dado que responde de manera optima ante los errores y consideraciones presentes en el modelamiento del sistema como las perturbaciones que el mismo puede tener.

## 5.2. Recomendaciones

Las coordenadas suministradas por el GPS presentan errores muy elevados, tiene una taza de refresco lenta que fluctúa los 0.5s ademas de tener una inestabilidad notable en los datos que entrega. Por tal motivo es recomendable emplear un sistema de geolocalización con una mayor tasa de refresco y que brinde una posición mas precisa como es el caso de un sistema RTK (Real Time Kinematic).

El control predictivo presentó resultados gratificantes, pero se aconseja emplear mas herramientas del mismo, tal es el caso de las restricciones como solución al problema de optimización, con el fin de obtener un sistema de control con mayor robustez y velocidad al momento de ejecutar la acción de control sobre el vehículo y con ello reducir aceleraciones fuertes que se observan en la figura 4.3.

Cada avance del proyecto implica una mayor adquisición de datos y procesamiento de algoritmos más complejos que tienen un elevado consumo computacional, como lo es el caso del filtro de Kalman, por tanto se aconseja hacer un cambio del ordenador a bordo del rover a uno con mayor capacidad de computo, como lo seria una Jetson Nano.

El robot realiza el seguimiento de trayectorias de forma autónoma y cuenta con todos los elementos necesarios para ello, pero ante posibles obstáculos o eventos que surgen en el trayecto no tiene la posibilidad de actuar sobre los mismos, razón por la cual es aconsejable incorporar otros sensores que permitan detectar las condiciones del medio por las que se moviliza el vehículo, por ejemplo sistemas de visión.

# Bibliografía

- [1] Instituto Geográfico Agustín Codazzi. Tolima, uno de los departamentos con mayor potencial agricola en colombia. Ultimo acceso: Abril 2020. [Online]. Available: <https://igac.gov.co/>
- [2] J. Chica L., Y. C. Tirado O., and J. M. Barreto O., “Indicadores de competitividad del cultivo del arroz en colombia y estados unidos,” 2016.
- [3] E. Gil, “Situación actual y posibilidades de la agricultura de precision,” 2010.
- [4] Y. M. Nova, “Desarrollo de un sistema de navegación autónoma para un vehículo eléctrico tipo rover, para aplicaciones de agricultura de precisión,” Tesis de Ingeniería Electrónica, Universidad de Ibagué, 2019.
- [5] L. Ylva, “A comparison between mpc and pid controllers for education and steam reformers,” Master’s thesis, Charlmers University of Technology, 2014.
- [6] V. P. G and R. J.A, “Comparison between an auto-tuned pi controller, a predictive controller and a predictive functional controller in elementary dynamic systems,” 2012.
- [7] J. Cai, H. Jiang, L. Chen, J. Liu, Y. Cai, and J. Wang, “Implementation and Development of a Trajectory Tracking Control System for Intelligent Vehicle,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 94, no. 1, pp. 251–264, 2018.
- [8] T. Luettel, M. Himmelsbach, and H. J. Wuensche, “Autonomous ground vehicles-concepts and a path to the future,” pp. 1831–1839, 2012.
- [9] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, “Perception, planning, control, and coordination for autonomous vehicles,” pp. 1–54, 2017.
- [10] L. E. Solaque Guzmán, M. A. Molina Villa, and E. L. Rodríguez Vásquez, “Seguimiento de trayectorias con un robot móvil de configuración diferencial,” 2014.
- [11] V. Valencia, A. Jhonny, O. Montoya, and L. Hernando, “Modelo cinemático de un robot móvil tipo diferencial y navegación a partir de la estimación odómetrica,” 2009.
- [12] W. Yu, O. Y. Chuy, E. G. Collins, and P. Hollis, “Analysis and experimental verification for dynamic modeling of a skid-steered wheeled vehicle,” 2010.

- [13] O. Barrero, S. Tilaguy, and Y. M. Nova, “Outdoors trajectory tracking control for a four wheel skid-steering vehicle,” 2018.
- [14] C. Clark, “Lecture notes in autonomous robot navigation,” Princeton University, 2011.
- [15] M. A. Javed, “A State Estimation Approach for a Skid-Steered Off-Road Mobile Robot,” Master’s thesis, University of Waterloo, 2013.
- [16] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
- [17] G. Welch, G. Bishop *et al.*, “An introduction to the kalman filter,” 1995.
- [18] M. S. Grewal and A. P. Andrews, *Kalman filtering*. Hoboken, New Jersey: John Wiley & Sons, Inc, 2015, pp. 619–620.
- [19] F. van der Heijden, R. Duin, D. de Ridder, and D. Tax, *Classification, Parameter Estimation and State Estimation*. The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons, Inc, 2005.
- [20] J. Zambrano and A. González, “Implementación de un algoritmo de control predictivo en espacio de estados sobre una plataforma de simulación desarrollada en Matlab,” *Ingenius*, no. 9, pp. 5–14, 2013.
- [21] L. Wang, *Model Predictive Control System Desing and Implementation Using MATLAB*. Melbourne, VIC 3000: Springer, 2009, ch. 1, pp. 7–13.
- [22] E. S. Dale, F. E. Thomas, A. M. Duncan, and J. D. I. Francis, *Process Dynamics and Control*. John Wiley & Sons, Inc, 2011, pp. 386–388.
- [23] J. Maciejowsky, *Predictive Control with Constrains*. Harlow: Prentice Hall, 2001.
- [24] NovatechRobo. Ros. Ultimo acceso: Septiembre 2020. [Online]. Available: [www.robotic-e-learning.com](http://www.robotic-e-learning.com)
- [25] ROS org. About ROS. Ultimo acceso: Abril 2020. [Online]. Available: <https://www.ros.org/about-ros/>
- [26] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. Birmingham, UK: Packt Publishing, 2013.
- [27] H. Yoshida, H. Fujimoto, D. Kawano, Y. Goto, M. Tsuchimoto, and K. Sato, “Range extension autonomous driving for electric vehicles based on optimal velocity trajectory and driving braking force distribution considering road gradient information,” pp. 4754–4759, 2015.
- [28] Microsoft Azure. Middleware. Ultimo acceso: Abril 2020. [Online]. Available: <https://azure.microsoft.com/es-es/overview/what-is-middleware/>

- [29] TESLA. Model s. Ultimo acceso: Octubre 2020. [Online]. Available: <https://www.tesla.com/models>
- [30] S. Valentin, “Autonomous tractor technology shows way forward for farming: enhancing efficiency and working conditions in agriculture,” Case IH and CNH Industrial, Tech. Rep., 2016.
- [31] JOHN DEERE. Autotrac. Ultimo acceso: Octubre 2020. [Online]. Available: <https://www.deere.com/en/technology-products/precision-ag-technology/guidance/auto-trac/>
- [32] NEW HOLLAND, “New holland nhdrive concept autonomous tractor makes appearances at major australian field days,” NEW HOLLAND, Tech. Rep., 2017.
- [33] KUBOTA Corporation. Kubota concept tractor. Ultimo acceso: Octubre 2020. [Online]. Available: <https://www.kubota.com/innovation/concept-tractor/index.html>
- [34] YANMAR. Robot tractor. Ultimo acceso: Octubre 2020. [Online]. Available: <https://www.yanmar.com/global/about/technology/vision2/robotics.html>
- [35] Dot Technology Corp. About dot. Ultimo acceso: Octubre 2020. [Online]. Available: <http://seedotrun.com/faq.php>
- [36] F. B. Argomedo, N. Marchand, and O. Sename, “Constrained Model Predictive Control of a skid-steering mobile robot,” pp. 4653–4658, 2009.
- [37] T. Kim, “Path Tracking for a Skid-steer Vehicle using Learning-based Model Predictive Control,” Master’s thesis, Seoul National University, 2017.
- [38] Open Source Robotics Foundation. Ros catkin. Ultimo acceso: Mayo 2020. [Online]. Available: [https://wiki.ros.org/catkin/conceptual\\_overview](https://wiki.ros.org/catkin/conceptual_overview)
- [39] N. Garcia and C. Molina, “Desarrollo de un robot móvil terrestre semi-autónomo con acceso remoto,” Tesis de Ingeniería Electrónica, Universidad de Ibagué, 2019.
- [40] SPEKTRUM. Ar8000 8-channel dsmx receiver. Ultimo acceso: Agosto 2020. [Online]. Available: <https://www.spektrumrc.com/Products/Default.aspx?ProdID=SPMAR8000>
- [41] RoboteQ. Hdc2460. Ultimo acceso: Agosto 2020. [Online]. Available: <https://www.roboteq.com/products/products-brushed-dc-motor-controllers/hdc2450-259-detail>
- [42] ARDUPILOT. Ublox gps+compass module. Ultimo acceso: Agosto 2020. [Online]. Available: <https://ardupilot.org/copter/docs/common-installing-3dr-ublox-gps-compass-module.html>
- [43] G. Baddeley. Gps - nmea sentence information. Ultimo acceso: Agosto 2020. [Online]. Available: <http://aprs.gids.nl/nmea/>
- [44] A. Coticchia and L. Surace, “Risoluzione di problemi geodetici con le minicalcolatrici elettroniche programmabili,” no. 1, pp. 111–113, 1978.

- [45] Tique Rangel, Juan Carlos. Rover. Ultimo acceso: Noviembre 2020. [Online]. Available: <https://github.com/JuanCarlos-TiqueRangel/Rover>
- [46] Autonics. Serie e50s. Ultimo acceso: Agosto 2020. [Online]. Available: <https://www.autonics.com/series/3000485>
- [47] K. Timo, “Time series analysis kalman filtering,” KTH Royal Institute of Technology, 2013.
- [48] O. Barrero Mendoza, *sistemas de control digital*. Ibagué: Universidad de Ibagué, 2018, ch. 2, pp. 37–40.