MAPAS DE ENTORNOS MEDIANTE NAVEGACIÓN DIFUSA Y SISTEMA DE TELEOPERACIÓN DE UNA PLATAFORMA PIONEER P3-DX

Daniel Granda Gutiérrez [#1] Danny Vásconez Chimbo [#2]

[#] *Departamento de Ingeneria Electrónica, ESCUELA POLITÉCNICA DEL EJÉRCITO Sangolquí-Ecuador*

[1] degranda87@gmail.com

[2] dannyvasconeze@gmail.com

***Resumen*—El presente proyecto describe el diseño e implementación de aplicaciones de Teleoperación, Adquisición de Datos, Control Difuso de Velocidad y Mapeo de Entornos en 2D, para la plataforma móvil Pioneer P3-DX mediante el uso de sonares, odometría y software libre GNU/Linux.**

El proyecto brinda una guía para utilizar los conceptos de programación en Python, que permite crear aplicaciones de manera versátil mediante el uso de librerías como: GTK para el desarrollo del entorno gráfico, PYFUZZY para el desarrollo del controlador difuso de velocidad y OPENCV para mostrar los mapas del entorno

I. INTRODUCCIÓN

En la actualidad uno de los mayores problemas en las plataformas móviles es la planificación de trayectorias y la navegación.

El mapeo de entornos se puede resolver de distintas maneras, generando mapas de entorno que pueden ser realizados por descripción geométrica, rejilla de ocupación, hitos (landmarks) y otras.

La navegación con visión artificial a veces no es posible ya sea por costos o por caracteristicas del entorno, una alternativa es la utilización de sonares que son una de las opciones más económicas.

En el proyecto se usó técnicas de lógica difusa, para controlar la velocidad de la plataforma y programación para realizar navegación y estimación del mapa del entorno.

A pesar de que la plataforma Pioneer 3-DX dispone del software MOBILE-EYES que permite realizar teleoperación, navegación y generación de un modelo del entorno, es una solución rígida a la hora de utilizarlo por ser un sistema propietario. Por esta razón se realiza aplicaciones con interfaz gráfica para la visualización y operación de la plataforma Pioneer 3-DX con software abierto.

Se utilizó la plataforma Pioneer 3-DX y su simulador, utilizando los sonares y un programa para generar un modelo de entorno realizando localización y mapeo simultáneo o SLAM por sus siglas en inglés, sin tener que recurrir a MOBILE-EYES.

La trayectoria se genera mediante la teleoperación de la plataforma móvil al tiempo que se realiza el mapeo del entorno. La plataforma móvil utiliza un control de velocidad difuso para evitar choques con posibles obstáculos que se encuentren en la trayectoria.

Se realizó una interfaz gráfica para realizar la teleoperación, adquisición de datos de los sonares y para mostrar el modelo de entorno. El sistema operativo para realizar el proyecto es una distribución de GNU/Linux, perteneciente a la familia Debian, por lo tanto las librerías ARIA son utilizadas para ese sistema operativo. El lenguaje de programación usado para la creación de las aplicaciones es Python.

II. MARCO TEÓRICO

II-A. Plataforma Pioneer 3-DX

Las plataformas Pioneer son una familia de robots inteligentes móviles, utilizados para educación e investigación.

Figura 1. Plataforma Robótica Pioneer 3-DX

Se define a una plataforma móvil como un dispositivo formado por componentes físicos y computacionales, divididos en cuatro subsistemas:

- Locomoción
- Percepción
- Razonamiento

- Comunicación

Las plataformas MOBILEROBOTS contienen componentes básicos para el sensamiento y navegación en un entorno real.

El desarrollo de software incluye: ARIA (Interfaz Robótica Avanzada para Aplicaciones) y ARNetworking, ambas publicadas bajo licencia pública GNU/Linux, con librerías en C, Java y Python.

ARIA con ArNetworking es una plataforma que permite la integración de software de control propio, ya que maneja los detalles de bajo nivel de interacción cliente-servidor, incluidas las comunicaciones de red y serial, comandos y la información del servidor de procesamiento de paquetes, tiempo de ciclo, y multihilo, así como el apoyo de una variedad de accesorios y controles, tales como giroscopios, sonares, y otros. Estas librerias vienen con código fuente, lo que permite examinar el software y modificarlo para sensores y aplicaciones propias.

II-B. Odometría

Es el método mediante el cual se determina la posición del vehículo (plataforma móvil) tomando la información de los encoders. La odometría se basa en poder obtener el desplazamiento realizado por la rueda asociada a un motor a partir de la medición de las vueltas realizadas por el mismo.

II-C. PYTHON

Python es un lenguaje de programación que tiene una sintaxis clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, hacen que desarrollar una aplicación en Python sea sencillo.

La sintaxis en Python es cercana al lenguaje natural, por este motivo es una de los mejores lenguajes para empezar a programar.

II-D. CONTROL DIFUSO

La teoría de conjuntos difusos permite representar el ser miembro de un conjunto como una distribución de posibilidades. La lógica difusa usa expresiones que no son ni ciertas ni falsas, es decir la lógica aplicada a conceptos que pueden tomar cualquier valor de veracidad dentro de un conjunto de valores intermedios.

La lógica difusa trabaja con conjuntos difusos, los cuales están definidos por sus funciones de pertenencia, la cual expresa la distribución de verdad de una variable.

Un conjunto difuso se puede definir matemáticamente al asignar a cada posible individuo que existe en el universo de discurso, un valor que representa su grado de pertenencia en el conjunto difuso. El grado de pertenencia indica cuando el elemento es similar o compatible con el concepto representado por el conjunto difuso.

La función de pertenencia se establece de una manera arbitraria, basándonos en la experiencia del usuario y contexto del problema.

II-D1. Controlador Difuso: Un sistema de control es un arreglo de componentes físicos conectados de tal manera que el arreglo se pueda manipular, dirigir o regular a sí mismo o a otro sistema.

El siguiente diagrama de bloques se muestra un controlador difuso, donde se muestra un control difuso en un sistema de control de bucle cerrado.

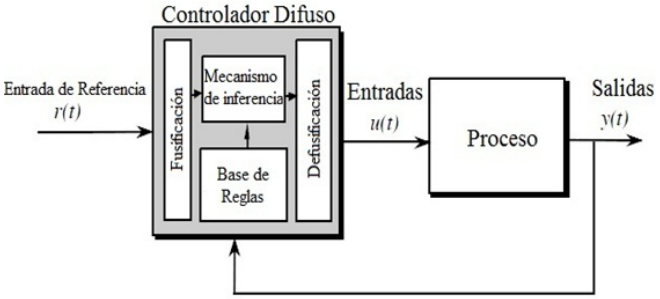

Figura 2. Arquitectura del controlador difuso

El controlador difuso tiene 4 componentes principales:

- Etapa de Fuzificación, que convierte las entradas del controlador en la información que el mecanismo de inferencia puede utilizar para activar y aplicar reglas. Como resultado de la fuzificación se obtiene valores lingüísticos medidos.
- La Base de reglas (conjunto de reglas si-entonces), que contiene una cuantificación lógica difusa de la descripción lingüística del experto de cómo lograr un buen control.
- El mecanismo de inferencia, que emula la decisión del experto en la interpretación y aplicación del conocimiento sobre la mejor manera de controlar la planta.
- Defuzificación, que convierte las conclusiones del mecanismo de inferencia en las entradas reales para el proceso.

II-E. LOCALIZACIÓN Y MAPEO DE ENTORNOS SIMULTÁEOS (SLAM)

La localización y el mapeo simultáneo SLAM (Simultaneous Localization And Mapping), plantea la posibilidad para una plataforma móvil de situarse en un posición desconocida dentro de un sistema desconocido y levantar de forma incremental un mapa del entorno mientras simultáneamente determina su posición utilizando el mapa.

El esquema del problema del SLAM consta de los siguientes pasos:

1. Adquirir la información sensorial.
2. Detectar los puntos de referencia marcados para identificar los puntos de interés del entorno.
3. Establecer correspondencias entre lo observado y lo esperado.
4. Cálculo de la posición.

II-E1. Tipos de Mapas: El tipo de mapa que se va a levantar mediante la resolución de la localización y el mapeo simultáneo es muy importante, tanto como el método que se lleva a cabo para la resolución del mismo.

El SLAM (Localización y Mapeo Simultáneo) es un sistema de navegación basado en la construcción de mapas. Una buena representación debe ser lo suficientemente detallada para que la plataforma móvil pueda localizarse y navegar de forma autónoma.

II-E1a. Mapas Métricos: Sirven para localizar la plataforma robótica con una gran precisión y delinear un camino en presencia de obstáculos, pero requieren mucho espacio de memoria. Se dividen en dos categorías principales:

- Mapas creados a partir de hitos: Describen el entorno como un conjunto de marcas localizadas espacialmente. Las ventajas de esta representación es su compacidad (compacto), lo que hace que sea muy útil para entornos de tres dimensiones y sistemas de dos dimensiones de gran extensión
- Mapas de densidad o de malla: Se usan cuando se necesita una alta resolución para planificar una ruta con exactitud, o el entorno tiene una estructura libre.
  Este tipo de mapas se desarrollan para celdas de dos o tres dimensiones, donde se guarda la probabilidad de estar libres u ocupadas.
  - Una de las principales características es la capacidad de representar entornos no estructurados, y se puede incrementar la exactitud cambiando la resolución de la celda.
  - Las celdas son independientes debido a esto no se puede expresar una relación entre las mismas.
  - Se construyen habitualmente ocupando sensores laser y sonares ya que es necesario obtener la medida de la distancia y la orientación.

Para este proyecto se desarrolló una variación de los mapas de densidad o de malla

III. DESARROLLO DE LA APLICACIÓN DE TELEOPERACIÓN Y ADQUISICIÓN DE DATOS

III-A. Diagrama del Sistema

La (Fig. 3.) muestra el diagrama de conexión entre cliente y servidor, además se observa los programas utilizados por cada uno de estos para el posterior desarrollo de las aplicaciones de Teleoperación, Adquisición de Datos, Control Difuso de Velocidad y Mapeo de Entornos.

El lenguaje de programación utilizado para el desarrollo de las aplicaciones es Python. Para el desarrollo de la interfaz gráfica de usuario de las diferentes aplicaciones se utiliza GTK (The GIMP Toolkit) y para el Mapeo de Entornos OpenCv.

III-B. SERVIDOR

Al implementar las aplicaciones de Control Difuso de Velocidad y Mapeo de entornos, se produjeron errores. Se debe tomar en cuenta que muchos de los errores que se producen

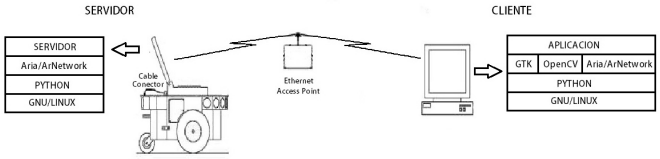

Figura 3. Diagrama General del Sistema

no son producidos por mal programación de las aplicaciones, sino por problemas de que ArNetwork no está correctamente implementado para funcionar con Python, uno de estos ejemplos es el comando ArMutex que a pesar de utilizar los ejemplos proporcionados por el fabricante no funcionaban en Python mostrando un error, es por esta razón que se implementó una librería de comunicación propia y por ende se realizó dos programas de servidores uno implementado utilizando la librería de comunicación de Aria (ArNetWorking) y otro servidor que usa una libreria de comunicación propia.

III-B1. SERVIDOR CON ARNETPACKET: La aplicación servidor permite utilizar la plataforma móvil de manera remota a través de los paquetes ArNetPacket, que son transmitidos y recibidos por el cliente, los paquetes ArNetPacket contienen información transmitida por el cliente o por el servidor. Las librerías y funciones del servidor son mostradas por la (Fig. 4.)

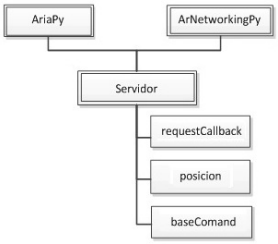

Figura 4. Esquema de funciones del programa Servidor

El servidor a pesar de ser exclusivo de las librerías ArNetworkingPy se necesita utilizar AriaPy para poder conectar la plataforma móvil al servidor. El servidor dispone de dos comandos adicionales del listado base de ArNetPacket que son *requestCallBack* y *posición* estos sirven para probar la conexión con el servidor y adquirir los valores del sonar en coordenadas X,Y,Th respectivamente.

El funcionamiento del servidor se puede describir como dos procesos que funcionan en hilos, estos dos procesos son el núcleo base del servidor y el objeto robot, por lo tanto el programa al final debe mantenerse en espera para poder trabajar con el servidor, esto se puede apreciar en la (Fig. 5.)

Como aprecia en la (Fig. 5.) se realiza la inicialización de las librerías y el objeto robot, despues se inicializa el núcleo base del servidor que activa los parámetros básicos

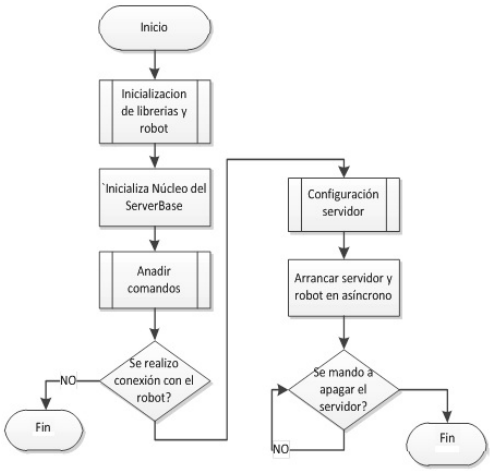

Figura 5. Diagrama de Flujo del programa Servidor

(comandos, parámetros de conexión), después se añaden comandos propios para poder realizar funciones adicionales, se verifica la conexión con el robot, configuración del servidor con las variables del robot, se ejecuta el servidor y el robot en hilos para que funcionen al unísono, al final se mantiene en ejecución la aplicación para poder trabajar con esta durante toda la operación de la plataforma móvil.

La primera subfunción es la inicialización de los elementos básicos y necesarios para poder comenzar a trabajar con la plataforma móvil como se puede apreciar en la (Fig. 6.)

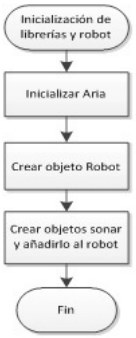

Figura 6. Inicialización de librerias y robot

Lo primero es inicializar Aria esto nos permitirá poder trabajar con el robot y todas sus funciones, luego se debe crear el objeto robot y el objeto dispositivo sonar a este último se le debe añadir el objeto robot para que pueda realizar y adquirir medidas de los sonares de la plataforma móvil. Lo siguiente es crear los comandos y estructuras para los paquetes ArNetPacket.

En la (Fig. 7a.) el diagrama de flujo es para explicar que se debe crear el esqueleto del paquete ArNetPacket y enlazarlo con una función en el programa. En Añadir el paquete para

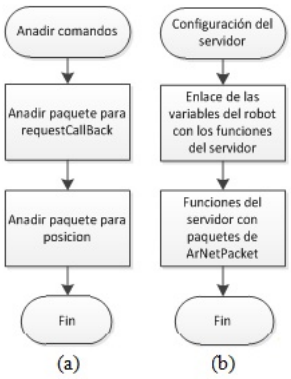

Figura 7. Comandos y paquete ArNetPacket

requestCallback se incluye la información de requestCallBack al paquete, el mismo que sirve para probar la conexión con el robot, de manera similar se hace en Añadir paquete para posición que se utiliza para adquirir los valores del sonar. Este caso es un ejemplo de cómo se realizó las funciones del servidor con ArNetPacket.

En la figura (Fig. 7b.) en cambio se enlazan las variables y valores de sensores del objeto robot a la función que esta enlazada al esqueleto del paquete.

III-B1a. PAQUETE ARNETPACKET: El paquete ArNetPacket sirve para comunicar el cliente con el servidor, dentro de este paquete se encuentran los parámetros y comandos para dar las órdenes a la plataforma móvil como los valores devueltos por esta.

El servidor y cliente funcionan a través de dichos paquetes. Al ser todos estos datos realizados para C al trabajarlos en Python es necesario darles un valor antes de utilizarlos como es el caso del parámetro String, ya que para Python no es necesario declarar el tipo de una variable solo se tiene que darle un valor para que este sepa a que pertenece, por lo tanto al ARIA y ARNETWOKING al ser realizados en C se tiene que acomodar la manera en cómo se utiliza Python con variables para no tener problemas ya que todas las funciones de estas librerías se encuentran escritas en C y se realiza una importación desde C a Python.

III-B2. SERVIDOR SIN ARNETPACKET: Como se puede ver en la (Fig. 8.) lo primero que se realiza es llamar a la clase NoPyArnetworking que es la encargada de tomar los datos de la plataforma robótica como son la odometría, sonares, motores y funciones de movimiento, después de esto se leen los datos adquiridos. Luego de esto se verifica la Dirección IP del servidor y el número de puerto, si los parámetros son los correctos se ejecuta las funciones NoPyArnetworking.robot y NoPyArnetworking.server, que son parte de la clase NoPyArnetworking.

Si la Dirección IP o el número de puerto no son los correctos, se muestra un mensaje y luego se setea la Dirección IP como localhost (de la máquina donde corre el programa) y el número de puerto con 74744 y se ejecuta las funciones NoPyArnetworking.robot y NoPyArnetworking.server.

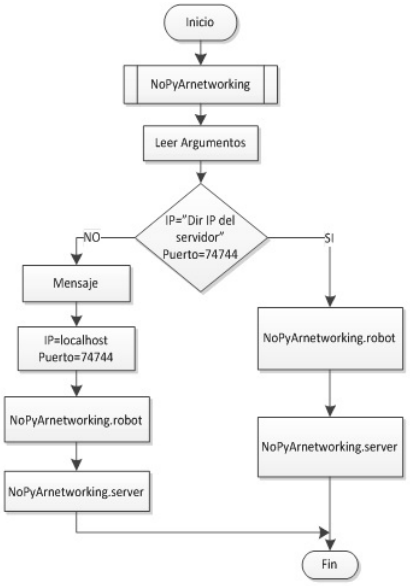

Figura 8. Diagrama de flujo del programa servidor sin ArNetWorking

A continuación se muestra la clase NoPyArnetworking con sus funciones respectivas ver (Fig. 9.).

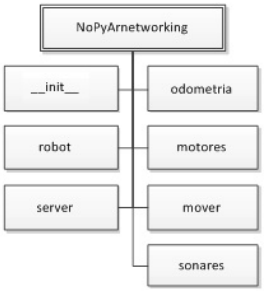

Figura 9. Esquema de funciones de NoPyArnetworking

La función *__init__* inicia las variables que van a contener los valores que envíen las funciones de *sonares* y *odometria*. La función *motores* es la encargada como su nombre mismo lo indica de prender o apagar los motores de la plataforma robótica. La función *mover* sirve para el movimiento y velocidad de la plataforma. La función *sonares* se usa para la triangulación de objetos. La función *odometria* sirve para adquirir los datos de posición de la plataforma en coordenadas (x,y,Th).

III-C. CLIENTE

El cliente fue realizado con funciones para: iniciar la conexión, enviar paquetes, leer paquetes y cerrar conexión, con la finalidad de que pueda ser utilizado por todas las aplicaciones como son: Teleoperación, Adquisición de Datos, Control Difuso de Velocidad y Mapeo de Entornos, sin tener que realizar modificaciones en el código. Por lo tanto las aplicaciones solo deben importarlo y llamar a la función que se va a utilizar, ver (Fig. 10.)

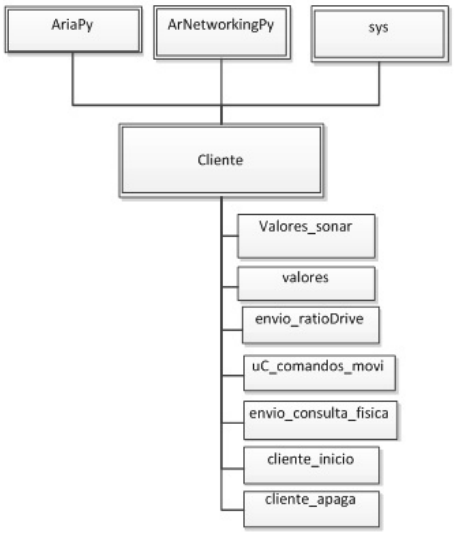

Figura 10. Esquema de funciones del programa Cliente

III-D. APLICACIÓN DE TELEOPERACIÓN

Las aplicaciones están divididas en dos archivos el archivo xml de gtk2 que es el que contiene la información de la interfaz gráfica y el script de Python, al realizar la aplicación de esta manera se facilita el script tanto en la comprensión como en su desarrollo ya que todo se vuelve señales de aviso tomadas de la interfaz gráfica al script.

La aplicación de Teleoperación permite realizar teleoperación básica de la plataforma móvil, tiene una interfaz simple ya que para el caso propuesto es necesario utilizar solo el movimiento.

La aplicación está estructurada en la siguiente forma:

- Clase prueba_teleoperacion
- Ejecución de gtk

La clase *prueba_teleoperacion* tiene la función *__init__* que es la encargada de cargar el archivo xml de gtk, las señales y widgets de la interfaz gráfica y el resto de funciones de la clase son solo para trabajar con la activación de las señales de gtk y realizar una acción según la señal, ver (Fig. 11.)

Por ejemplo para el caso de la función *on_stop_clicked* si el programa detecta la activación de esta función se procede a enviar un paquete ArNetPacket con el comando stop que realiza un parado de emergencia y así ocurre con el resto de funciones.

III-E. APLICACIÓN DE ADQUISICIÓN DE DATOS

La aplicación de adquisición de datos permite realizar lectura de los sonares de la plataforma móvil, es necesario

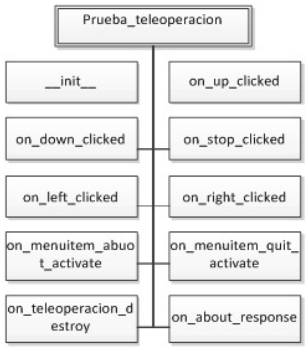

Figura 11. Esquema de funciones de la aplicación de Teleoperación

tener estos valores para que puedan ser utilizados por el controlador difuso y el generador de mapas.

El funcionamiento de esta aplicación es similar al de la Teleoperación excepto por los paquetes de ArNetPacket que envía, ver (Fig. 12.)

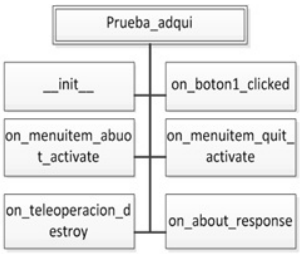

Figura 12. Esquema de funciones de la aplicación de Adquisición de Datos

Por ejemplo la función *on_boton1_clicked*, utiliza la librería cliente_lib. Toma los valores de los sonares, los acomoda en una estructura y los convierte en texto para poder visualizarlos en la interfaz de usuario.

Ambas aplicaciones pueden funcionar al unísono por que el servidor no se bloquea al tener una conexión activa, permitiendo probar al mismo tiempo el movimiento y adquisición de datos para pruebas.

IV. CONTROLADOR DIFUSO DE VELOCIDAD

IV-A. DISEÑO DEL CONTROLADOR DIFUSO

Lo que se pretende lograr es que mientras la plataforma se encuentre a una distancia segura de un obstáculo vaya a una velocidad alta, teniendo en cuenta que la mayor velocidad a la que puede llegar la plataforma es de 1.2 m/s; a medida que la plataforma vaya acercándose al obstáculo esta irá disminuyendo la velocidad hasta que finalmente pare si es que se encuentra muy cerca del obstáculo.

El control difuso para la velocidad de la plataforma Pioneer 3-DX funciona en base al Set Point de entrada que va a ser la distancia con respecto al obstáculo más cercano a la que se quiera que la plataforma frene.

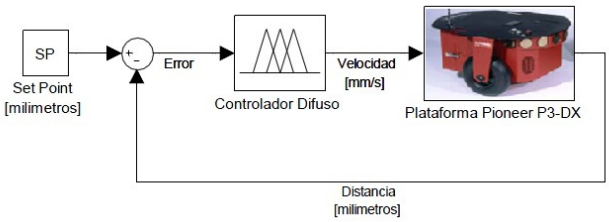

Figura 13. Diagrama del controlador difuso

IV-A1. Rango de operación de las variables de entrada y salida: Para realizar el control de velocidad se ha tomado en cuenta una variable de entrada y una variable de salida. La variable de entrada es: El error que está dado por el rango de funcionamiento de los sonares, los cuales tienen un alcance de 0,12 - 5 m. El error va a estar dado por el Set Point que es la distancia a la cual se quiera que pare el robot con respecto a un obstáculo menos la distancia a la que se encuentre la plataforma de un obstáculo. La variable de salida está dada por la velocidad máxima que puede alcanzar la plataforma: 0 - 1,2 m/s.

Variable de Entrada	**Variable de Salida**
Error	Velocidad

IV-A2. Definición de las funciones de pertenencia: Se definen los adjetivos y rangos para cada una de las funciones de pertenencia.

El Error se define en 5 valores lingüísticos:

Error	**Rango**
Muy Grande Negativo (MNG)	[-5000 3000]
Grande Negativo (GN)	[-4500 -3000 -2000]
Negativo (N)	[-3000 -2000 -1000]
Cero (Z)	[-2000 -1000 0]
Positivo (P)	[-1000 1000 5000]

En (Fig. 14.) se muestra la función de pertenencia de la variable de entrada Error, está dada en [mm].

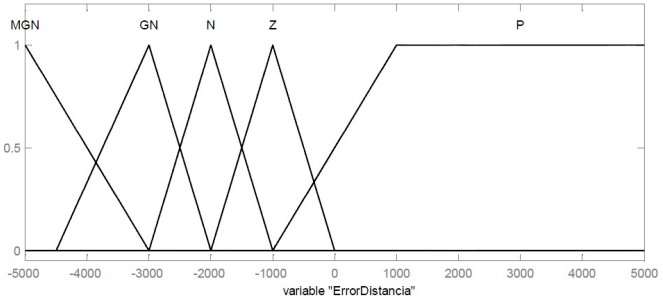

Figura 14. Función de pertenencia de la Variable Error

La variable de salida del controlador es la velocidad, la cual se define en 5 valores lingüísticos:

Velocidad	Rango
Velocidad Muy Baja (VMB)	[-120 255]
Velocidad Baja (VB)	[30 355 600]
Velocidad Media (VM)	[255 -600 950]
Velocidad Alta (VA)	[600 875 1130]
Velocidad Muy Alta (VMA)	[950 1300]

La (Fig. 15.) muestra la función de pertenencia de la variable de salida Velocidad, dada en [mm/s].

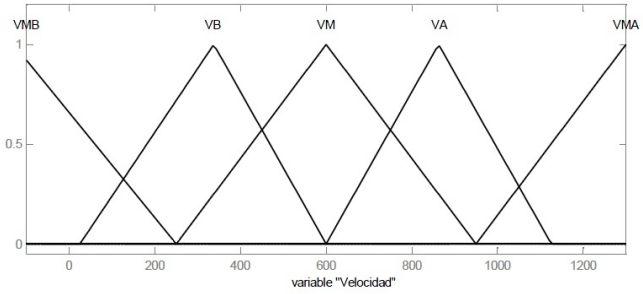

Figura 15. Función de pertenencia de la Variable Velocidad

IV-A3. Desarrollo de la Base de Reglas: En las reglas se tiene una representación implícita del modelo, por lo que de ellas se puede seguir el comportamiento aproximado del modelo. Por lo tanto se definen las reglas que determinan el comportamiento del sistema.

Nro	REGLA
1	Si(Error es MGN) Entonces (VMA)
2	Si(Error es GN) Entonces (VA)
3	Si(Error es N) Entonces (VM)
4	Si(Error es Z) Entonces (VB)
5	Si(Error es P) Entonces (VMB)

En la (Fig. 16.) se muestra una gráfica de Distancia Vs Velocidad para un Set Point de 1.5 metros.

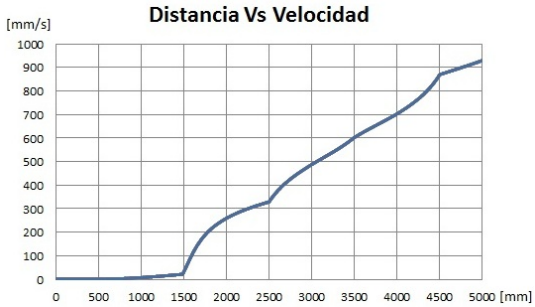

Figura 16. Distancia Vs Velocidad para un Set Point de 1.5 m

V. DESARROLLO DE LA APLICACIÓN PARA MAPEO DE LA PLATAFORMA MÓVIL

V-A. LOCALIZACIÓN

Para la localización se utiliza la odometría, es decir se leen los encoders que se encuentran en las llantas de la plataforma móvil, con esto se reciben datos que entregan el desplazamiento que ha tenido la plataforma con respecto al punto de inicio.

Los sensores miden la posición y orientación del robot. Donde X y Y representan la posición de la plataforma y Th la orientación del robot. Estas coordenadas del sistema odométrico están referenciadas a un sistema absoluto externo, fijo, coincidente con la posición y orientación del robot cuando se enciende.

Desde esas lecturas y conocidas las coordenadas de su posición anterior (x(t - 1), y(t - 1), Th(t - 1)) se calcula la nueva posición. Estos cálculos los hace el sistema operativo del microcontrolador, que realiza la conversión a coordenadas absolutas X, Y, Th.

V-B. MAPEO

Para realizar el mapeo del entorno se utilizan los sonares, los cuales entregan la información de la distancia a la que se encuentran los obstáculos, estas medidas son necesarias para construir los mapas del lugar en el que se encuentra la plataforma móvil. Para renderizar la imagen en dos dimensiones del entorno se ha utilizado OpenCV.

El área que se puede mapear es de 25 x 17.5 metros, es decir desde el punto donde fue encendida la plataforma 12.5 m en el eje Y positivo, 12.5m en el eje Y negativo, 8.75 m en el eje X positivo y 8.5m en el eje X negativo, lo que da un área total de 437.5 m2, área suficiente para mapear una habitación o un pasillo. La (Fig. 17.) ilustra el sistema de coordenadas, inicio de la plataforma y medidas del mapa.

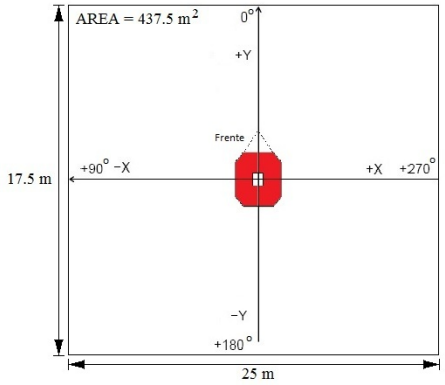

Figura 17. Sistema de coordenadas, inicio de la plataforma y medidas del mapa

V-C. MAPA

El tipo de mapa que este trabajo presenta es una variación del Mapa de Densidad o de Malla que a su vez es una

subclase de un mapa Métrico; ya que captura las características geométricas del entorno, y las representa como un conjunto de objetos cuyas coordenadas están definidas en un plano cartesiano. Este método define una matriz de NxM celdas que representa una cierta región de igual tamaño, asociadas a su localización real según la posición en la matriz. Cada celda puede estar ocupada o vacía, según haya o no un obstáculo;

Para generar el mapa del entorno lo primero que se hace es crear una imagen en blanco del 500 x 250 píxeles en OpenCV, luego se grafican en el visor las coordenadas dadas por la plataforma como círculos de 2 pixeles de radio esto se hace con la finalidad de que se pueda visualizar con mayor facilidad, finalmente se guarda como una imagen.

V-D. APLICACIÓN DE MAPEO DE ENTORNOS

La aplicación para desarrollar la generación del entorno y la localización, necesita de las aplicaciones de Teleoperación y Adquisición de datos.

A continuación se detalla el proceso mediante el cual se logró obtener el mapa del entorno.

La (Fig. 18.) muestra como se comenzó desarrollando la aplicación para esto se utilizó el simulador de la plataforma móvil (4). Se va generando el mapa en un visor (3) a través de la librería OpenCv. Se puede ver que para mover la plataforma se necesita de la aplicación de Teleoperación (2) y se usa también la aplicación de Adquisición de Datos (1) para la lectura de sonares que sirve para tomar los puntos donde se encuentran los obstáculos que después son graficados en el visor. Aquí únicamente se muestra un cuadrante del mapa ya que se asume que el origen del robot es el punto 0,0 es decir que como se mostró anteriormente en la (Fig. 17.) la plataforma puede trabajar con valores positivos o negativos en las coordenadas X y Y tomando en cuenta que el punto donde fue encendida la plataforma es el punto 0,0. El visor en cambio únicamente representa valores positivos de las coordenadas X y Y, es por esto que en esta parte del desarrollo de la aplicación de mapeo de entornos únicamente se grafican los puntos positivos que envía la plataforma, que vienen a formar el primer cuadrante.

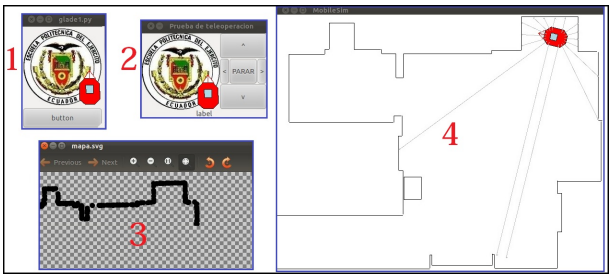

Figura 18. Sistema de coordenadas, inicio de la plataforma y medidas del mapa

La (Fig. 19.) muestra ya un mapa completo generado y la plataforma localizada en el mismo, esto se llevó a cabo con el simulador de la plataforma móvil, se puede observar que ya se genera el mapa completo con los cuatro cuadrantes tomando en cuenta que la plataforma móvil comenzará en el centro de la imagen para hacer esto se mueve el sistema de coordenadas, evitando de esta forma que existan coordenadas negativas y poder así graficar el mapa en todos sus cuadrantes.

Las ecuaciones utilizadas para esto son:

$$\begin{aligned} Xpixel &= \frac{Xreal}{50} + 500 \\ Ypixel &= \frac{Yreal}{50} + 350 \end{aligned}$$

Dónde:
Xpixel = Número de pixeles del mapa creado en el eje X
Ypixel = Número de pixeles del mapa creado en el eje Y
Xreal = Medida en milímetros del entorno real en el eje X
Yreal = Medida en milímetros del entorno real en el eje Y

En estas ecuaciones se puede notar por tanto que la resolución del mapa creado es de 1 pixel = 50 mm. Para graficar el mapa en el centro del visor se hace un desplazamiento de 500 pixeles en X y de 350 pixeles en Y.

Para localizar la plataforma se utiliza los datos tomados de la odometría y se los plasma en el mapa, cada vez que la plataforma se mueve se toma los datos y se actualiza el mapa, lo cual da la idea que el robot se mueve dentro del mapa.

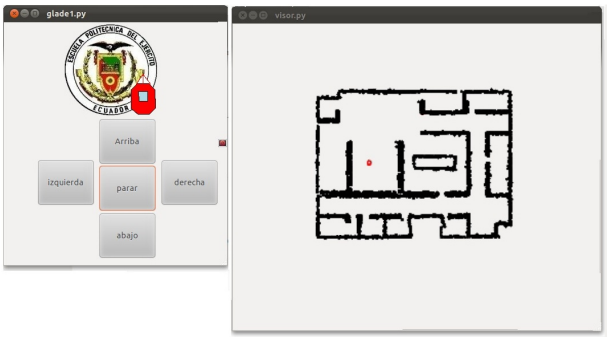

Figura 19. Etapa final para la elaboración del mapa (probado en simulador)

El funcionamiento de la aplicación de Mapeo de Entornos es similar al de la aplicación de Teleoperación, a continuación se muestra un esquema de funciones del programa.

La función *generador_mapa* utiliza la librería *renderizado* y la librería *cliente_lib* para generar el mapa. La librería *renderizado* se la estableció para crear el mapa a través de OpenCv.

VI. APLICACIÓN FINAL TELVEMAP Y APLICACIÓN DE NAVEGACIÓN AUTÓNOMA

VI-A. Libreria de Comunicación Propia

La librería de Comunicación se implementó con el afán de reemplazar la librería de comunicación ArNetworking

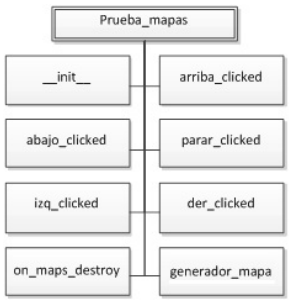

Figura 20. Esquema de funciones para la aplicacion de Mapeo de Entornos

de ARIA que presentó problemas al momento de ejecutar las aplicaciones creadas. Se la realizó para trabajar con el protocolo TCP/IP.

Cabe indicar que toda la comunicación se realiza a través de sockets, por lo que para esto es necesario definir una dirección IP y un puerto de comunicación TCP, tanto para el cliente como para el servidor

A continuación se muestra una tabla de los mensajes que envía el cliente al servidor y los acuses de recibo que envía el servidor al cliente.

Mensajes Cliente	**Respuesta Servidor**
adelante, 5	adelante
atras, 1200	atras
izquierda,456	izquierda
derecha, 763	derecha
sonares	(x1,y1)
(x1,y1)	Listo
	(x2,y2)
(x2,y2)	Listo
⋮	⋮
	(x7,y7)
(x7,y7)	Listo
odometria	(x,y,Th)

VI-B. APLICACIÓN FINAL TELVEMAP

TelVeMap es la aplicación donde convergen las aplicaciones de Teleoperación, Adquisición de Datos, Control Difuso de Velocidad y Mapeo de Entornos.

Debido a que la aplicación TelVeMap, usa un mando (palanca) conectado al cliente para el movimiento de la plataforma, se presenta la configuración del mando ver (Fig. 21.)

El botón Hombre Muerto, se utiliza como un sistema de seguridad de la plataforma robótica.

La Fig. 22. muestra la interfaz de la aplicación TelVeMap.

La (Fig. 23.) muestra el esquema de funciones de la aplicación TelVeMap.

Figura 21. Configuración de Mando para la aplicación TelVeMap

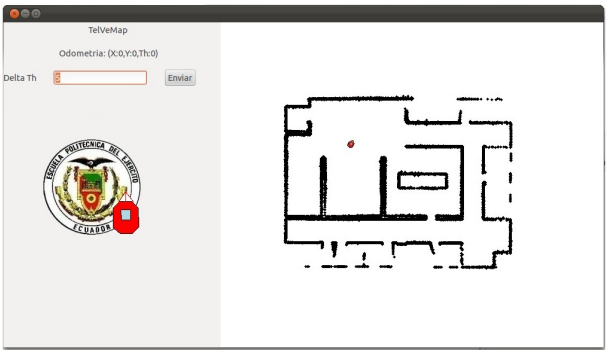

Figura 22. Interfaz de la aplicación TelVeMap

La función *__init__* se encarga de inicializar las variables. La función *button1_onClic* sirve para ajustar la coordenada Th en el gráfico para que coincida con el entorno real al momento de iniciar la aplicación. La función *Etapa1* es un objeto timer que se ejecuta cada 300 m/s y llama a la función *robot_ordenes* que es la encargada de manejar la palanca para el movimiento de la plataforma, esta función *robot_ordenes* a su vez llama a las funciones *fuzzy_velocidad* encargada del control de velocidad difuso y *generador_mapas* que se usa para realizar el mapeo de entornos.

VI-C. APLICACIÓN DE NAVEGACIÓN AUTÓNOMA

La aplicación de Navegación Autónoma se implementa en el equipo servidor. Esta aplicación realiza la navegación del entorno de forma independiente y además incluye el control difuso de velocidad. La Fig. 24. muestra el esquema de funciones del programa.

La función *Actions* es la encargada del movimiento de la plataforma robótica, por lo que esta sirve para realizar los movimientos y giros de la misma. La función *Task* es la encargada de llamar al control difuso de velocidad y a los datos de velocidad que recibe del controlador los escribe en un archivo de texto, esto posteriormente sirve para tener una base de datos de la velocidad a la que fue la plataforma y permite generar también una gráfica del comportamiento de velocidad.

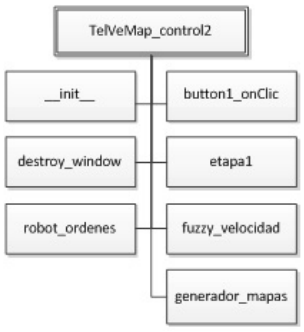

Figura 23. Esquema de Funciones del programa TelVeMap

Figura 24. Esquema de funciones del programa de Navegación Autónoma

VII. Pruebas y resultados

Esta prueba se la llevó a cabo dentro del laboratorio, para simular las paredes de madera se puso láminas de plywood conjuntamente con las paredes de hormigón del laboratorio.

El área tomada por la plataforma móvil fue de aproximadamente 4.35 metros de largo por 2.45 metros de ancho, lo que da un área de 10.66 m2.

En la (Fig. 25.) se aprecia el entorno donde se realizó la prueba, y el mapa de entorno obtenido por la plataforma, los puntos dispersados que no corresponden al entorno se producen por los errores que presentan los sonares en ambientes reales, ya que en pruebas realizadas en simulación no sucede esto.

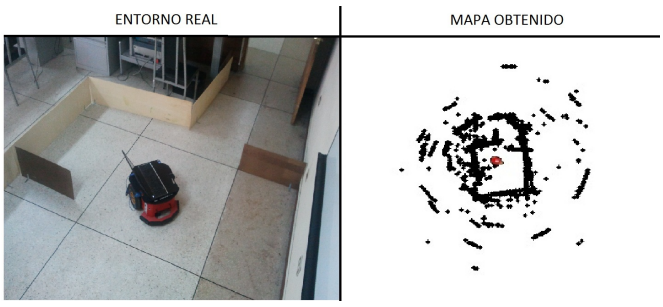

Figura 25. Pruebas en un entorno real

VIII. Conclusiones

- Mediante este proyecto se ha logrado implementar satisfactoriamente las aplicaciones de Teleoperación, Adquisición de Datos, Control de Velocidad y Mapeo de Entornos para la plataforma robótica Pioneer P3-DX mediante el lenguaje de programación Python y el sistema operativo GNU/Linux.
- La ventaja de utilizar sistemas basados en GNU/Linux es que se puede manipular el sistema de tal manera que solo se ejecute lo que se necesita, permitiendo utilizar un computador a bordo con menores prestaciones y teniendo un rendimiento igual a un computador de superiores características que utilice Windows.
- La aplicación de Mapeo de Entornos es eficiente aunque presenta limitaciones al mapeo de paredes debido al error que presentan los sonares en las esquinas, a pesar de todo cumple su función de realizar mapas en 2D, aunque cabe recalcar que no se comporta de la misma forma en la simulación como en el entorno real debido al error que presentan los sensores en ambientes reales.
- Un inconveniente que se observó en el manejo de ARIA con Python, es que ARIA no permite manejar el comando ArMutex que sirve para dormir los hilos y trabajar con distintas aplicaciones y procesos a la vez. Llegan a provocar un desbordamiento de memoria en las diferentes librerías de C que utiliza PyAria y PyNetworking.

IX. Recomendaciones

- Los mapas se pueden mejorar realizando procesamiento de imágenes, para reducir el grosor de las líneas y mejorar la forma, es decir líneas más rectas y planas.
- Debido a que ARIA no permite el manejo del comando ARMUTEX con Python, se puede utilizar lenguaje C para el desarrollo de las aplicaciones.
- Como trabajo futuro se puede utilizar teléfonos inteligentes como dispositivos hápticos ya que estos disponen de acelerómetro, giroscopio y pantalla para realizar la inmersión de una cámara o hacer mapas del entorno realizando una conexión a la plataforma robótica con los protocolos HTTP.

Referencias

[1] MobileRobots.Inc, *Pioneer3 Operations Manual*, 75 págs, 2007.
[2] Morales Jimena, Jaramillo Daniel , *Desarrollo de Aplicaciones y Documentación de las Plataformas Robóticas Pioneer P3-DX y Pioneer3 P3-AT*, 265 págs, 2010.
[3] Passino Kevin, *Fuzzy Control*, 522 págs, 1998.