ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE UNA APLICACIÓN PARA EL TRAZO DE RUTAS PLANIFICADAS DE UN VEHÍCULO A ESCALA REDUCIDA USANDO UN DISPOSITIVO EMBEBIDO Y GPS"

INFORME DE MATERIA INTEGRADORA

Previa a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

RICARDO DAVID MAYA HERRERA

MARLON VINICIO LOAYZA FEIJÓO

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTOS

Quiero primero dar gracias a Cristo Jesús mi Señor y Salvador el cual me ha permitido culminar esta nueva etapa en mi vida y sin el cual nada de esto sería posible.

A mis padres, Carlos Maya y Rocío Herrera, y mi hermana Doménica Maya, por haber estado junto a mí en todo este tiempo apoyándome y dándome lo necesario para cumplir este anhelo que hoy se hace realidad.

A mi enamorada Gabriela Paredes, por su apoyo incondicional durante todo este tiempo que hemos estado juntos, la cual ha sido la ayuda idónea de parte de Dios.

Agradezco también a todos mis profesores que me han impartido sus conocimientos y experiencias, lo cual me ha ayudado a crecer profesionalmente.

Ricardo Maya

Agradezco a Dios por sus tantas bendiciones y por haberme permitido llegar a este momento clave de mi vida. A mis padres, Alfonso Loayza y Kelita Feijóo, y a mis hermanos, Norelis, Alfonso y Kevin, por brindarme su apoyo incondicional, haberme brindado constantes ánimos y por haberme guiado a donde estoy ahora.

A mis amigos, ustedes se han convertido en una segunda familia para mí. Al PhD. Dennis Romero, gracias por su apoyo y guía en el desarrollo de este proyecto.

Marlon Loayza

DEDICATORIA

Este proyecto se lo dedico primero a Dios El cual merece toda la gloria y la honra, que a su vez me ha dado unos padres y una familia incondicional a quienes también dedico con todo mi corazón.

Ricardo Maya

El presente proyecto lo dedico a Dios y a mis padres. A Dios por estar conmigo día a día en todo el transcurso de mi vida, dándome la fortaleza para continuar y mis padres, quienes me han apoyado a lo largo de mi vida en mi educación y velando por mi bienestar.

Marlon Loayza

TRIBUNAL DE EVALUACIÓN

Ph. D. Dennis Romero

PROFESOR EVALUADOR

Ph. D. Angel Sappa

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Ricardo Maya Herrera

Marlon Vinicio Loayza

RESUMEN

El presente documento muestra la implementación de un sistema de control para el seguimiento autónomo de rutas planificadas en un vehículo a escala, en el cuál se usa un dispositivo embebido (raspberry pi 3) y GPS (vector nav).

Se ha decidido dividir el problema en 4 módulos de tal manera que puedan funcionar tanto de forma independiente como integrada, los cuales son: Determinar posición absoluta, cálculo de distancia entre 2 puntos geográficos, cálculo de ángulo entre 2 puntos geográficos con respecto al norte geográfico y el Control de motores del robot, además de contar con el desarrollo de una aplicación en Android para generar la planificación de la ruta a seguir, siendo este el primer paso a realizar para la ejecución del sistema en donde intervienen cada uno de los módulos mencionados y que se detallarán en este trabajo.

Dentro de los resultados nos refleja el error obtenido al momento de llegar a cada uno de los puntos marcados en la ruta, así como el resultado de la integración del sistema en un nuevo robot con la finalidad de probar su portabilidad.

Índice General

CAPÍTULO 1

1. DISPOSITIVOS EMBEBIDOS PARA EL USO DE ROBOTS AUTÓNOMOS UTILIZANDO RUTAS PLANIFICADAS

En los últimos 20 años la tecnología ha evolucionado a tal punto que tareas que en un principio parecían imposibles o demasiado repetitivas y monótonas para un humano, hoy en día son posibles llevarlas a cabo gracias a sistemas automáticos para el manejo y control de tareas, los cuales con el paso de los años fortalecen su presencia en muchas de estas actividades, para hacerlas más eficientes y eficaces.

A pesar de que con la ayuda de estas tecnologías podemos facilitar en gran medida dichas actividades, se ha visto la necesidad de que cuenten con una planificación previa donde se indique, que deben hacer y donde lo deben hacer, siendo esto un gran paso para el desarrollo tecnológico y de investigación.

Para llevar a cabo este proyecto se ha pensado en la problemática que conlleva realizar ciertas actividades, específicamente aquellas en las que es necesaria la intervención de un agente que incorpore un sistema de navegación autónoma dada una ruta planificada. Esto sería productivo en caso de requerir que un drone o un robot siga un camino planificado por el usuario para llevar a cabo su cometido, como puede ser, por ejemplo, el monitoreo y análisis de terreno utilizando imágenes multiespectrales para la fumigación de un área en particular. A parte, este tipo de aplicaciones hacen uso de dispositivos embebidos, los cuales brindan portabilidad y bajo consumo eléctrico, razón por la cual ha incrementado su utilización para el monitoreo de cultivos, vigilancia, entre otros ámbitos.

Este trabajo presenta la construcción de un prototipo que servirá de base para sistemas de propósito específico que tengan como objetivo cubrir rutas específicas en una zona.

Adicionalmente, se hace uso de un kit de navegación de alta precisión el cual nos permite probar el sistema y mostrar el funcionamiento del mismo. Mencionando esto, podremos tomar en consideración que para la validación del sistema desarrollado se usará tecnología de punta con el fin de obtener resultados más precisos. Cabe recalcar que cada parte de este trabajo ha sido separado por

módulos de tal manera que se puedan mostrar resultados parciales en cada componente del proyecto.

1.1 Causas

Uno de los principales campos donde se observa la problemática descrita anteriormente es la agricultura, la cual presenta carencias en eficiencia y eficacia considerando también el hecho del decremento abrupto de la mano de obra dedicada a la agricultura. Tanto así que en los últimos años en países de escasos recursos más del ⅔ de la población se dedica a este tipo de trabajo, mientras que en países desarrollados es de tan solo el 5% como porcentaje máximo, dando así que el incremento de la productividad en este campo sea la causante de la reducción de la mano de obra [1], incremento que comenzó a tomar terreno gracias a la tecnificación de la agricultura y el aprovechamiento del combustible [2], lo cual presenta también sus desventajas, como lo es la sobreutilización de los recursos naturales y uso de productos químicos para lograr dicho crecimiento productivo en la mayoría de los casos. La Figura 1.1 muestra la proporción de la fuerza de trabajo de la agricultura en algunos países Europeos a lo largo de los años (1400 – 2012).

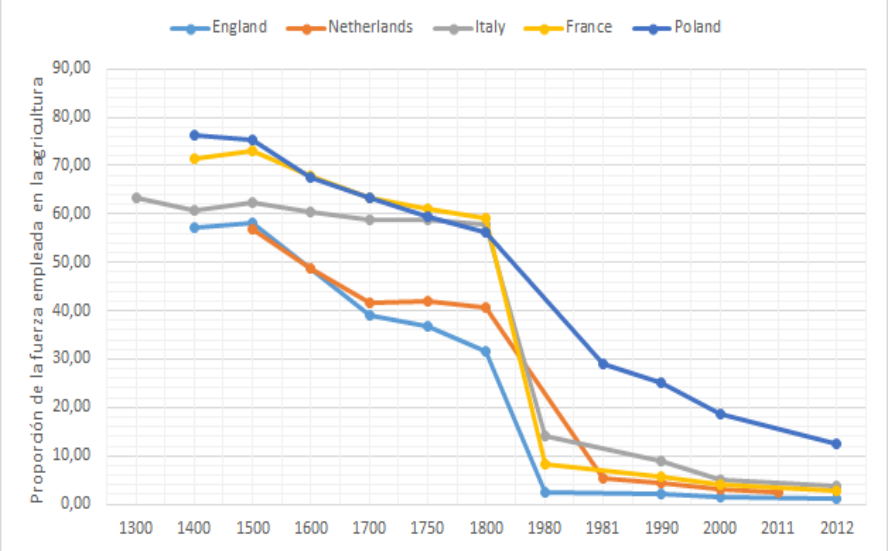


Figura 1.1. Proporción de la fuerza empleada en la agricultura [1].

Debido a esto se requiere que el porcentaje de optimización de los recursos sea cada vez mayor, de tal manera que estos sean utilizados en el momento y lugar adecuado, siguiendo un esquema de medición, procesamiento de información y actuación en base a los resultados obtenidos, proceso que está plenamente estructurado para ser realizado por un robot gracias al avance tecnológico tanto en hardware como en software, lo que hace posible que todos estos factores sean medianamente fáciles de calcular y dependiendo del caso, altamente fiables.

En la actualidad el avance tecnológico sorprende a la humanidad y no solo con la innovación de nuevos dispositivos celulares y computadores, sino también en campos como la robótica, con lo que se ha logrado hacer de los trabajos pesados una automatización casi completa con tan solo presionar unos botones, aun así, no se ha conseguido aún tomar todo el potencial que pueden ofrecer los mismos, ya que estos equipos aún dependen de las instrucciones humanas para su ubicación y movimiento.

Aunque se cuenta en la actualidad con avances tecnológicos en el campo de la robótica, hay áreas específicas en las cuales aún es necesario madurar, entre ellas la robótica agrícola, que aunque ha alcanzado un avance significativo [3], aún no es posible su aplicación generalizada debido a muchos factores, el principal de ellos es la alta demanda de recursos económicos necesarios para su construcción y mantenimiento [4], la Tabla 1 se muestra los precios de algunos robots dedicados a la agricultura.

Modelo Robot	**Precio**
Rover Robótico	$934.000
Robot Autopropulsado (“Wall-Ye”)	$32.000

Tabla 1. Precios de Robots usados en la agricultura [4].

Es debido al alto costo de las tecnologías lo que hace buscar una forma alternativa para su implementación, usando recursos accesibles y con gran precisión, los cuales incluirán dispositivos embebidos, módulos programables de navegación y una aplicación para la planificación de rutas.

1.2 Efectos y beneficios

Dado que este trabajo representa una base para una infinidad de proyectos, el impacto que presenta al avance tecnológico de nuestro país es grande, siendo el inicio de grandes aplicaciones para los robots autónomos que facilitaría procesos como: el recorrido de una plantación en un área agrícola, exploración de zonas de alto riesgo, entre otras.

Debido a los beneficios no solo económicos sino también de reducción de esfuerzo único que ofrece la era robótica, los gobiernos y la industria global de tecnología necesitan actuar de manera inmediata para identificar las fórmulas innovadoras que mejoren los esquemas de trabajo [5].

1.3 Soluciones similares

En la actualidad existen algunos proyectos que se encuentran en producción, los cuales ofrecen la funcionalidad de rutas planificadas, unos de estos productos es el drone Parrot Bedop [6], este drone además de permitirle al usuario realizar tomas aéreas manualmente, también crea un plan de vuelo por medio de una aplicación en su dispositivo móvil, de esta manera el drone efectuará la ruta planificada al mismo tiempo que efectúa tomas preestablecidas por el usuario.

Otro producto similar es el drone eBee [7], pensado para la agricultura. Por medio de una aplicación de escritorio, el usuario puede generar un plan de vuelo para que el drone obtenga imágenes multiespectrales del campo, para luego analizarlas y así detectar zonas específicas de cultivo, permitiendo así el ahorro de recursos al momento de la siega.

Como se puede observar, aunque existen soluciones similares, su campo es reducido en aplicaciones, a la vez de que estos productos no son desarrollados

como proyectos de código abierto, por lo que no pueden ser integrados en algún otro tipo de trabajo.

Muchas aplicaciones basadas en robótica cuentan con los accesorios necesarios para una adecuada integración del módulo de ruta planificada, pero no se cuenta con las respectivas herramientas de desarrollo para personalizarlas, dando al presente trabajo una innovación con herramientas simples y de fácil acceso.

CAPÍTULO 2

2. METODOLOGÍA.

Antes de detallar la metodología a utilizar para llevar a cabo el objetivo del proyecto, es necesario primero tener en cuenta algunos factores:

1. El alcance de este sistema solo abarca la navegación autónoma del robot, más las tareas complementarias a realizarse para poder cumplir con un fin específico como puede ser: el reconocimiento del estado de madurez de una fruta para su recolección dentro de la planificación del rumbo del robot, quedan para un futuro trabajo. Esto no impide el hecho de que el módulo sea fácilmente integrable con otros.
2. Ya que el sistema está enfocado a ser utilizado en áreas externas y de campo abierto e incluso en futuros usos con drones, el sistema no incorpora al robot la funcionalidad de detección y evasión de obstáculos a diferencia de muchos otros sistemas de navegación autónoma.
3. Las pruebas se han implementado en un robot controlado por las señales que la raspberry envíe por sus gpio, por lo que más adelante se detalla el proceso realizado para controlar a este robot en específico. Si se requiere la puesta en producción en un robot distinto con alguna funcionalidad en concreto, sería necesario solo la implementación del módulo de control de motores del nuevo robot, ya que por lo demás está pensado que sea fácilmente integrable, teniendo en cuentas sus debidas modificaciones.

La implementación de la solución se la ha diseñado de tal manera que se puedan utilizar sistemas embebidos por muchos factores, pero los que mayormente destacan son la portabilidad y la ejecución en tiempo real de tareas específicas, de tal manera que este módulo pueda ser utilizado en cualquier tipo de robot con las configuraciones necesarias según sea el caso. Adicional a esto se utiliza un sistema de navegación el cual proporciona con precisión, la posición y orientación del robot, sistema del cual se explicará su funcionamiento en secciones posteriores.

2.1 Requerimientos de hardware y software

El proyecto planteado implementa algunas tecnologías tanto en hardware como en software para llevar a cabo su solución, a continuación se muestran cada una de ellas de manera independiente, para luego proceder con la explicación de la integración de cada uno de ellos junto con los módulos desarrollados.

2.1.1 Raspberry PI3

La raspberry podría decirse que es el cerebro del robot, ya que es el que se encargará de ejecutar todo el proceso principal, este dispositivo es un pequeño computador a bajo costo a la cual se le puede añadir o incorporar un teclado, un mouse y un monitor. Lo que permite tener la experiencia de poseer un computador para el desarrollo en los lenguajes como Python, Scratch, etc.

En el presente trabajo se hace uso de la tercera generación de los raspberry, con las características que presenta la Tabla 2.

A 1.2GHz 64-bit quad-core ARMv8 CPU
802.11n Wireless LAN
Bluetooth 4.1
Bluetooth Low Energy (BLE)

Tabla 2. Características nuevas de RaspBerry PI 3.

2.1.2 Kit de desarrollo de navegación VectorNav VN200

El VectorNav es un sistema de navegación que consta de algunos sensores como el acelerómetro, giroscopio, entre otros, y es el dispositivo que permitirá obtener la orientación del robot. Poseedoras de la unidad de medición inercial (IMU) que a pesar de que acumulan errores conforme vaya siendo usado, el VectorNav mantiene un proceso de calibración para reducir ese porcentaje de error [8].

Para este trabajo se utiliza la versión VN-200 la cual incorpora además de los sensores previamente mencionados, un módulo GPS e IMU,

haciendo que estos datos se fusionen a través de un filtro de Kalman, incorporando lo último en avances sobre este filtro [9]. La Tabla 3 detalla las particularidades del módulo IMU.

Acelerómetro de 3 Ejes
Giroscopio de 3 Ejes
Magnetómetro de 3 Ejes
Sensor de presión
Módulo GPS

Tabla 3. IMU VectorNav VN-200.

Aunque es un dispositivo eficaz para la navegación, el mismo posee un conjunto de herramientas para la calibración, esto es debido a los campos magnéticos a los que pueda estar sujeto el dispositivo en su uso.

La Figura 2.1 muestra el dispositivo, en el cual en la entrada GPS va ubicada la antena que será encargada de captar los satélites a su disposición, dicha antena es capaz de generar un gran campo magnético, debido a esto el módulo de desarrollo ya considera este campo con la finalidad de contrarrestarlo de tal manera que no afecte la obtención de los valores y así poder trabajar junto al mismo sin ningún problema. Adicional a esto el kit de desarrollo facilita un conjunto de librerías e interfaces escritas en C para poder interactuar con él.

Figura 2.1. VectorNav VN-200.

2.1.3 Leaflet

Leaflet es una librería opensource que permite la creación de mapas interactivos para ser embebidos en un ambiente móvil o web.

Algunas de sus ventajas se muestran a continuación:

- Sencillo y fácil de aprender
- HTML 5 y CCS3
- Funcional tanto en antiguas como nuevas versiones de los navegadores
- Excelente documentación
- Más ligero

La razón por la cual se optó a usar esta librería fue debido a su fácil e inmediata incorporación a la aplicación, además de que las funcionalidades requeridas se encontraban al alcance de la mano. Esta herramienta es usada por Flickr, Wikipedia y Foursquare [10].

2.1.4 Módulo L298N

Módulo que permite controlar los motores DC del robot tanto en sentido de giro como la velocidad.

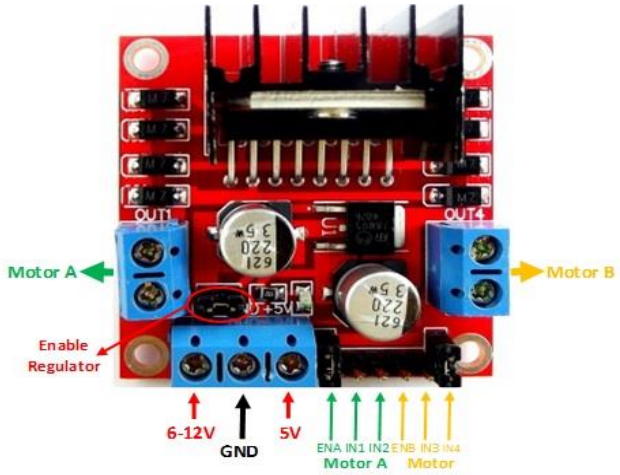


Figura 2.2. Módulo L298N.

Contiene un conector de 6 pines que serán los encargados de manejar los motores tanto para habilitarlos y manejar su sentido de giro, borneras de salida para los motores y una bornera de 3 pines para la alimentación del mismo, la Figura 2.3 muestra el esquemático de la conección con la raspberry pi.

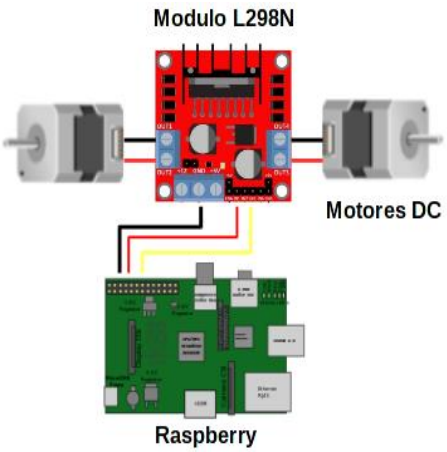


Figura 2.3. Esquema de conexión módulo 298 – raspberry.

2.1.5 Flask

Flask es un framework implementado en python, el cual nos permite de una manera rápida y sencilla poder crear aplicaciones web. Entre sus características destacables se tiene:

- Consta con su propio servidor web, es decir no es necesario de la instalación de otros servidores externos como Apache o Nginex.
- Con pocas líneas de código se puede tener una aplicación lista para empezar a recibir y enviar peticiones.
- Fácil integración con socketIO.
- Soporte para realizar pruebas unitarias y debugging.

2.1.6 Xfuzzy3

Es un entorno de desarrollo de software libre que puede ser ejecutado sobre cualquier plataforma que disponga del "Java Runtime Environment" (JRE), el cual facilita la creación, simulación y monitorización de sistemas difusos, así como también la creación de compilados para lenguajes como C, C++ y JAVA, facilitando de esta manera la integración del sistema al proyecto.

2.2 Diseño de la solución

La navegación autónoma dada una ruta planificada no es una tarea trivial, pues abarca una serie de subtareas que se deben de llevar a cabo para poder lograrlo, estas tareas se listan a continuación:

- Obtención de los puntos geográficos, dada una ruta planificada
- Muestreo de los puntos geográficos de tal manera que sean equidistantes
- Determinación de la posición absoluta del robot (Latitud y Longitud)
- Determinación de la orientación del robot (Yaw , Pitch , Roll)
- Cálculo de la distancia entre 2 puntos geográficos
- Cálculo del ángulo entre 2 puntos geográficos con respecto al norte magnético
- Manejo del control del robot

2.3 Obtención de los puntos geográficos, dada una ruta planificada

Se ha usado ell ámbito web, usando una librería opensource llamada “leaflet” en la cual se le presenta al usuario la oportunidad de graficar una ruta si así lo desea. Para poder obtener ese resultado final, el usuario deberá seguir ciertos pasos que detallaremos a continuación.

Para poder hacer un reconocimiento rápido de la zona, la aplicación le pedirá aceptar los términos de localización para mostrar el mapa con su ubicación actual, como lo ilustra la Figura 2.4.

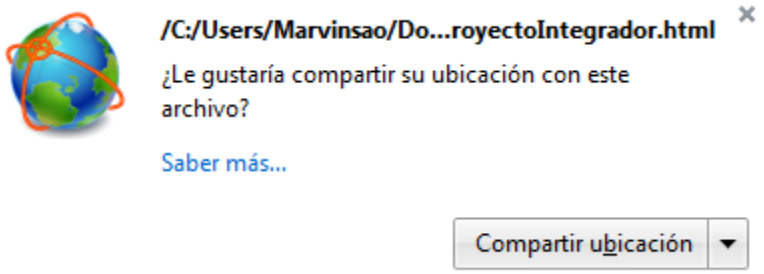


Figura 2.4. Compartir ubicación en el navegador.

Una vez mostrado el mapa, el usuario es capaz de determinar una ruta a seguir, cabe mencionar que la forma en que podrán realizarlo es tomando en consideración lo siguiente:

- Click para empezar a crear la ruta
- Mueva el mouse para dibujar la ruta
- Click para terminar la gráfica de la ruta

Siguiendo esas indicaciones, se mostrará la ruta dibujada dentro del workspace del mapa, quedando de la manera que lo muestra la Figura 2.5.

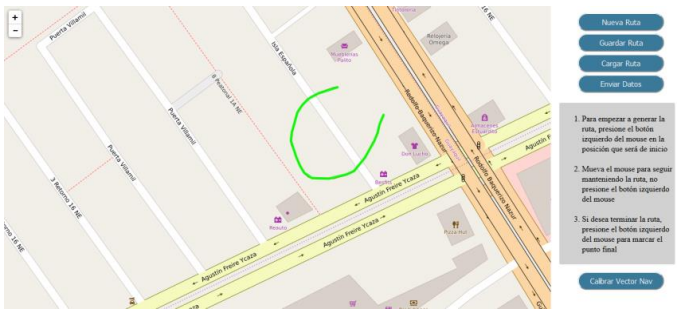


Figura 2.5. Gráfico de ruta en la aplicación.

Una vez generada la ruta y tomado los puntos GPS que conforman dicha ruta se procede a realizar un remuestreo a los mismos de tal manera que no exista demasiado dato redundante y a la vez disminuyan los cantidad de puntos haciendo que sea más eficiente los cálculos de distancia entre un par de puntos.

2.4 Muestreo de los puntos geográficos

Para el muestreo de los puntos geográficos obtenidos por el usuario, se establece un umbral de 5 metros como mínimo entre punto y punto, de tal manera que el proceso de identificación de puntos no tenga un costo elevado.

Dado también que la generación de la ruta en la aplicación depende de qué tan rápido realice el usuario un trazo, se identificó que entre más lento o preciso se quiere ser, se genera mucho dato redundante o basura, dando paso a que una pequeña ruta generaría cerca de 250 puntos y entre cada par de puntos continuos no existe más 50 cmts de distancia, por lo que la práctica de este muestreo nos sirve para disminuir esta cantidad de puntos un 60%.

2.5 Determinación de la posición absoluta del robot

Determinar la posición de un robot en ambientes adversos y poco estructurados es algo relativamente complicado, más aún cuando existen errores no intencionados en la captura de datos debido a los sensores usados y a los sistemas propios que controlan el robot [11].

Es debido a esto que existen técnicas que permiten determinar la posición del robot, sea esta relativa o absoluta [12], una de las más utilizadas debido a su bajo coste y su alta fidelidad a corto plazo es la odometría, la cual es una técnica que permite estimar la posición relativa del robot, es decir que en vez de tomar como referencia las “coordenadas del mundo” toma como referencia un punto de inicio, el cual es identificado como el punto inicial del cual se empieza a tomar datos para el cálculo de la odometría, cálculos que varían dependiendo de la cinemática del robot (configuración de motores y ruedas del robot) [13] la Figura 2.6 muestra un esquemático sencillo de la cinemática diferencial. Sin embargo, aunque esta técnica puede resultar beneficiosa en muchos casos, la acumulación de errores ya sean por factores externos como el ambiente con el que interactúa el robot, o internos como la resolución de los encoders, representa un gran inconveniente [14].

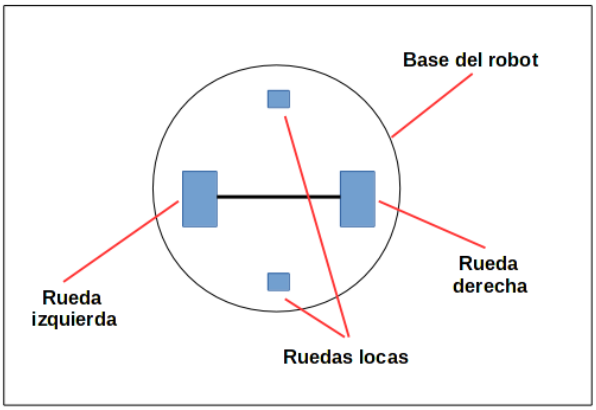


Figura 2.6. Cinemática diferencial del robot.

Por otro lado los métodos de posicionamiento absoluto permiten determinar la posición del robot tomando como referencia las coordenadas geográficas, ayudándose de sensores que miden la relación que existe con el entorno, como lo son: el acelerómetro, el giroscopio, el magnetómetro y los receptores GPS.

Debido a que para el proyecto se propone que el usuario pueda trazar la ruta que desee en cualquier lugar del mapa, y de que el robot pueda ser capaz de orientarse y posicionarse en el punto de inicio para luego proceder a seguir la ruta planificada todo de una manera autónoma, es necesario el uso del posicionamiento absoluto.

El uso de este método de posicionamiento también tiene algunos inconvenientes; este depende mucho del grado de precisión y latencia en la obtención de los datos por parte del receptor GPS, el cual aunque ha tenido una notable mejora en la precisión de los datos a lo largo de los años, la cual puede ser contrastada desde su época inicial en la cual solo se tenía una precisión no menos de 10 metros, pasando por el sistema GPS diferencial [15] usado actualmente y que nos provee precisiones de hasta 1 metro, y muy pronto llegando a obtener precisiones de 1 cm [16], es inevitable la obtención de errores causados por ruidos en el sistema. Es por ese motivo que para la obtención de estos datos se utilizará un dispositivo de gama alta el cual es el VectoNav VN-200, utilizado en sistemas de control y posicionamiento aéreo, el cual entre sus características está la implementación del filtro de kalman [17] para mejorar la estimación de la posición y así contrarrestar el error capturado

por el dispositivo. En cuanto a la implementación, el dispositivo cuenta con una librería escrita en C, que sirve de interfaz para la obtención de los datos, por lo que se creó un modelo cliente-servidor, de tal manera que el proceso principal pueda consumir este servicio por medio de paso de mensajes.

2.6 Determinación de la orientación del robot

Para determinar la orientación del robot se utiliza el sistema de ángulos de navegación, los cuales son utilizados para poder determinar la orientación de un objeto en un sistema tri dimensional, que dependiendo del área en el cual sean usados reciben varios nombres como: cabeceo, alabeo y guiñada en sistemas de navegación aérea, o deriva, inclinación y escora si se trata de sistemas de navegación marítima. Para este apartado se hace referencia a ellos utilizando la nomenclatura aérea en inglés: Yaw, Pitch, Roll (Como se observa en la Figura 2.7), pero para ser más exactos solo trabajaremos con el ángulo Yaw ya que el robot solo necesita orientarse dentro del plano XY, es decir, el eje de rotación a utilizar será el eje Z.

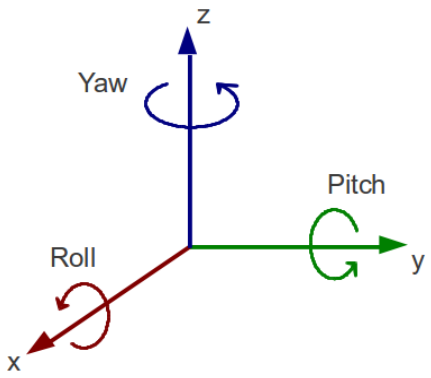


Figura 2.7. Ángulos de navegación.

Para la determinación de este ángulo, se hace uso nuevamente del VectoNav VN-200 dado su sistema de referencia AHRS (Sistema de Referencia de Actitud y Rumbo), formado por giroscopio, acelerómetro y magnetómetros en los 3 espacios, que a la vez nos permite usar el receptor GPS para poder obtener una mejor estabilidad en los giroscopios aplicando filtros de Kalman, siendo de gran apoyo al antiguo sistema de navegación inercial que se basaban en el magnetómetro y datos sin procesar del giroscopio mediante los receptores GPS.

2.7 Cálculo de la distancia entre 2 puntos geográficos

Debido a que es necesario considerar la curvatura terrestre para el cálculo de la distancia entre 2 puntos geográficos tal como lo muestra la Figura 2.8, no es algo relativamente sencillo de obtener, por tal motivo existen algunas fórmulas que ayudan con el cálculo de esa distancia tomando en cuenta ese factor, entre ellos están la Ley Esférica del Coseno y la fórmula del Haversine [18].

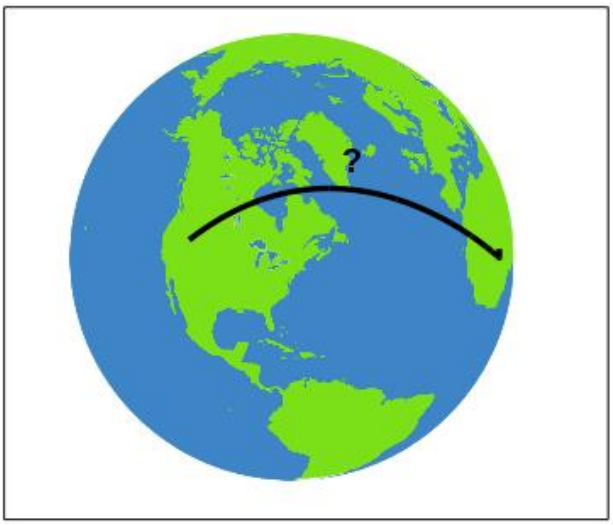


Figura 2.8. Distancia entre 2 puntos geográgicos.

Para esta implementación se ha optado por seleccionar la fórmula del Haversine ya que en este proyecto trabaja con rango de distancias relativamente pequeñas (no mayor a 10 metros entre puntos) y este método nos ofrece una mayor precisión al trabajar con este tipo de distancias [19].

Este método facilita el cómputo de la distancia recibiendo como parámetros los puntos geográficos (latitud y longitud) de origen y destino, a continuación se muestran las fórmulas correspondientes:

$$a = \sin^2(\Delta \varphi / 2) + \cos(\varphi 1) \cdot \cos(\varphi 2) \cdot \sin^2(\Delta \lambda / 2) \quad (2.1)$$

$$c = 2 \cdot \arctan(\sqrt{a}, \sqrt{(1-a)}) \quad (2.2)$$

$$d = R * C \quad (2.2)$$

Dónde:

Δ φ = Diferencia de las latitudes de origen y destino en radianes.

$\Delta\lambda$ = Diferencia de las longitudes de origen y destino en radianes.

a = Cuadrado de la mitad de la longitud entre los puntos $\Delta\varphi$ y $\Delta\lambda$.

c = Distancia angular en radianes.

R = Radio equivolumen de la tierra (6371 km).

d = Distancia entre los 2 puntos geográficos.

2.8 Cálculo del ángulo entre 2 puntos con respecto al norte geográfico

Conociendo la distancia entre los 2 puntos geográficos y la orientación del robot, el único factor que hace falta calcular para poder direccionarlo, es el ángulo entre el punto de origen y de destino. Este ángulo es conocido como Azimuth, el cual hace referencia a la orientación de un objeto en una superficie esférica, el cual dependiendo del contexto en que se lo usa, sus puntos de referencia y orientación varían. Se lo utilizará según el contexto náutico el cual toma como referencia el norte magnético en sentido horario de 0 a 360 grados como se puede observar en la Figura 2.9.

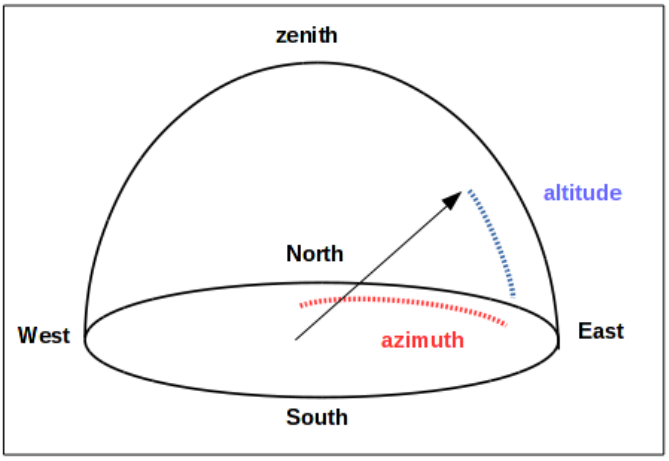


Figura 2.9 Ilustración del azimuth y altitud utilizados en los sistemas de navegación.

La ecuación 2.5 la cual deriva de Ley Esférica del Coseno mencionada anteriormente, ayuda con el cálculo de este ángulo, la cual depende en gran medida de los puntos que son referenciados como inicio y fin.

$$y = \sin \Delta\lambda \cdot \cos \varphi 2 \quad (2.3)$$

$$x = \cos \varphi 1 \cdot \sin \varphi 2 - \sin \varphi 1 \cdot \cos \varphi 2 \cdot \cos \Delta\lambda \quad (2.4)$$

$$\theta = \arctan(y , x) \quad (2.5)$$

De esta manera se puede saber a cuantos grados se encuentra el punto de inicio al punto objetivo con respecto al norte magnético, este valor luego es analizado junto con el ángulo de orientación del robot para calcular el ángulo final de giro, la Figura 2.10 muestra un ejemplo del cálculo de este valor.

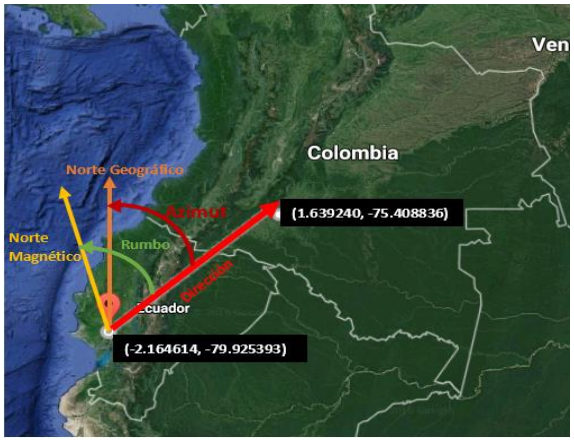


Figura 2.10 Ángulo de azimut entre 2 puntos geográficos.

2.9 Manejo del robot

Como se lo mencionó anteriormente la odometría es una técnica la cual permite saber la posición relativa del robot, con respecto a un punto de inicio tomado como referencia. Esta técnica utiliza ecuaciones odométricas que dependen en gran medida de la cinemática del robot, y es por medio de estas ecuaciones que se puede estimar el avance del robot en un tiempo determinado.

La ecuación a utilizar varía dependiendo del tipo de robot no sólo en cuanto a su cinemática sino también en cuanto al diámetro de las llantas y a la resolución de los encoders, los cuales son sensores que dependiendo del tipo, monitorean de una u otra manera la codificación de un disco rotatorio, convirtiendo estos valores en pulsos los cuales permiten determinar el ángulo de giro del motor en un tiempo determinado. La precisión de este valor depende en gran medida de

la codificación del disco rotatorio, por lo que sabiendo esto y conociendo el diámetro de las llantas, se puede determinar la distancia la cual ha avanzado el robot con un error controlable, la Figura 2.11 muestra el encoder y el disco utilizado.

Figura 2.11 Encoder y disco rotatorio.

Aunque la técnica detallada es una de las más comunes al momento de querer que el robot avance una cierta distancia o gire un determinado número de grados, al realizar las pruebas se obtuvieron errores bastante significativos. Esto debido a que a pesar de que los encoders para cada motor tenían en teoría diferente disco rotatorio con la misma codificación girando a la misma frecuencia, los encoders no sensan el mismo valor, dando como resultado que el robot al finalizar su acción termine con un giro incierto, lo que ocasiona un error muy significativo para el seguimiento de la ruta.

Debido a esto se pensó en otra solución que abarca el manejo de las señales PWM (pulse width modulation) que envía la raspberry a los motores a través del módulo h, de tal manera que aumentando este valor en un motor y disminuyendolo en el otro se pueda conseguir el control y giro requerido como lo muestra la Figura 2.12.

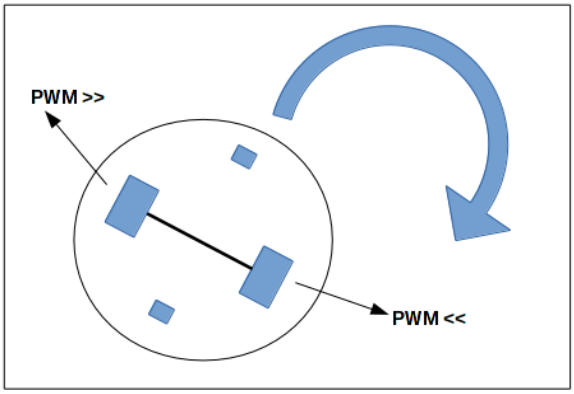


Figura 2.12 Giro del robot utilizando los gpio de la raspberry.

Ya que no existe una ecuación matemática para poder determinar los valores de PWMD (valor PWM asignado al motor derecho) ni PWMI (valor PWM asignado al motor izquierdo) dado un ángulo de giro que se desee que el robot efectúe, se ha implementado un sistema difuso que controle en tiempo real el control de estas señales, sensando en todo momento los grados a los que se encuentra el robot del objetivo, haciendo uso del VectoNav.

Para la implementación del sistema se utilizó funciones de pertenencia triangulares debido a la facilidad de su cómputo en tiempo real, además del uso de 2 variables las cuales son:

- **YAW :** Variable de entrada que describe los grados hacia su objetivo. Está compuesta por 4 etiquetas que describen las curvas a realizar por el robot, tal lo muestra la Figura 2.13.

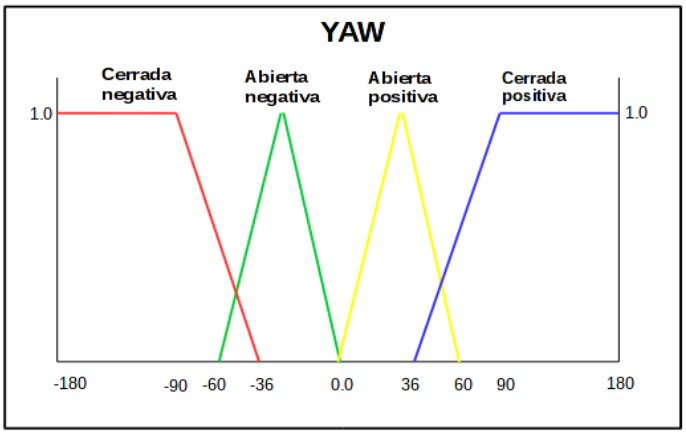


Figura 2.13 Variable de entrada YAW la cual representa los grados del robot hacia su objetivo.

- **PWM:** Variable de salida la cual representa el valor de PWM que se le asignará al gpio correspondiente de cada motor. Debido a que son 2 motores el mismo tipo de variable es usado para ambos, pero en la práctica son consideradas como 2 variables independientes (PWMD y PWMI) ver Figura 2.14.

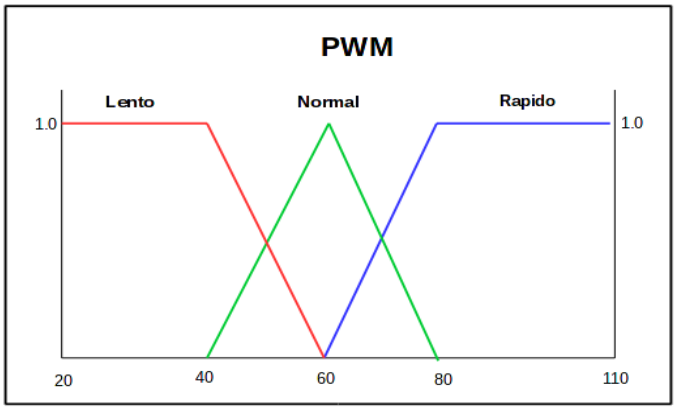


Figura 2.14 Variable de salida que representa el valor de PWM a asignar a cada motor para llevar a cabo el giro correspondiente.

El sistema al recibir un ángulo de giro como entrada, empieza a sensar en tiempo real la orientación a la cual se encuentra el robot de su ángulo destino, por lo cual en t = 0 la variable YAW toma el valor del ángulo, mientras que en t = Ω la variable YAW toma el valor de 0, donde Ω es el tiempo que le tomó al robot en efectuar el giro provisto como entrada.

De esta manera el sistema en tiempo real reaccionará a estos valores, modificando el PWM asignado a cada motor para alcanzar su ángulo objetivo utilizando las reglas de inferencias mostradas en la Tabla 4.

YAW	PWMD	PWMI
CERRADA_NEGATIVA	LENTO	RÁPIDO
ABIERTA_NEGATIVA	LENTO	NORMAL
ABIERTA_POSITIVA	NORMAL	LENTO
CERRADA_POSITIVA	RÁPIDO	LENTO

Tabla 4. Reglas de inferencia para mover los motores

2.10 Integración de los módulos

Listados los requerimientos de hardware utilizados, y las tareas que resolverán individualmente el problema, a continuación se explica la integración de todos los componentes para llevar a cabo la navegación autónoma del robot.

La Figura 2.15 muestra un esquemático sencillo de la funcionalidad del sistema, en donde el usuario desde un dispositivo móvil o pc podrá conectarse a la aplicación, aquí podrá dibujar la ruta deseada para que el robot la siga de manera autónoma, a continuación se procede a realizar un muestreo de los puntos geográficos generados por el usuario de tal manera que exista una distancia mínima promedio entre ellos para posteriormente enviarlos al servidor montado en la raspberry, este se encargará de llamar al módulo del sistema de navegación quien direccionará al robot dado los puntos geográficos enviados como parámetro.

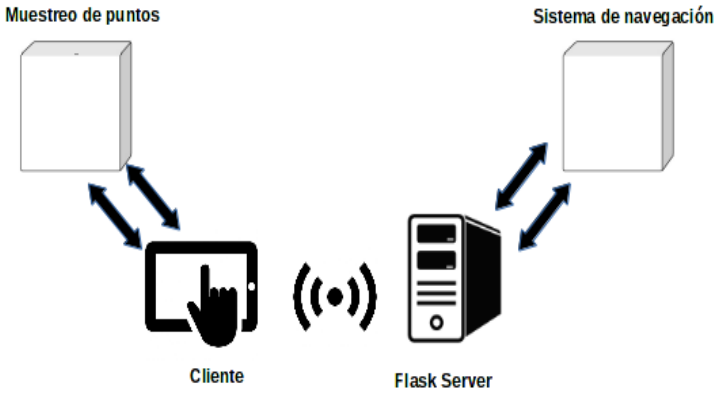


Figura 2.15. Esquemático sencillo del flujo del sistema.

Este módulo a su vez es la implementación de cada una de las tareas detalladas anteriormente como módulos independientes que ofrecen interfaces para la comunicación entre ellos, la Figura 2.16 muestra el diagrama de componentes de este sistema, el cual consta de algunos módulos.

El main o principal escrito en python es el módulo que sirve de interface con el servidor, y que además es quien se encarga de interactuar con los demás módulos, el módulo VectorNav es un servicio escrito en C que se encarga de obtener los puntos geográficos y orientación provistas por el dispositivo en cuestión, el módulo de calibración se encarga de obtener la orientación en un instante de tiempo de tal manera que se pueda referenciar al norte magnético, esto debido a que el VectorNav no entrega un valor de YAW = 0 al encontrase referenciado a él, luego tenemos el módulo de obtención de la distancia y ángulo entre 2 puntos geográficos enviados como parámetros, y por último el módulo de control de motores es quien se encargará de controlar el PWM enviado por la raspberry a los motores del robot, este módulo se encuentra escrito en C gracias a que el software utilizado para crear el sistema difuso facilita un compilado en este lenguaje, por lo que por temas de integración con el proceso principal, se procedió a crear un envoltorio de este módulo para python.

Antes de que el usuario pueda realizar la petición al servidor para que el robot se mueva por la ruta planificada, el sistema asume que el usuario ha calibrado el dispositivo VectorNav con el norte magnético llamando al módulo de calibración, generando así un offset el cual es guardado en un archivo aparte para luego ser leído por el proceso principal.

El módulo principal al recibir la lista de puntos geográficos seguirá el flujo que se detalla a continuación para cada punto hasta haber recorrido toda la lista.

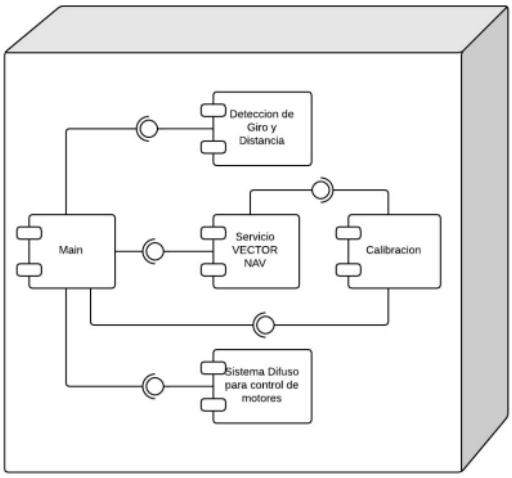


Figura 2.16. Diagrama de componentes del sistema de navegación.

1 El módulo principal consume del servicio VectorNav para obtener la posición y orientación del robot (latitud , longitud , yaw).

2 Se resta el offset obtenido por el proceso de calibración del ángulo de orientación, para obtener la orientación correcta del robot.

3 Se llama al módulo de detección de giro y distancia para obtener la distancia y ángulo de giro entre la posición en la cual se encuentra el robot, y el punto correspondiente del arreglo, para esto también se envía como parámetro el ángulo de orientación del robot (yaw) de tal manera que el módulo pueda calcular este ángulo considerando este parámetro, por defecto si este valor es 0 el módulo calcula el ángulo tomando como referencia el norte magnético.

4 Se comprueba si la distancia entre la posición del robot a la posición del punto destino es menor a un umbral, si éste es el caso se prosigue con el siguiente punto geográfico en la lista en el caso de que lo haya y se repite el paso 1, sino continúa con el flujo del proceso.

5 Se envía como parámetro al sistema difuso de control de motores el ángulo al que se encuentra el robot a su punto de destino, de esta manera se controla el direccionamiento del robot, para luego regresar al paso 1.

2.11 Aplicación Android

Para realizar un mejor manejo en la integración entre los módulos antes mencionados, se realiza una aplicación nativa en android usando Android Studio llamada “RouGen” de tal manera que, mediante esta se conecta a la raspberry y a la vez con la aplicación mediante una conexión WI-FI.

Dentro de la cual se proporciona al usuario las opciones de conexión, calibración del sistema GPS y como parte principal de la aplicación la generación y ejecución de la ruta deseada.

Como primer paso, el usuario conecta el raspberry con la aplicación en android usando el dispositivo hotspot, de tal manera que le provea una dirección IP y se pueda acceder al sistema en la raspberry.

Una vez establecida la conexión, accedemos a los archivos dentro de la raspberry hasta llegar a “run.py” un script que nos va a permitir levantar el servidor que permitirá la comunicación mediante sockets entre la aplicación de android y la raspberry.

Tanto en el paso uno como en el paso 2 se generan 2 datos importantes, que son la dirección IP para conectarnos con el raspberry y el puerto que nos va a permitir comunicarnos con el servidor, datos que serán colocados en la página de inicio de la aplicación como se muestra la Figura 2.17:

Figura 2.17. Login RouGen.

Una vez que hemos logueado se genera una nueva interfaz en la que se puede interactuar con un mapa y ciertas funcionalidades que se le da la aplicación, como lo pueden ver en la Figura 2.18. Un dato que se considera es que la aplicación puede llegar a este punto sin que se haya logueado, pero al momento de querer ejecutar las funciones, la misma le informará de la falla de conexión.

Fig 2.18. Pantalla para generar la ruta

Dentro de las funcionalidades proporcionadas por la aplicación podemos listar las siguientes:

- Calibrar Robot
- Deshacer
- Reiniciar Ruta
- Cerrar Ruta
- Iniciar Recorrido
- Parar Recorrido

Calibrar Robot.- Función que me permite interactuar con el Vector Nav, de tal manera que pueda identificar hacia dónde se encuentra el norte magnético y a cuantos grados se encuentra el frente del carro del mismo. El sistema puede dar 3 resultados a esta operación, que son:

- **Calibración Satisfactoria:** Todo está correcto
- **Se completó la calibración pero contiene algunos errores:** Es un aviso de que debe volver a efectuar la calibración, se da cuando existen demasiada varianza entre los valores tomados.

- **No se ha podido calibrar:** Se da cuando no se receptan valores del Vector Nav.

Deshacer.- Deshace el último punto colocado en el mapa.

Reiniciar Ruta.- Deshace todos los puntos realizados en el mapa, para que el usuario pueda empezar a crear una nueva ruta.

Cerrar Ruta.- Una vez marcado el primero punto que será el punto de partida, la función unirá el punto inicial de la ruta con el final, de tal manera que se pueda crear una ruta cerrada.

Iniciar Recorrido.- Ya con la ruta en el mapa y realizada la respectiva calibración, se usa esta función para empezar el recorrido cubriendo todos los puntos GPS marcados.

Detener Recorrido.- En caso de alguna emergencia, este botón puede ser considerado como una “freno de emergencia” para que finalice inmediatamente la ejecución del recorrido.

CAPÍTULO 3

3. PRUEBAS Y RESULTADOS DE SOLUCIÓN.

Las pruebas finales tienen como objetivo analizar el comportamiento y a la vez la precisión del vehículo a escala al momento de establecer una ruta para cumplir con los objetivos propuestos en el proyecto.

3.1 Construcción del primer prototipo “Milagrito”

Para esta fase de construcción, se hace uso de un modelo base de la estructura del auto a escala que consta de las siguientes partes:

- 2 Motores
- 2 Llantas
- Chasis
- Módulo L298N

Teniendo la estructura de nuestro vehículo se empieza a realizar las conexiones necesarias para el funcionamiento, es decir, conectar los motores a las borneras del Módulo L298N para que este les administre el voltaje a cada uno y este pueda rodar.

Se realiza el montado del dispositivo embebido (raspberry) y GPS sobre el chasis del auto y a la vez realizar la conexión respectiva de los motores a los pines de la raspberry así como el GPS, tal y como se aprecia en la Figura 3.1.

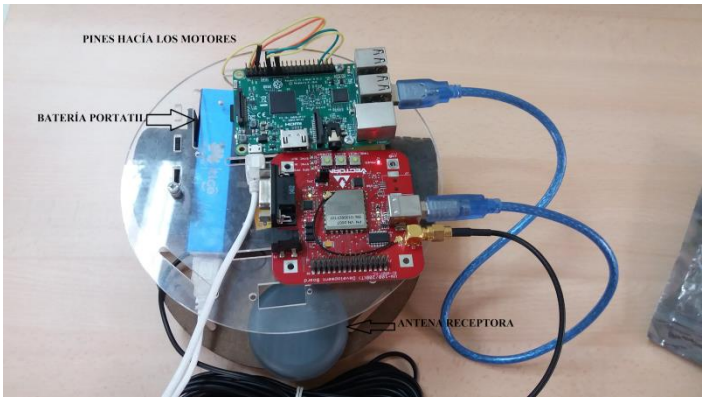


Figura 3.1. Dispositivo embebido y GPS montado en chasis.

En la Figura 3.1 se puede apreciar que la raspberry está siendo alimentada por una batería portátil y el GPS mantiene conectada la antena receptora la cual está posicionada sobre la base del vehículo, quedando de esta manera armado el vehículo a escala que para fines del proyecto fue llamado “Milagrito”, el cual está listo para las pruebas de control.

3.2 Prueba del sistema difuso para el control de motores

Antes de poner al robot en marcha, se vio la necesidad de poner a prueba el sistema difuso para el giro de este, y así poder comprobar el error que podría existir en un ambiente real de pruebas, la tabla 5 muestran estos resultados.

Angulo	PWMI	PWMD	Angulo Final	Error
-109	33.55	89.49	-114.14	5.14
-50	38.03	83.9	-51.45	1.45
-29	35.70	60	-32.67	3.67
60	87.16	37.73	62.92	2.92
170	89.49	35.55	174.45	4.45
152	89.49	35.5	157.36	5.36
50	74.96	38.39	52.76	2.76
68	87.77	37.15	71.32	3.32
14	60	37.67	15.26	1.26
43	67.01	37.32	46.4	3.4
-63	37.73	87.16	-65.7	2.7

Tabla 5. Valores resultantes del sistema difuso.

Aquí se observa los valores PWM asignados a cada motor por el sistema difuso, en el cual para ángulos negativos (en contra de la manecilla del reloj) el sistema asigna mayor potencia al motor derecho mientras que para ángulos positivos la

mayor potencia va al motor izquierdo. Adicional a esto también se puede visualizar el ángulo final de giro que dio el robot, el cual como se puede observar tiene un error no muy significativo, siendo el error máximo obtenido de 5.36 grados.

3.3 Pruebas funcionales con Milagrito

Para empezar a realizar estas pruebas, se lo ha divido en algunas fases de control, para determinar ciertos factores que podrían hacer que el sistema falle.

3.3.1 Pruebas de distancia máxima de conexión

Para poder iniciar las pruebas, se presenta la necesidad de establecer la distancia máxima en la que los dispositivos pueden establecer una conexión, con el fin de mantener comunicación para el paso de datos y/o generar otra acción desde la aplicación, dado que existen factores variables como la IP generada por el hotspot al conectarse con la raspberry que puede presentar un obstáculo para realizar una reconexión.

Con esta información previa obtenida, se procede a realizar una prueba de distancia de conexión, dando como resultado que la máxima distancia debido al dispositivo celular usado como hotspot es de 10mts.

Una vez que se proporciona la IP de conexión con la raspberry al hotspot, se ingresa a la interfaz del dispositivo embebido para levantar el servidor que comunicará las acciones desde la aplicación celular hacia el programa en la raspberry, estos datos como la IP y el puerto que aparece al momento de levantar el servidor serán los que se necesitan para crear la comunicación y son ubicados en la aplicación como lo muestra en la Figura 3.2.

Fig 3.2. Comunicación aplicación android con raspberry.

3.3.2 Prueba de seguimiento de ruta

Una vez que se ha determinado la distancia máxima para tener una conexión con el dispositivo embebido y la aplicación, se procedió a generar la primera ruta en la aplicación para que “Milagrito” sea capaz de seguirla y poner a prueba. En la Figura 3.3 se encuentra la ruta que se ha creado para realizar la prueba de “Milagrito”.

Figura 3.3 Ruta de prueba “Milagrito”.

Ejecutando la prueba de seguimiento de ruta surge la necesidad de comprobar el éxito de la prueba, para lo cual se realiza la medida de la distancia que el robot debe cruzar con la que cruza, mostrando los resultados en la Figura 3.4.

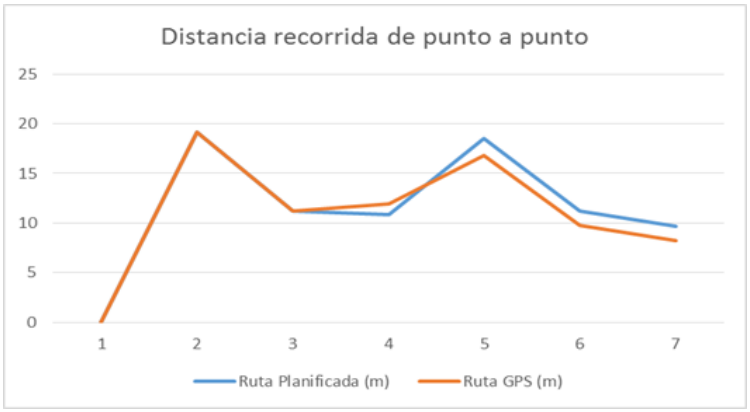


Fig 3.4. Distancia a recorrer vs Distancia recorrida.

Analizando la Figura 3.4, se ha obtenido los respectivos valores de error para cada punto que el robot ha recorrido en la ruta propuesta para la ejecución de la prueba. En la Figura 3.5 se muestran los errores porcentuales obtenidos en el recorrido por cada punto de la ruta, demostrando la precisión en el sistema al ejecutarse.

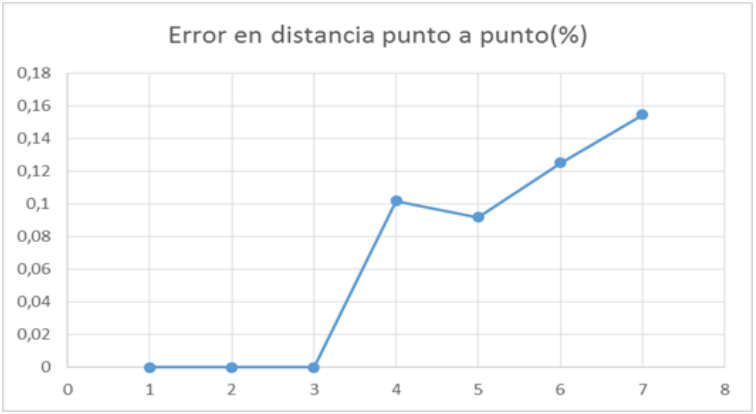


Figura 3.5. Error porcentual por cada punto de la ruta.

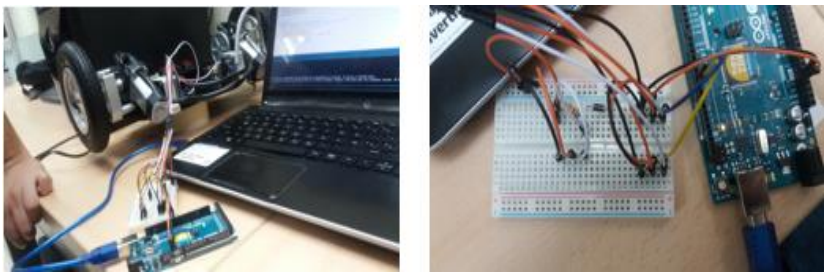
3.4 Integración del módulo con un segundo robot “Mashi”

En el capítulo anterior se mencionó la posibilidad de poder integrar el módulo, dado que se quiso probar este acontecimiento realizamos una integración en la plataforma móvil de un robot el cual es denominado como “Mashi”, en el cual se emplearon los siguientes componentes:

- Arduino
- Protoboard
- Batería 12V

Estos nos permitieron realizar de manera exitosa la comunicación entre nuestro sistema con la plataforma móvil.

Para poder realizar la comunicación con los encoders de la plataforma que controlan a los motores con el arduino, se vio la necesidad de hacer una interfaz de comunicación que consta de una resistencia y un diodo, la cuál va conectada a los pines del arduino y hacia los encoders, como lo vemos en la Figura 3.6(a) y Figura 3.6 (b).



(a) (b)

Figura 3.6. (a) Conexión interfaz de comunicación y encoders. (b) Conexión interfaz de comunicación y arduino.

Una vez creada la comunicación con los motores, se procede a realizar el montado de “Milagrito” con el “Mashi”, de tal manera que se pueda comunicar la raspberry y el arduino, para completar la integración de los mismos. En la Figura 3.7 se puede observar el montaje final de la integración.

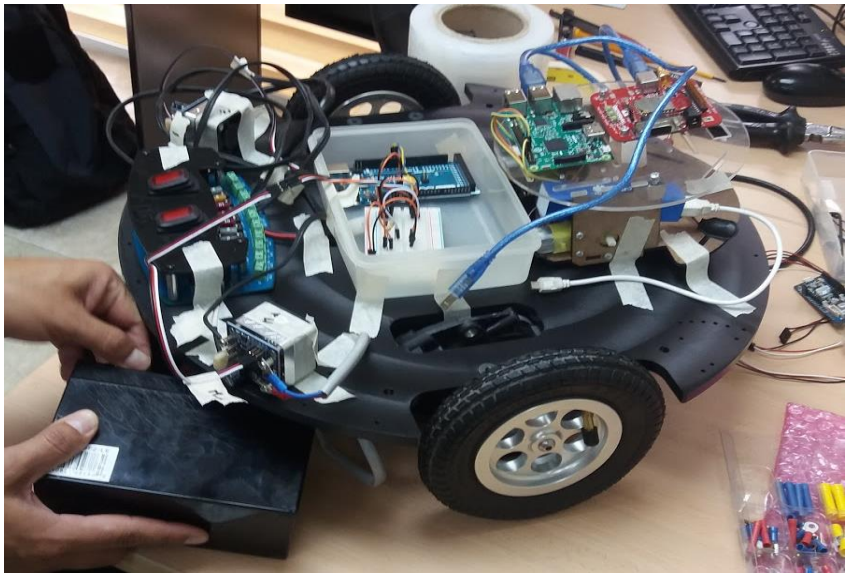


Figura 3.7. Integración de módulo en “Mashi”.

3.5 Pruebas funcionales con “Mashi”

Dada la necesidad de probar la parte funcional de la integración, se realizan las mismas pruebas que en “Milagrito”, tanto la prueba de conexión como la de distancia máxima que se han probado con el primer robot, se han probado con ésta integración obteniendo un resultado satisfactorio.

La programación que llevan los motores en la integración contiene un delay de 1 segundo, sumando que las instrucciones de potencia para cada motor se envían por separado, esto hace que se tome otras medidas en cuanto al control de los mismos, para lo cual se realiza una nueva prueba e implementación. Considerando estos nuevos factores para la solución, se integra una pequeña sección de código potenciando una unidad más al motor que está recibiendo primero la instrucción de moverse.

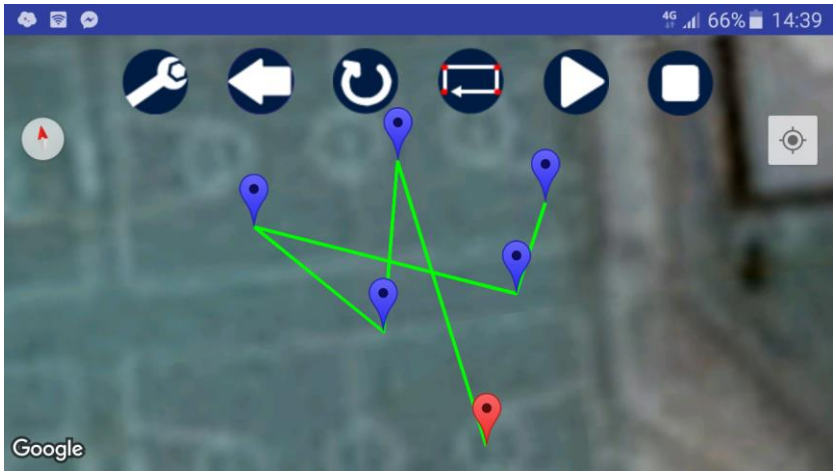


Figura 3.8. Ruta de prueba (“Mashi”).

En la Figura 3.8 se muestra la ruta que se ha designado al robot resultado de la integración, aunque la ruta es diferente que la usada en el primer robot los resultados obtenidos fueron muy similares en cuanto a la distancia que debía recorrer en el nuevo escenario, como lo pueden observar en la Figura 3.9.

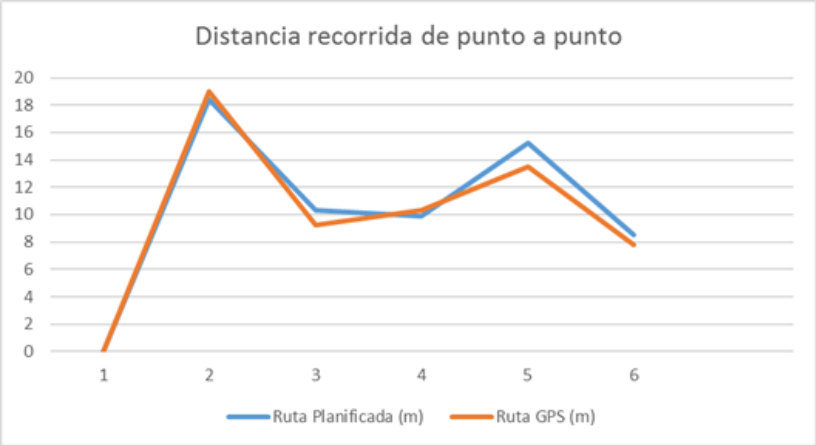


Fig 3.9. Distancia a recorrer vs Distancia recorrida (“Mashi”).

De la misma manera se elabora la gráfica de error porcentual en cada punto en el nuevo escenario, véase en la FIgura 3.10.

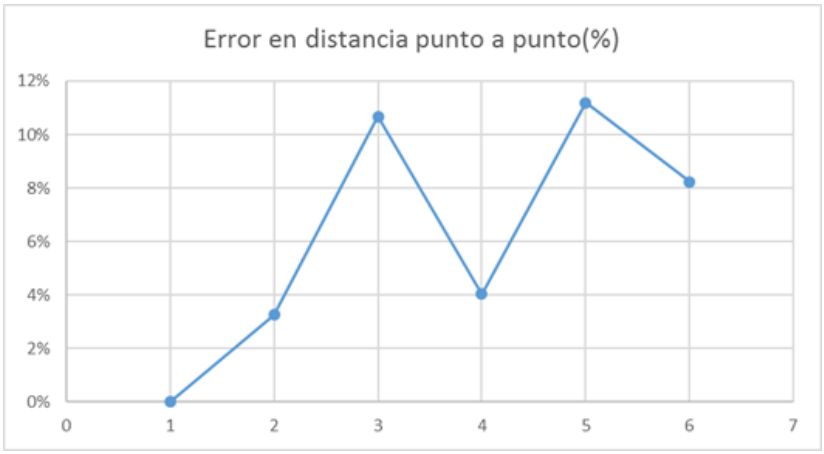


Figura 3.10. Error porcentual por cada punto de la ruta (“Mashi”).

Aunque los resultados obtenidos son diferentes que la prueba con el primer robot y tomando en consideración los nuevos factores con el control de motores, se afirma el éxito en la integración.

CONCLUSIONES Y RECOMENDACIONES

Se logró implementar un robot a escala reducida que siga una ruta planificada por el usuario, haciendo uso de dispositivos embebidos.

A pesar de no haber usado los encoders para darle la funcionalidad de giro al robot, se logró hacerlo gracias al sistema difuso y al VectorNav el cual indica la orientación del robot.

Se comprobó que el error al momento del giro no fue significativo, sin embargo este se producía debido a que existe un retardo entre el momento de generar la instrucción de detención de los motores una vez haya alcanzado el ángulo indicado, y la ejecución por parte del mismo.

El proyecto desde un principio fue pensado que fuese fácilmente integrable con otros tipos de robots, para ello se logró hacer una prueba de este supuesto integrando los módulos en la base de un robot diferente. No obstante debido a que éste hacía uso de un arduino para el control de motores, fue necesario la implementación aparte del módulo de control para este robot en específico. Llegando a la conclusión de que es posible la integración, la cual puede requerir mayor o menor complejidad dependiendo del tipo de robot a utilizar.

El sistema difuso fue probado en el “Mashi” robot, sin embargo a pesar de que los resultados fueron muy similares a los de “Milagrito”, el robot se comportó de una manera diferente, esto debido a que para que el robot pueda asignar correctamente los valores de PWM a cada motor, es necesario que exista un retardo de 1 segundo, por lo que si las consignas cambian a mayor velocidad este no tendrá tiempo para procesar la información y por ende no funcionará, a diferencia de “Milagrito” el cual respondía a estos valores en tiempo real.

Para el caso de “Milagrito” el terreno de pruebas debía de ser lo más regular posible debido a sus llantas, mientras que para el “Mashi” este no era un factor que pudiese impedir su movilización, a pesar de ello el ambiente de pruebas fue el mismo para los 2 ya que era un ambiente sin obstáculos y amplio, de tal manera que se pueda visualizar en el mapa de Google sin ningún problema.

Debido a que el control difuso no va a funcionar para todos los robots ya que dependería de muchos factores externos, el control de giro y direccionamiento debe ser implementado conforme a las especificaciones de cada robot, es decir llevar a cabo una integración de las librerías específicas provistas por el fabricante, y los módulos del seguimiento autónomo provistos en este proyecto.

Para el caso de vehículos terrestres, se recomienda la implementación de detección de obstáculos, con la finalidad de que puedan manejarse en cualquier tipo de terreno y en caso de ser posible, realizar un análisis del objeto detectado para clasificación de zonas.

BIBLIOGRAFÍA

[1] Max Roser (2016) – 'Agricultural Employment' [Online]. Disponible en: https://ourworldindata.org/data/food-agriculture/agricultural-employment/

[2] Max Roser (2016) – 'Land Use in Agriculture' [Online]. Disponible en: https://ourworldindata.org/land-use-in-agriculture/

[3]Technology readines levels T.R.L. [Online]. Disponible en: http://www.minetur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/393/NOTAS.pdf

[4] Jeffrey Hagenmeier (2014, Agosto 29) –'El Futuro ha Llegado con la Agricultura Robotizada' [Online]. Disponible en:

http://daytradingacademy.co/inversiones/agricultura-robotizada/

[5] Ana Gaona. (2013,11,27). El futuro de los robots y vehículos autónomos, ¿cuánto nos falta? [Online]. Disponible en: http://www.mediatelecom.com.mx/

[6] Parrot Bedop, Acerca del dron Parrot Bedop [Online]. Disponible en: https://www.parrot.com/fr/drones/parrot-bebop-drone#parrot-bebop-drone

[7] eBee, Acerca del dron eBee [Online]. Disponible en: https://www.sensefly.com/home.html

[8] B. Stephen Dr. (2015, Abril 17) – 'Calibration of Deterministic IMU Errors' [Online]. Disponible en: http://commons.erau.edu/pr-honors-coe/2/

[9] C.Brooks, R.Dobson, D.Banach, D. Dean, T.Oommen, R.Wolf, T.Havens, T. Ahlborn, B.Hart, "Evaluating the Use of Unmanned Aerial Vehicles for Transportation Purposes", MDOT, Houghton, Michigan, MTRI-MDOTUAV-FR-2014, Mar. 27, 2015

[10] Aurelio Morales (2015, Nov 17), 'Cómo crear un mapa con Leaflet' [Online]. Disponible en: http://mappinggis.com/2013/06/como-crear-un-mapa-con-leaflet/

[11] D. Guinea, Comportamiento autónomo de robots en entornos con complejidad e incertidumbre, Proc.: Jornadas de Robótica e Inteligencia Artificial, Alcalá de Henares, 1996, pp. 11-16.

[12] Borenstein, J., H. R. Everett, L. Feng y D. Wehe, “Mobile Robot Positioning – Sensors and Techniques”, Journal of Robotic Systems: 14(4), 231–249 (1997).

[13] Estimación de posición por odometría [Online]. Disponible en: https://cuentos-cuanticos.com/2011/12/15/robotica-estimacion-de-posicion-por-odometria/

[14] Kelly, A., “Linearized Error Propagation in Odometry, Int. Journal of Robotics Research: 23(2), 179–218 (2004).

[15] Morgan-Owen, G. J., & Johnston, G. T. (1995). Differential GPS positioning. Electronics & Communication Engineering, 7, 11-21.

[16] Yiming Chen, Sheng Zhao & Jay A. (2015) Farrell Computationally Efficient Carrier Integer Ambiguity Resolution in Multiepoch GPS/INS: A Common-Position-Shift Approach.IEEE Transactions on Control Systems Technology, PP, 1-16.

[17] S. J. Julier and J. K. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in Proc. AeroSense: 11th Int. Symp. Aerospace/Defense Sensing, Simulation and Controls, 1997, pp. 182–193.

[18] Don Josef de Mendoza y Rios, F.R.S. Recherches sur les principaux Problemes de l'Astronomie Nautique, Proc. Royal Soc., Dec 22, 1796

[19] 'Geographic Information SystemsFAQ:'(1997, Marzo 13) [Online]. Disponible en: http://www.faqs.org/faqs/geography/infosystems-faq/